SLR Code Summary Report

Name	500+ times faster than deep learning~ a case study exploring faster methods for text mining stackoverflow
Number of Coding References	8
Number of Codes Coding	2
Coverage	6.41%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	10/02/2022 11:26
Coded Text	• RQ1: Can we reproduce Fu et al.'s results for tuning SVM with differential evolution (DE)? Our DE with SVM perform no worse than Fu et al. • RQ2: How do the local models compare with global models in both tuned and untuned versions in terms model training time? Local models perform comparably to their global model counterparts, but are 570 times faster in model training time.(To be precise, that 570 figure comes fromrunning on a single core. Ifwe distribute the execution cross the eight cores of a standard laptop computer, our training times become 965 times faster.) • RQ3: How does the performance of local models compare with global models and state-of-the-art deep learner when used with SVM and KNN? Local models performance very nearly as well (within 2% F1 Score) as their global counterpart and the state-of-the-art deep Learner.
Modified on	11/02/2022 13:33
Name	500+ times faster than deep learning~ a case study exploring faster methods for text mining stackoverflow
Number of Coding References	8
Number of Codes Coding	2
Coverage	6.41%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	10/02/2022 11:26
Coded Text	Based on these experiments and discoveries, our contribution and outcome from the paper are: • A dramatically faster solution to the Stack Overflow text mining task first presented by Xu et al. This new method runs three orders of magnitude faster than prior work. • Support for "not everything needs deep learning"; i.e. sometimes, applying deep learning to a problem may not be the best approach. • Support for a simplicity-first approach; i.e. simple method like K-Means_DE_SVM can performs as good some of the state of the art models but with a (much) faster training time. • Support for local modeling. Such local models can significantly reduce training time by clustering data then restricting learning to on each cluster. • A reproduction package - which can be used to reproduce, improve or refute our results1.
Modified on	11/02/2022 13:34

Name	500+ times faster than deep learning~ a case study exploring faster methods for text mining stackoverflow
Number of Coding References	8
Number of Codes Coding	2
Coverage	6.41%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	10/02/2022 11:26
Coded Text	For example, recent results show that for finding related Stack Overflow posts, a tuned SVM performs similarly to a deep learner, but is significantly faster to train. This paper extends that recent result by clustering the dataset, then tuning every learners within each cluster. This approach is over 500 times faster than deep learning (and over 900 times faster ifwe use all the cores on a standard laptop computer). Significantly, this faster approach generates classifiers nearly as good (within 2% F1 Score) as the much slower deep learning method. Hence we recommend this faster methods since it is much easier to reproduce and utilizes far fewer CPU resources. More generally, we recommend that before researchers release research results, that they compare their supposedly sophisticated methods against simpler alternatives (e.g applying simpler learners to build local models).
Modified on	11/02/2022 13:33
Name	500+ times faster than deep learning~ a case study exploring faster methods for text mining stackoverflow
Number of Coding References	8
Number of Codes Coding	2
Coverage	6.41%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	10/02/2022 11:26
Coded Text	RQ1: Can we reproduce Fu et al.'s results for tuning SVM with differential evolution (DE)? This study uses same differential evolution with SVM for both global and local models. Thus to compare with Fu's DE with SVM as global model, the first task as part ofthis experiment was to recreate Fu et al.'s work so that this study have a baseline to measure against. Hence, this research question is a "sanity check" that must be passed before moving on to the other, more interesting research questions. The study uses the same SVM from Scikit-learn with the pa- rameters tuned as mentioned in Table 2. Here the training time of the DE+SVM model is also compared with Fu et al.'s model. Table 6 shows the class by class comparison for all the performance measure this study is using. From Table 6 it can be seen that our results with SVM with DE for hyperparameter tuning [9] [12] similar to the results of Fu et al. It can be observed from this figure that for most of the cases apart from class 3, the model has performed a little better, but the delta between the performance is very small. Hence the answer to our RQ1, is that this study has success- fully implemented Fu et al.'s SVM. Hence, we can move to more interesting questions.
Modified on	11/02/2022 13:34

Name	500+ times faster than deep learning~ a case study exploring faster methods for text mining stackoverflow
Number of Coding References	8
Number of Codes Coding	2
Coverage	6.41%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	10/02/2022 11:26
Coded Text	RQ2: How do the local models compare with global models in both tuned and untuned versions in terms model training time? For RQ2, this experiment built one model for each clusters using either normal or tuned versions ofSVM or KNN (where tuning was performed with DE): For the default SVM and KNN the experiment uses the default parameters, described in Table 1. As discussed above, this study have used the GAP statistic [33] [45] for finding the best number of clusters, using minimum and maximum number of clusters as 3 and 15, respectively. As part of the experiment we learned that 13 clusters achieves best results (measured as per the GAP statistic). This study measures the time taken for this model to train which includes time taken by GAP statistic, K-Means training time, and SVM/KNN with DE training time. Figure 6 compare the model training time in log scale of all models with the results from XU et al.'s CNN approach. Its apparent from the figure 6 that for this domain KNN and SVM has the fastest runtimes. That said, as describe below, we cannot recommend these methods since, as shown below, they achieve poor F1 Scores.
Modified on	11/02/2022 13:34

Name	500+ times faster than deep learning~ a case study exploring faster methods for text mining stackoverflow
Number of Coding References	8
Number of Codes Coding	2
Coverage	6.41%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	10/02/2022 11:26
Coded Text	RQ3: How does the performance oflocal models compare with global models and state-of-the-art Deep Learner when used with SVM and KNN? The final part of our research question was to check if the local models performance is comparable to Fu et al.'s DE_SVM and the XU's state of the art CNN. To evaluate the performance of the models this study compares F1 performance measures described in Section 3.3. As mentioned in the section, a 10 fold * 10 repeat cross validation was performed, so all the results are mean of 100 models created. Figure 7 shows our F1 Score results (mean result across all 4 class of Table 6). The numbers on top of each bar show the results of statistical tests. Bars with the same rank are statistically indistinguishable. Note that these results should be discussed with respect to the runtime results shown above:
Modified on	11/02/2022 13:34

Name	500+ times faster than deep learning~ a case study exploring faster methods for text mining stackoverflow
Number of Coding References	8
Number of Codes Coding	2
Coverage	6.41%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	10/02/2022 11:26
Coded Text	This paper further extends the Fu et al. Using very simple widely used data mining method (K-Means), we can train even faster that Fu et al. and 500 times faster than deep learning (and over 900 times faster if we use all the cores on a standard laptop computer). The core to our approach is (1) building multiple local models then (2) tuning per local model. This paper evaluates this divide and conquer approach by: (1) Exploring the Xu et al. task using SVMandK-nearest-neighbor (KNN) classifiers; (2) Repeating step 1 using hyperparameter tuning– specifically, differential Evolution (DE)– to select control parameters for those learners; (3) Repeats steps 1 and 2 using local modeling; i.e. clustering the data then apply tuning and learning to each cluster; (4) Evaluating these local models in terms of both their training time and performance
Modified on	11/02/2022 13:33

Name	500+ times faster than deep learning~ a case study exploring faster methods for text mining stackoverflow
Number of Coding References	8
Number of Codes Coding	2
Coverage	6.41%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	10/02/2022 11:25
Coded Text	Deep learning methods are useful for high-dimensional data and are becoming widely used in many areas of software engineering. Deep learners utilizes extensive computational power and can take a long time to train– making it difficult to widely validate and repeat and improve their results. Further, they are not the best solution in all domains. For example, recent results show that for finding related Stack Overflow posts, a tuned SVM performs similarly to a deep learner, but is significantly faster to train. This paper extends that recent result by clustering the dataset, then tuning every learners within each cluster. This approach is over 500 times faster than deep learning (and over 900 times faster ifwe use all the cores on a standard laptop computer). Significantly, this faster approach generates classifiers nearly as good (within 2% F1 Score) as the much slower deep learning
Modified on	11/02/2022 13:33

Name	A comprehensive study on challenges in deploying deep learning based software
Number of Coding References	20
Number of Codes Coding	3
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	06/01/2022 15:05
Coded Text	
Modified on	11/02/2022 13:25

Name	A comprehensive study on challenges in deploying deep learning based software
Number of Coding References	20
Number of Codes Coding	3
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	06/01/2022 15:05
Coded Text	7 IMPLICATIONS Based on the preceding derived findings, we next discuss our insights and some practical implications for developers, researchers, and DL framework vendors. 7.1 Researchers As demonstrated in our study
Modified on	11/02/2022 13:27

Name	A comprehensive study on challenges in deploying deep learning based software
Number of Coding References	20
Number of Codes Coding	3
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	06/01/2022 15:05
Coded Text	7.2 Developers (1) Targeted learning of required skills. DL software deployment lies in the interaction between DL and SE. Therefore, DL software deployment requires developers with solid knowledge of both fields, making this task quite challenging. Our taxonomy can serve as a checklist for developers with varying backgrounds, motivating the developers to learn necessary knowledge before really
Modified on	11/02/2022 13:27

Name	A comprehensive study on challenges in deploying deep learning based software
Number of Coding References	20
Number of Codes Coding	3
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	06/01/2022 15:05
Coded Text	7.3 Framework Vendors (1) Improving the usability ofdocumentation.As shown in our results, many developers even have difficulty in the entire procedure of deployment (i.e., how to deploy DL software). For instance, such questions account for 13.4% in mobile deployment. As described earlier, developers often complain about the poor documentation in these questions, revealing that the usability [71] of relevant documentation should be improved. Specifically, DL framework
Modified on	11/02/2022 13:28

Name	A comprehensive study on challenges in deploying deep learning based software
Number of Coding References	20
Number of Codes Coding	3
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	06/01/2022 15:05
Coded Text	To fill this knowledge gap, this paper presents a comprehensive study on understanding challenges in deploying DL software. We mine and analyze 3,023 relevant posts from Stack Overflow, a popular Q&A website for developers, and show the increasing popularity and high difficulty of DL software deployment among developers. We build a taxonomy of specific challenges encountered by developers in the process ofDL software deployment through manual inspection of 769 sampled posts and report a series ofactionable implications for researchers, developers, and DL framework vendors.
Modified on	11/02/2022 13:20

Name	A comprehensive study on challenges in deploying deep learning based software
Number of Coding References	20
Number of Codes Coding	3
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	06/01/2022 15:15
Coded Text	
Modified on	11/02/2022 13:24

Name	A comprehensive study on challenges in deploying deep learning based software
Number of Coding References	20
Number of Codes Coding	3
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	06/01/2022 15:15
Coded Text	
Modified on	11/02/2022 13:26

Name	A comprehensive study on challenges in deploying deep learning based software
Number of Coding References	20
Number of Codes Coding	3
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	06/01/2022 15:15

Coded Text	6.1 Common Challenges in Server/Cloud, Mobile, and Browser To avoid duplicate descriptions, we first present the common inner categories in Server/Cloud, Mobile, and Browser. 6.1.1 General Questions. This category shows general challenges that do not involve a specific step in the deployment process, and contains several leaf categories as follows. Entire procedure of deployment. This category refers to gen- eral questions about the entire procedure of deployment, mainly raised without practical attempts. These questions are mainly in the form of “how”, such as “how can I use that model in android for image classification” [6]. In such questions, developers often complain about the documentation, e.g., “there is no documentation given for this model” [7]. Answerers mainly handle these questions by providing existing tutorials or documentation-like information that does not appear elsewhere, or translate the jargon-heavy documentation into case-specific guidance phrased in a developer-friendly way. Compared to Server/Cloud (9.7%) and Mobile (13.4%), Browser contains relatively fewer such questions (3.2%). A possible explanation is that since DL in browsers is still in the early stage [91], developers are mainly stuck in DL’s primary usage rather than being eager to explore how to apply DL to various scenarios. Conceptual questions. This category includes questions about basic concepts or background knowledge related to DL software deployment, such as “is there any difference between these Neural Network Classifier and Neural Network in machine learning model type used in iOS ”
Modified on	11/02/2022 13:26

Name	A comprehensive study on challenges in deploying deep learning based software
Number of Coding References	20
Number of Codes Coding	3
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	06/01/2022 15:15
Coded Text	6.2 Common Challenges in Mobile and Browser 6.2.1 Data Extraction. To deploy DL software successfully, developers need to consider any stage that may affect the final performance, including data extraction. This category is observed only in Mobile and Browser, accounting for 1.7% and 3.2% of questions, respectively. This finding indicates the difficulty of extracting data in mobile devices and browsers. 6.2.2 Inference Speed. Compared to server/cloud platforms, mobile and browser platforms have weaker computing power. As a result, the inference speed of the deployed software has been a challenge in mobile devices (3.9%) and browsers (7.2%).
Modified on	11/02/2022 13:27

Name	A comprehensive study on challenges in deploying deep learning based software
Number of Coding References	20
Number of Codes Coding	3
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	06/01/2022 15:15
Coded Text	6.3 Common Challenges in Server/Cloud and Browser Environment. This category includes challenges in setting up the environment for DL software deployment, and accounts for 19.4% and 19.2% of questions in Server/Cloud and Browser, respectively. For Mobile, its environment related questions are mainly distributed in DL Library Compilation and DL Integration into Projects categories that will be introduced later. When deploying DL software to server/cloud platforms, developers need to configure various environment variables, whose diverse options make the configuration task challenging. In addition, for the server deployment, developers also need to install or build necessary frameworks such as TF Serving. Issues that occur in this phase are included in Installing/building frameworks. Similarly, when deploying DL software
Modified on	11/02/2022 13:27

Name	A comprehensive study on challenges in deploying deep learning based software
Number of Coding References	20
Number of Codes Coding	3
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	06/01/2022 15:15
Coded Text	6.4 Remaining Challenges in Server/Cloud 6.4.1 Request. This category covers challenges in making requests in the client and accounts for 13.3% of questions in Server/Cloud. For Request, developers have difficulty in configuring the request body [34], sending multiple requests at a single time (i.e., batching request) [35], getting information ofserving models via request [36], etc.
Modified on	11/02/2022 13:27

Name	A comprehensive study on challenges in deploying deep learning based software
Number of Coding References	20
Number of Codes Coding	3
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	06/01/2022 15:15
Coded Text	6.5 Remaining Challenges in Mobile 6.5.1 DL Library Compilation. This category includes challenges in compiling DL libraries for target mobile devices and covers 7.8% of questions in Mobile. Since Core ML is well supported by iOS, developers can use Core ML directly without installing or building it. For TF Lite, pre-built libraries are officially provided for developers' convenience. However, developers still need to compile TF Lite from source code by themselves in some cases (e.g., deploying models containing unsupported operators). Since the operators supported by TF Lite are still insufficient to meet developers' demand [43], developers sometimes need to register unsupported operators manually to add them into the run-time library. It may be challenging for developers who are unfamiliar with TF Lite. In addition, for compilation, developers need to configure build command lines and edit configuration files (i.e., Build configuration). Wrong configurations [44] can result in build failure or library incompatibility with target platforms.
Modified on	11/02/2022 13:27

Name	A comprehensive study on challenges in deploying deep learning based software
Number of Coding References	20
Number of Codes Coding	3
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	06/01/2022 15:15
Coded Text	6.6 Remaining Challenges in Browser Model Loading. This category includes challenges in loading DL models in browsers, being the most common challenges in browser deployment (accounting for 24.0% of questions). For browsers, TF.js provides a tf.loadLayersModel method to support loading models from local storage, Http endpoints, and IndexedDB. Among the three ways, we observe that the main challenge lies in loading from local storage (8.0%). In the official document of TF.js [50], “local storage” refers to the browser’s local storage, which is interpreted in a hyperlink [51] contained in the document as that “the stored data is saved across browser sessions.”
Modified on	11/02/2022 13:27

Name	A comprehensive study on challenges in deploying deep learning based software
Number of Coding References	20
Number of Codes Coding	3
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	06/01/2022 15:15
Coded Text	6.7 Unclear Questions Although unclear questions are not included in our taxonomy, we also manually examine them to seek for some insights. All unclear questions have no accepted answers and do not have informative discussions or question descriptions to help us determine the challenges behind the questions. Among these unclear questions, 53% report unexpected results [54] or errors [55] when making predictions using the deployed models. However, no anomalies occur at any phase before the phase of making predictions, making it rather difficult to discover the underlying challenges. In fact, various issues can result in the errors or unexpected results in this phase. Take the server deployment as an example.
Modified on	11/02/2022 13:27

Name	A comprehensive study on challenges in deploying deep learning based software
Number of Coding References	20
Number of Codes Coding	3
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	06/01/2022 15:15
Coded Text	Deep learning (DL) becomes increasingly pervasive, being used in a wide range of software applications. These software applications, named as DL based software (in short as DL software), integrate DL models trained using a large data corpus with DL programs written based on DL frameworks such as TensorFlow and Keras. A DL program encodes the network structure of a desirable DL model and the process by which the model is trained using the training data. To help developers of DL software meet the new challenges posed by DL, enormous research efforts in software engineering have been devoted. Existing studies focus on the development of DL software and extensively analyze faults in DL programs. However, the deployment of DL software has not been comprehensively studied.
Modified on	11/02/2022 13:20
Name	A comprehensive study on challenges in deploying deep learning based software
Number of Coding References	20
Number of Codes Coding	3
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	06/01/2022 15:15
Coded Text	DL software deployment. After DL software has been well validated and tested, it is ready to be deployed to different platforms for real usage. The deployment process focuses on platform adaptations, i.e., adapting DL software for the deployment platform. The most popular way is to deploy DL software on the server or cloud platforms [107]. This way enables developers to invoke services powered by DL techniques via simply calling an API endpoint. Some frameworks (e.g., TF Serving [68]) and platforms (e.g., Google Cloud ML Engine [61]) can facilitate this deployment. In addition, there is a rising demand in deploying DL software to mobile devices [102] and browsers [91]. For mobile platforms, due to their limited computing power, memory size, and energy capacity, models that are trained on PC platforms and used in the DL software cannot be deployed directly to the mobile platforms in some cases. Therefore, some lightweight DL frameworks, such as TF Lite for Android and Core ML for iOS, are specifically designed for converting pre-trained DL models to the formats supported by mobile platforms. In addition, it is a common practice to perform model quantization before deploying DL models to mobile devices, in order to reduce memory cost and computing overhead [83, 102]. For model quantization, TF Lite supports only converting model weights from floating points to 8-bit integers, while Core ML allows flexible quantization modes, such as 32 bits to 16/8/4 bits [83]. For
Modified on	24/02/2022 10:28

Name	A comprehensive study on challenges in deploying deep learning based software
Number of Coding References	20
Number of Codes Coding	3
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	06/01/2022 15:15
Coded Text	Figure 3 shows the popularity trend of deploying DL software in terms of the number of users and questions on SO. The figure indicates that this topic is gaining increasing attention, demonstrating the timeliness and urgency of this study. For deploying DL software on server/cloud platforms, we observe that users and questions increase in a steady trend. In 2017, most major vendors roll out their DL frameworks for mobile devices [102]. As a result, we can observe that both the number of users and the number of questions related to mobile deployment in 2017 increase by more than 300% compared to 2016. For deploying DL software on browsers, questions start to appear in 2018 due to the release of TF.js in 2018. As found by Ma et al. [91], DL in browsers is still at
Modified on	11/02/2022 13:25

Name	A comprehensive study on challenges in deploying deep learning based software
Number of Coding References	20
Number of Codes Coding	3
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	06/01/2022 15:15
Coded Text	Limitations ofplatforms/frameworks. This category is about limitations of relevant platforms or DL frameworks. For example, a senior software engineer working on the Google Cloud ML Platform team apologizes for the failure that a developer encounters, admitting that the platform currently does not support batch prediction [9]. Besides, some issues reflect bugs in current deployment related frameworks. For instance, an issue reveals a bug in the TocoConvert.from_keras_model_file method of TF Lite [10]. 6.1.2 Model Export andModel Conversion. Both categories cover challenges in converting DL models in DL software into the formats supported by deployment platforms. Model export directly saves the trained model into the expected format, and it is a common way for deploying DL models
Modified on	11/02/2022 13:27

Name	A comprehensive study on challenges in deploying deep learning based software
Number of Coding References	20
Number of Codes Coding	3
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	06/01/2022 15:15
Coded Text	RQ2: DIFFICULTY For deployment and other aspects (in short of non-deployment) of DL software, the percentages of relevant questions with no accepted answer (%no acc.) are 70.7% and 62.7%, respectively. The significance of this difference is ensured by the result of proportion test (2 = 78.153, df = 1, -value < 2.2e-16), indicating that questions related to DL software deployment are more difficult to answer than those related to other aspects of DL software. More specifically, for server/cloud, mobile, and browser deployment, the values of%no acc. are 69.8%, 71.6%, and 69.1%, respectively. In terms of this metric, questions about deploying DL software are also more difficult to resolve than other well-studied challenging topics in SE, such as big data (%no acc. = 60.5% [75]), concurrency (%no acc. = 43.8% [72]), and mobile (%no acc. = 55.0% [96]). Figure 4 presents the boxplot of response time needed to receive an accepted answer for deployment and non-deployment related questions.We can observe that the time needed for non-deployment questions is mostly concentrated below 600 minutes, while
Modified on	11/02/2022 13:25

Name	A comprehensive study on challenges in deploying deep learning based software
Number of Coding References	20
Number of Codes Coding	3
Coverage	11.14%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56
Coded Text	9 RELATEDWORK In this section, we summarize related studies to position our work within the literature. Challenges that ML/DL poses for SE. The rapid development ofML technologies poses new challenges for software developers. To characterize these challenges, Thung et al. [99] collect and analyze bugs in ML systems to study bug severity, efforts needed to fix bugs, and bug impacts. Alshangiti et al. [74] demonstrate that ML questions are more difficult to answer than other questions on SO and that model deployment is most challenging across all ML phases. In addition, Alshangiti et al. [74] find that DL related topics are most popular among the ML related questions. In recent years, several studies focus on the challenges in DL. By inspecting DL related posts on SO, Zhang et al. [106] find that program crashes,
Modified on	11/02/2022 13:28

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	
Modified on	11/02/2022 13:15

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	
Modified on	11/02/2022 13:17

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	
Modified on	11/02/2022 13:17

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	3.1 Data Bugs Finding 1: Data Bugs appear more than 26% of the times
Modified on	11/02/2022 13:15

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	3.2 Structural Logic Bugs Finding 2: Caffe has 43% Structural Logic Bugs
Modified on	11/02/2022 13:15

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	3.3 API Bugs Finding 3: Torch, Keras, Tensorflow have 16%, 11% and 11% API bugs respectively
Modified on	11/02/2022 13:15

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	4.1 Incorrect Model Parameter (IPS) Finding 5: IPS is the most common root cause resulting in average 24% of the bugs across the libraries
Modified on	11/02/2022 13:16

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	4.2 Structural Inefficiency (SI) Finding 6: Keras, Caffe have 25% and 37% bugs that arise from SI
Modified on	11/02/2022 13:16

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	4.3 Unaligned Tensor (UT) Finding 7: Torch has 28% of the bugs due to UT
Modified on	11/02/2022 13:16

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	4.4 Absence of Type Checking Finding 8: Theano has 30% of the bugs due to the absence of type checking
Modified on	11/02/2022 13:16

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	4.5 API Change Finding 9: Tensorflow and Keras have 9% and 7% bugs due to API change
Modified on	11/02/2022 13:16

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	4.6 Root Causes in Github Data Finding 10: Except API Misuse all other root causes have similar patterns in both Github and StackOverflow root causes ofbugs
Modified on	11/02/2022 13:16

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	4.7 Relation of Root Cause with Bug Type Finding 11: SI contributes 3% - 53% and IPS contributes 24% - 62% of the bugs related to model
Modified on	11/02/2022 13:16

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	5.1 Crash Finding 12: More than 66% of the bugs cause crash.
Modified on	11/02/2022 13:16

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	5.2 Bad Performance Finding 13: In Caffe, Keras, Tensorflow, Theano, Torch 31%, 16%, 8%, 11%, and 8% bugs lead to bad performance respectively
Modified on	11/02/2022 13:16

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	5.3 Incorrect Functionality Finding 14: 12% of the bugs cause Incorrect Functionality
Modified on	11/02/2022 13:16

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	5.4 Effects of Bugs in Github Finding 15: For all the libraries the P value for Stack Overflow and Github bug effects reject the null hypothesis to confirm that the bugs have similar effects from Stack Overflow as well as Github bugs
Modified on	11/02/2022 13:17
Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	6.1 Data Preparation Finding 16: 32% of the bugs are in the data preparation stage
Modified on	11/02/2022 13:17

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	6.2 Training Stage Finding 17: 27% of the bugs are seen during the training stage
Modified on	11/02/2022 13:17

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	6.3 Choice of Model Finding 18: Choice of model stage shows 23% of the bugs
Modified on	11/02/2022 13:17

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	8 EVOLUTION OF BUGS In this section, we explore the answer to RQ6 to understand how the bug patterns have changed over time. 8.1 Structural Logic Bugs Are Increasing Finding 20: In Keras, Caffe, Tensorflow Structural logic bugs are showing increasing trend
Modified on	11/02/2022 13:17

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	8.2 Data Bugs Are Decreasing Finding 21: Data Bugs slowly decreased since 2015 except Torch
Modified on	11/02/2022 13:17

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	Finding 4: All the bug types have a similar pattern in Github and Stack Overflow for all the libraries
Modified on	11/02/2022 13:15

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	Therefore, we study 2716 high-quality posts from Stack Overflow and 500 bug fix commits from Github about five popular deep learning libraries Caffe, Keras, Tensorflow, Theano, and Torch to understand the types of bugs, root causes of bugs, impacts of bugs, bug-prone stage of deep learning pipeline as well as whether there are some common antipatterns found in this buggy software. The key findings of our study include: data bug and logic bug are the most severe bug types in deep learning software appearing more than 48% of the times, major root causes of these bugs are Incorrect Model Parameter (IPS) and Structural Inefficiency (SI) showing up more than 43% of the times. We have also found that the bugs in the usage of deep learning libraries have some common antipatterns.
Modified on	11/02/2022 13:12

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	We have used two sources of data in our study: posts about these libraries on Stack Overflow and also Github bug fix commits. The first dataset gives us insights into bugs that developers encounter when building software with deep learning libraries. A number of these bugs would, hopefully, be fixed based on the discussion in Q&A forum. The second dataset gives us insights into bugs that were found and fixed in open source software. Our study focuses on following research questions and compares our findings across the five subject libraries. RQ1: (Bug Type) What type of bugs are more frequent? RQ2: (Root cause) What are the root causes of bugs? RQ3: (Bug Impact) What are the frequent impacts of bugs? RQ4: (Bug prone stages) Which deep learning pipeline stages are more vulnerable to bugs? RQ5: (Commonality) Do the bugs follow a common pattern? RQ6: (Bug evolution) How did the bug pattern change over time? Findings-at-a-glance. Our study show that most of the deep learning bugs are Data Bugs and Logic Bugs [5], the primary root causes that cause the bugs are Structural Inefficiency (SI) and Incorrect Model Parameter (IPS) [27], most of the bugs happen in the Data Preparation stage ofthe deep learning pipeline. Our study also confirms some of the findings of Tensorflow conducted by Zhang et al. [27]. We have also studied some antipatterns in the bugs to find whether there is any commonality in the code patterns that results in bugs. Our findings show that there is strong correlation among the distribution of bugs as well as in the antipatterns. Finally, we conclude with a discussion on our findings suggesting immediate actions and future research directions based on these findings.
Modified on	11/02/2022 13:13

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	03/02/2022 14:16
Coded Text	2.4.4 Structural Bug (SB). A vast majority of the deep learning bugs are occurring due to incorrect definitions of the deep learning model’s structure. These include mismatch of dimensions between different layers of deep learning models, the presence of anomaly between the training and test datasets, use of incorrect data structures in implementing a particular function, etc. These type of bugs can be further classified into four subcategories.
Modified on	11/02/2022 13:14

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	03/02/2022 14:16
Coded Text	2.4.5 Non Model Structural Bug (NMSB). Unlike SB, NMSB occur outside the modeling stage. In other words, this bug can happen in any deep learning stage except the modeling stage such as the training stage or the prediction stage. NMSB has similar subcategories as SB. The subcategories ofNMSB are Control and Sequence Bug, Logic Bug, Processing Bug, and Initialization Bug. We do not define Non Model Structural Data Flow Bug like Structural Data
Modified on	11/02/2022 13:14

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	03/02/2022 14:16

Coded Text	2.5 Classification of Root Causes of Bugs 2.5.1 Absence ofInterAPICompatibility. The main reason for these bugs is the inconsistency of the combination of two different kinds of libraries. For example, a user cannot directly use Numpy function in Keras because neither Tensorflow backend nor Theano backend ofKeras has the implementation of Numpy functions. 2.5.2 Absence ofType Checking. This kind of bugs involves a type mismatch problem when calling API methods. These bugs are usually mistakes related to the use of wrong type of parameters in an API. 2.5.3 API Change. The reason for these bugs is the release of the new versions of deep learning libraries with incompatible APIs. In other words, the bug happens when the new API version is not backward compatible with its previous version. For example, a user updates the new version of a deep learning library which has new API syntax; however, the user does not modify his/her code to fit with the new version, which leads to the API change bug. 2.5.4 API Misuse. This kind of bugs often arises when users use a deep learning API without fully understanding. Missing conditions can be one kind of API misuse, and this bug occurs when a usage does not follow the API usage constraints to ensure certain required conditions. Crash is the main effect of these bugs. 2.5.5 Confusion with Computation Model. These bugs happen when a user gets confused about the function of deep learning API, which leads to the misuse of the computation model assumed by the deep learning library. For instance, a user gets confused between the graph construction and the evaluation phase. 2.5.6 IncorrectModel Parameteror Structure (IPS). IPS causes problems with constructing the deep learning model, e.g. incorrect model structures or using inappropriate parameters. IPS is a common bug in the deep learning software because of both the lack of deep learning knowledge among the users and the incomprehensibilty of deep learning models. This kind of bugs causes the functional incorrectness; thus, the effect of this bug is a crash. 2.5.7 Others. These bugs are not related to deep learninng software. In other words, these bugs are mostly related to mistakes in the development process like incorrect syntax.
Modified on	11/02/2022 13:14

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	03/02/2022 14:16
Coded Text	2.5.10 WrongDocumentation. Incorrect information in library documentation leads to these bugs. Deep learning library users may face this kind of bugs when they read an incorrect definition or an incorrect usage of a deep learning API from documentation.
Modified on	11/02/2022 13:15

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	03/02/2022 14:16
Coded Text	2.5.8 Structure Inefficiency (SI). SI causes problems related to modeling stage in deep learning software like IPS; however, SI leads to bad performance of the deep learning software while IPS leads to a crash. 2.5.9 UnalignedTensor (UT). These bugs often occur in the computation graph construction phase. When a user builds the computation graph in deep learning process, they have to provide correct input data that satisfies input specifications of the deep learning API; however, many users do not know the API specifications, or they misunderstand API signature leading to UT bugs.
Modified on	11/02/2022 13:14

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	03/02/2022 14:16
Coded Text	2.6 Classification of Effects of Bugs 2.6.1 Bad Performance. Bad performance or poor performance is one of common kind of effect in deep learning software. Furthermore, the major root causes of this effect are SI or CCM that are related to the model construction. Even though developers can use deep learning libraries correctly, they still face model construction problems because APIs in these libraries are abstract. 2.6.2 Crash. Crash is the most frequent effect in deep learning. In fact, any kind of bugs can lead to Crash. A symptom of crash is that the software stops running and prints out an error message. 2.6.3 Data Corruption. This bug happens when the data is corrupted as it flows through the network. This effect is a consequence of misunderstanding the deep learning algorithms or APIs. When Data Corruption occurs, a user will receive unexpected outputs. 2.6.4 Hang. Hang effect is caused when a deep learning software ceases to respond to inputs. Either slow hardware or inappropriate deep learning algorithm can lead to Hang. A symptom of Hang is that the software runs for a long period of time without providing the desired output. 2.6.5 Incorrect Functionality. This effect occurs when the software behaves in an unexpeced way without any runtime or compile-time error/warning. This includes the incorrect output format, model layers not working desirably, etc. 2.6.6 Memory Out ofBound. Deep learning software often halts due to unavailability of the memory resources. This can be caused by, either the wrong model structure or, not having enough computing resources to train a particular model.
Modified on	11/02/2022 13:15

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	03/02/2022 14:16
Coded Text	6.2 Training Stage Finding 17: 27% of the bugs are seen during the training stage
Modified on	11/02/2022 13:17

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	03/02/2022 14:16
Coded Text	Control and Sequence Bug. This subclass of the bug is caused by the wrong structure of control flow. In many scenarios, due to wrong if-else or loop guarding condition, the model does not perform as expected. This type of bug either leads to a crash when a part of deep learning model does not work or, leads to incorrect functionality due to mishandling of data through the layers.
Modified on	11/02/2022 13:14

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	03/02/2022 14:16
Coded Text	Data Flow Bug. The main difference between the Data Flow Bug and the Data Bug is the place of origin. If a bug occurs due to the type or shape mismatch of input data after it has been fed to the deep learning model, we label it as Data Flow Bug. It includes those scenarios where model layers are not consistent because ofdifferent data shape used in consecutive layers. To fix these bugs, developers need to modify the model or reshape the data.
Modified on	11/02/2022 13:14

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	03/02/2022 14:16
Coded Text	Developers mainly rely on libraries and tools to add deep learning capabilities to their software. What kinds of bugs are frequently found in such software? What are the root causes of such bugs? What impacts do such bugs have? Which stages of deep learning pipeline are more bug prone? Are there any antipatterns? Understanding such characteristics of bugs in deep learning software has the potential to foster the development of better deep learning platforms, debugging mechanisms, development practices, and encourage the development of analysis and verification frameworks.
Modified on	11/02/2022 13:12

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	03/02/2022 14:16
Coded Text	Flow Bug because Data Bug already covers the meaning of Non Model Structural Data Flow Bug. Control and Sequence Bug. This subclass is similar to Control and Sequence Bug in SB. The bug is caused by an incorrect structure of control flow like wrong if-else condition; however, this kind of bug happens outside modeling stage. Initialization Bug. This subclass is similar to Initialization Bug in SB. The bug is caused by incorrect initialization of a parameter or a function prior to its use. Logic Bug. This subclass is similar to Logic Bug in SB. The bug is caused by misunderstanding the behavior of case statements and logical operators. Processing Bug. This subclass is similar to Processing Bug in SB. The bug is caused by an incorrect choice of algorithm.
Modified on	11/02/2022 13:14

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	03/02/2022 14:16
Coded Text	Initialization Bug. In deep learning, Initialization Bug means the parameters or the functions are not initialized properly before they are used. This type of bugs would not necessarily produce runtime error but it will simply make the model perform worse. Here, the definition of functions includes both user-defined and API defined. We also categorize a bug into this category when the API has not been initialized properly.
Modified on	11/02/2022 13:14

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	03/02/2022 14:16
Coded Text	Logic Bug. In deep learning, the logical understanding of each stage of the pipeline is an integral part of the coding process. With an incorrect logical structure of the deep learning model, the output of a program may result in either a runtime error or a faulty outcome. These bugs are often generated in the absence of proper guarding conditions in the code.
Modified on	11/02/2022 13:14

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	03/02/2022 14:16
Coded Text	Processing Bug. One of the most important decisions in the deep learning model structure is to choose the correct algorithm for the learning process. In fact, different deep learning algorithms can lead to different performance and output [14]. Also, to make different layers be compatible with each other, the data types of each layer need to follow a contract between them. Processing Bugs happen due to the violation of these contracts.
Modified on	11/02/2022 13:14

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	03/02/2022 14:16
Coded Text	those that have studied bugs in the implementation of machine learning libraries themselves, and those that have studied bugs in the usage of a specific deep learning library. A key work in the first category is Thung et al. [24] who studied bugs in the implementation ofthree machine learning systems Mahout, Lucene, and OpenNLP. In the second category, Zhang et al. [27] have studied bugs in software that make use of the Tensorflow library. While both categories of approaches have advanced our knowledge of ML systems, we do not yet have a comprehensive understanding of bugs encountered by the class of deep learning libraries.
Modified on	11/02/2022 13:12

Name	A comprehensive study on deep learning bug characteristics
Number of Coding References	41
Number of Codes Coding	2
Coverage	15.85%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	03/02/2022 14:16
Coded Text	Types of Bugs in Deep Learning Software Developers often encounter different types of bugs while trying to write deep learning software. To understand those bugs and their root causes, we have classified them into different categories. The classification is inspired from [5] and adapted based on all the Stack Overflow posts that we have analyzed. 2.4.1 API Bug. This group ofbugs is caused by deep learning APIs. Generally, when a developer uses a deep learning API, different bugs associated with that API are inherited automatically without the knowledge of the user. The prime causes for triggering of deep learning API bugs can be because of the change of API definition with different versions, lack of inter-API compatibility and sometimes wrong or confusing documentation. 2.4.2 Coding Bug. These kind of bugs originate due to programming mistakes. This in turn, introduces other types of bugs in the software which lead to either runtime error or incorrect results. A big percentage of the deep learning bugs that we have checked arises from syntactic mistakes that cannot be fixed by changing only some lines of code. This type of bugs are not identified by the programming language compiler resulting in wrong output. 2.4.3 Data Bug. This bug may arise ifan input to the deep learning software is not properly formatted or cleaned well before supplying
Modified on	11/02/2022 13:13

Name	A Dark Art~ The Machine Learning Labour Process
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	A hybrid method for missing value imputation
Number of Coding References	14
Number of Codes Coding	2
Coverage	18.86%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	
Modified on	11/02/2022 13:03

Name	A hybrid method for missing value imputation
Number of Coding References	14
Number of Codes Coding	2
Coverage	18.86%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	A hybrid method for missing value imputation PCI'19, November, 2019, Nicosia, Cyprus all estimations that are different to the mean, so it is better to be avoided in real-world research. Numerous imputation methods, involving the well-known KNN algorithm can be found in literature [29]. Imputing missing values considering their nearest neighbors, also referred to as kNN imputation (KNNI) is an easy to apply and understand approach. In this case, to make an estimation of a missing data point, the k most similar instances according to a distance metric are taking into consideration [3]. The missing values are imputed using the mean value of these instances if the attribute is numeric and the most common value if it is nominal. The greater drawback of this method is that in large datasets the algorithm has to search the whole dataset to find the nearest neighbors, so the procedure is time-consuming. A variant of this technique is weighted KNNI (WKNNI) in which a weighted mean of k nearest neighbors is used for numeric values, while categorical are imputed with the most common value
Modified on	20/06/2022 11:28

Name	A hybrid method for missing value imputation
Number of Coding References	14
Number of Codes Coding	2
Coverage	18.86%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	A widely used imputation method that can be found in libraries of the most noted statistical and Machine Learning suites is IRMI. In this work, we propose a variant of IRMI in order to maintain the advantages of this famous imputation method, while outperforming its traditional variant used in many Machine Learning software tools. To achieve this, the benefits of boosting as well as decision tree theory are exploiting. To test the efficiency of our method, a series of experiments over 30 datasets was executed, measuring the classification accuracy of the proposed method to prove that outperforms its rivals, which include classic, as well as more sophisticated imputation strategies. Finally, the results of our study are provided, along with the conclusions that arise from them.
Modified on	11/02/2022 13:00

Name	A hybrid method for missing value imputation
Number of Coding References	14
Number of Codes Coding	2
Coverage	18.86%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	Considering the first approach, ignoring the missing data in Missing completely at random (MCAR): The missing values occur completely at random and are distributed evenly among the observations. In other words, all the observations share an equal probability to be missed. The reason of missingness is not related to the observed variables or unobservable parameters of interest. In this case, missing values are a random subset of the dataset, and no other data (missing or observed) are related to them. • Missing at random (MAR): The MAR values are not related to the missing data, but are related to some of the observed data. This means that missing values are related to one or more variables of the dataset. To be more specific, a value is MAR, when the probability to be missing depends only on available information. MAR values are more common than MCAR. • Missing not at random (MNAR): The value of missing data is related to the reason of missingness. The phenomenon that all the values of an attribute are missing due to their values is referred to as censoring [13], but in real-world scenarios is extremely hard to take place. As already mentioned, it is critical to know which missing values mechanism corresponds to the data in a study. MCAR data can easily be ignored without leading to biased results if the sample size is appropriate. On the other hand, MNAR values are the hardest to manage, as even sophisticated methods tend to fail in handling them [5]. fact shifts the problem from the preprocessing stage, to the stage of the execution of the selected learner. Algorithms' implementations usually employ as a default setting one of the tradition mechanisms to treat missing values. Although it seems convenient, the user loses the opportunity to take advantage of the nature of his data and choose the proper technic. The deletion of missing values is also a trivial but risky approach. Discarding missing values from the actual dataset can be performed in different ways. One recipe is to discard all the examples that contain missing values. This can be done in cases when unobserved data consist of a small proportion of the initial dataset. When the initial dataset is already short or contains a huge proportion of unknown data (usually greater than 20%), this method should be avoided, as results in significant information loss. Another way to discard unknown values is to drop variables (features) that contain a great number on them. This approach is unreliable if the associations between the variable are not completely clear. Discarding a feature with a critical role in the final decision can lead to biased and inaccurate results
Modified on	20/06/2022 11:27

Name	A hybrid method for missing value imputation
Number of Coding References	14
Number of Codes Coding	2
Coverage	18.86%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	In the presentation of IRMI direction guidelines are given about methods that can be used according to the type of the target variable. Considering continuous and categorical values, in classic IRMI a selection of a robust regression model (Logistic algorithm is preferred usually in IRMS's software implementations) and Linear Regression are suggested respectively. Our proposed method differentiates in this part, using: M5P regression trees [22] for imputing numeric values. a boosting learner, Logitboost [10] [25], in charge of imputing categorical values, and
Modified on	11/02/2022 13:03
Name	A hybrid method for missing value imputation
Number of Coding References	14
Number of Codes Coding	2
Coverage	18.86%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	In this work, a method for imputing missing values, based on the well known IRMI, is proposed. This specific variant for IRMI exploits the advantages of both boosting and decision tree theory as employs Logitboost and M5P instead of Logistic and Linear regression as traditional IRMI does, in order to impute nominal and numeric missing values respectively. The results of the extended experimental procedure proved that our method is not only efficient as compared to four other well-known imputation strategies, but also statistical independent. In total, the results of the experimental procedure proved that considering a quite large percentage of missing values in a dataset, specifically 20% to 50% the proposed method performed better than its' four rivals. In total, 18.000 single experiments were executed (30 datasets × 10 fold cross-validation × 10 times × 6 different cases). The number of experiments, in combination with the results of the following statistical test, verify that the experimental process is reliable.
Modified on	20/06/2022 11:35

Name	A hybrid method for missing value imputation
Number of Coding References	14
Number of Codes Coding	2
Coverage	18.86%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	Learners that are not statistically independent are connected with bold horizontal lines, while statistically independent learners are not connected. In all six cases, the proposed method was statistically independent, as can easily be assumed by the provided CD plots. According to the results provided by computing the accuracy metric and the statistical test that followed, it is almost safe to conclude that our method outperforms its rivals in all six scenarios, not only by achieving high accuracy but also proving its statistical independence among them.
Modified on	11/02/2022 13:04

Name	A hybrid method for missing value imputation
Number of Coding References	14
Number of Codes Coding	2
Coverage	18.86%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	The purpose of this work is to propose an efficient imputation method for missing values based on the well-known IRMI imputation strategy, which stands for Iterative Robust Modelbased Imputation. As implies its name, the idea behind this algorithm is quite simple. To provide an estimation of a missing data value, IRMI uses the missing value as a target value and the remaining variables as regressors. This way, the entire dataset is used as a multivariate model whose final prediction is the computation of an estimation of the initial missing value. IRMI was initially introduced and described in [28] as an improvement of IVEWARE algorithm [23], while here is provided only a brief description of its function. The first step of the IRMI algorithm consists of an initialization of all missing values in the dataset, by using a simple imputation method, like k-nearest neighbors. Then the variables are sorted concerning the initial number of missing values existing in them. After that, a two-step iterative procedure takes place consisting of two nested loops. In every iteration of the inner loop, the missing values of one variable are updated, starting with the one that contains the greater amount of missing information, and moving towards the one which contains the less. In order to achieve this, the cells of the considering variable are split into two subsets: one that contains the cells with initially unknown values and a second, which contains the observed ones.
Modified on	11/02/2022 13:03

Name	A hybrid method for missing value imputation
Number of Coding References	14
Number of Codes Coding	2
Coverage	18.86%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	The second approach, imputation, is more attracting. The easiest way to perform this method is to use a specific value – a constant – to replace every missing data point. There is always the worry of using a misleading value as a filler, for example using zeros to fill all the empty cells in a dataset, which are corresponding to the annual salaries of employees. A smarter, but still easy to apply method has been proposed and used in many tasks. Instead of using a random constant, using known values in order to compute an estimation of the missing one, is preferred. The common way is to use the global average value for numerical attributes and the global most common value, i.e. the value that appears most of the time, for categorical attributes. Taking this method a step further, the same idea can be performed, considering only the instances that share the same class with the target value, in order to compute the mean or select the most common value respectively
Modified on	20/06/2022 11:28

Name	A hybrid method for missing value imputation
Number of Coding References	14
Number of Codes Coding	2
Coverage	18.86%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	To reduce the computational burden of KNNI, a method that combines it with Self-Organized Maps (SOM) has also been demonstrated [19]. This algorithm uses adaptive imputation based on belief function theory. If the class of the object can be predicted unambiguously by using only the known information, then it is committed to the class, otherwise SOR is applied to each class, so it can be represented by the corresponding weighted vectors. Then k nearest weight vectors is used to impute the values. All the aforementioned techniques share a common trait.
Modified on	20/06/2022 11:28

Name	A hybrid method for missing value imputation
Number of Coding References	14
Number of Codes Coding	2
Coverage	18.86%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	31/01/2022 14:44
Coded Text	Legitimate missing values, due to their nature are easier to deal with and in most times there is no need to employ sophisticated methods, like imputation, to deal with them. Moreover, in some cases the missing values belong to this category can provide the researchers with useful information about the reliability of the questionnaire. Unfortunately, not all missing data belong to the previous category. Illegitimately missing data can be found in all kinds of datasets and can be caused by numerous factors.
Modified on	11/02/2022 13:02

Name	A hybrid method for missing value imputation
Number of Coding References	14
Number of Codes Coding	2
Coverage	18.86%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	31/01/2022 14:44
Coded Text	Missing values are a common incurrence in a great number of real-world datasets, emerging from diverse domains of interest. In research, missing data constitute a significant problem as it can affect the conclusions drawn from them. Considering this, the difficulty of data preprocessing is increasing as selecting an inappropriate way to handle missing information can lead to untrustworthy results. Unfortunately, like in most cases in Machine Learning, there is not a single solution that fits in every task related to the problem. For this reason, many strategies have been proposed to successfully deal with this issue. One of the most well-known, besides efficient, is imputation. Replacing a missing value with an estimation apparently eliminates the problem and provides complete datasets but the difficulty shifts in selecting the right method to impute missing values.
Modified on	11/02/2022 13:00

Name	A hybrid method for missing value imputation
Number of Coding References	14
Number of Codes Coding	2
Coverage	18.86%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	31/01/2022 14:44
Coded Text	The missing data mechanism affects how missing values bias the results of a study, so it is essential to know its type in order to choose the most appropriate approach to deal with them. Missing data can be categorized into three major categories depending on the mechanism causing them [4]: • 2 Related work The stage of data preprocessing is fundamental in Machine Learning tasks as can influence remarkably the quality of the extracted results. Considering this matter of fact, it is clear why dealing with missing values is a very active research field. Although the related literature is rich and plenty of work had been done on this specific issue, unfortunately, there is not a single way that can handle every individual case that lie in this field. Missing values reside in datasets emerging from different domains, and as long as each of them has its specific features it is obvious that the nature of missing data that exist in a dataset has a principal role in the selection of the right treatment approach. The methods that already have been proposed to deal with missing values can be clustered in two categories. The first and simplest method, suggests to ignore or discard missing data. The second one suggests to replace the missing value, with a new one, or in other words to impute it. Both approaches are discussed below. Considering the first approach, ignoring the missing data in Missing completely at random (MCAR): The missing values occur completely at random and are distributed evenly among the observations. In other words, all the observations share an equal probability to be missed. The reason of missingness is not related to the observed variables or unobservable parameters of interest. In this case, missing values are a random subset of the dataset, and no other data (missing or observed) are related to them. • Missing at random (MAR): The MAR values are not related to the missing data, but are related to some of the observed data. This means that missing values are related to one or more variables of the dataset. To be more specific, a value is MAR, when the probability to be missing depends only on available information. MAR values are more common than MCAR. • Missing not at random (MNAR): The value of missing data is related to the reason of missingness. The phenomenon that all the values of an attribute are missing due to their values is referred to as censoring [13], but in real-world scenarios is extremely hard to take place.
Modified on	11/02/2022 13:02

Name	A hybrid method for missing value imputation
Number of Coding References	14
Number of Codes Coding	2
Coverage	18.86%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	31/01/2022 14:44
Coded Text	A familiar problem to Machine Learning researchers and data analysts is the occurrence of missing data, which reside in almost every dataset, setting obstacles in the stage of data preprocessing. Missing data, also referred as missing values, occur when no value is stored for the variable of an attribute, leaving the actual value of the observation unknown. The most common scenario is that multiple values, usually in different attributes are missing in a single dataset, leading to the absence of a not negligible subset of it. The fact that a portion of the actual dataset is missing means that the amount of information that can be drawn from the data is reduced, a mater that strongly affects the ability to understand and explain the phenomenon of interest and raises concern about the reliability of the study results [1].
Modified on	11/02/2022 13:00
Name	A Machine Learning Based Framework for Verification and Validation of Massive Scale Image Data
Number of Coding References	5
Number of Codes Coding	1
Coverage	3.66%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	
Modified on	11/02/2022 12:50

Name	A Machine Learning Based Framework for Verification and Validation of Massive Scale Image Data
Number of Coding References	5
Number of Codes Coding	1
Coverage	3.66%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	
Modified on	11/02/2022 12:51

Name	A Machine Learning Based Framework for Verification and Validation of Massive Scale Image Data
Number of Coding References	5
Number of Codes Coding	1
Coverage	3.66%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03

Coded Text	Abstract—Big data validation and system verification are crucial for ensuring the quality of big data applications. However, a rigorous technique for such tasks is yet to emerge. During the past decade, we have developed a big data system called CMA for investigating the classification of biological cells based on cell morphology which is captured in diffraction images. CMA includes a collection of scientific software tools, machine learning algorithms, and a large-scale cell image repository. In order to ensure the quality of big data system CMA, we developed a framework for rigorously validating the massive scale image data as well as adequately verifying both the software tools and machine learning algorithms. The validation of big data is conducted by iteratively selecting the data using a machine learning approach. An experimental approach guided by a feature selection algorithm is introduced in the framework to select an optimal feature set for improving the machine learning performance. The verification of software and algorithms is developed on the iterative metamorphic testing approach due to the non-testable property of the software and algorithms. A machine learning approach is introduced for developing test oracles iteratively to ensure the adequacy of the test coverage criteria. Performance of the machine learning algorithm is evaluated with a stratified N-fold cross validation and confusion matrix. We describe the design of the proposed big data verification and validation framework with CMA as the case study, and demonstrate its effectiveness through verifying and validating the dataset, the software and the algorithms in CMA.
Modified on	11/02/2022 12:50

Name	A Machine Learning Based Framework for Verification and Validation of Massive Scale Image Data
Number of Coding References	5
Number of Codes Coding	1
Coverage	3.66%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	The focus of research presented in this paper is on the validation and verification of data analytics software and algorithms as well as big data. To achieve the best validation and verification performance, feature representation, feature extraction and feature selection for machine learning used in the framework are also discussed. The verification and validation framework proposed in this paper is illustrated in Fig. 1, which includes tasks in three layers. The foundation layer provides techniques for big data validation through automated selection and classification of big data. The middle layer features an approach for verification and validation of machine learning algorithms including feature representation, extraction and optimization. Lastly, the top layer provides an approach for testing domain modeling systems, data analytics tools and applications. The framework covers the essential verification and validation tasks that are needed for any big data application,
Modified on	11/02/2022 12:50

Name	A Machine Learning Based Framework for Verification and Validation of Massive Scale Image Data
Number of Coding References	5
Number of Codes Coding	1
Coverage	3.66%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	The verification and validation of the scientific software in CMA is conducted with an iterative metamorphic testing [7], which is a metamorphic testing extended with iterative development of test oracles [8]. One major component of CMA is a collection of scientific software for supporting scientific investigation and decision making [9]. For example, 3D structure reconstruction software and light scattering modeling software are two such pieces of software in CMA. Many scientific software systems are non-testable because of the absence of test oracles [7], [9]. Metamorphic testing [7], [10] is a novel software testing technique and a promising approach for solving oracle problems. It creates tests according to metamorphic relation (MR) and verifies the predictable relation among the outputs of the related tests.
Modified on	11/02/2022 12:51

Name	A Meta Learning Approach for Automating Model Selection in Big Data Environments using Microservice and Container Virtualization Technologies
Number of Coding References	17
Number of Codes Coding	6
Coverage	11.52%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	
Modified on	11/02/2022 12:44

Name	A Meta Learning Approach for Automating Model Selection in Big Data Environments using Microservice and Container Virtualization Technologies
Number of Coding References	17
Number of Codes Coding	6
Coverage	11.52%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38

Coded Text	In the following, some meta learning approaches developed as frameworks with wizard [12][26][8][5][11] are discussed. A parallelized, component-based, modular and easily extendable meta learning system for univariate and multivariate time series load forecasting can be found in [12]. Matijaš et. al. [12] built the meta learner as an ensemble method. As meta features, minimum, maximum, Standard Deviation (SD), skewness, to name a few, were considered. Auto-WEKA [26] is a framework for automatically selecting classi□ers and hyperparameters implemented in WEKA. In the updated version Auto-WEKA 2.0 [8], they also supported regression algorithms and a more tightly integration with WEKA. Auto-Sklearn [5] is a meta learning framework based on scikitlearn which uses the same principles as Auto-WEKA. To solve the Combined Algorithm Selection and Hyperparameter optimization (CASH) problem, they built on the research from Auto-WEKA and used the same Sequential Model based Algorithm Con□guration (SMAC) algorithm as Bayesian optimizer for hyperparameter tuning. The drawback in Auto-WEKA and Auto-Sklearn is that they are implemented as monolithic applications which limit the scalability and increase the di□culty of maintenance. Moreover, they did not provide the possibility to handle model selection for large amount of data. SmartML [11] is a meta learning framework based on the R language. It is implemented as web application with REST APIs. SmartML can recommend a classi□cation algorithm, including hyperparameter tuning based on a total of 25 meta features. The limitation here is also that SmartML does not support Big Data environment for large scale processing. In contrast to the aforementioned works, the current framework in the present paper is implemented as a microservice architecture to increase the scalability and facilitate maintainability. Moreover, the utilization of a powerful Big Data stack gives the ability to perform model selection for large amount of data.
Modified on	11/02/2022 12:44

Name	A Meta Learning Approach for Automating Model Selection in Big Data Environments using Microservice and Container Virtualization Technologies
Number of Coding References	17
Number of Codes Coding	6
Coverage	11.52%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	Building a ML model by nonexpert users is a complex, time consuming and error prone process. According to the no-free-lunch theorem [31], no single learning algorithm has always the lowest performance error on a broad problem domain. As a result, for a given usage scenario, a dedicated learning algorithm must be selected. The selection process is de ned as an Algorithm Selection Problem (ASP) [18], in which the selection of the best learning algorithm for a given application scenario can be long lasting and has a high computational complexity due to the size of the search space of possible algorithm candidates.
Modified on	15/06/2022 13:08
Name	A Meta Learning Approach for Automating Model Selection in Big Data Environments using Microservice and Container Virtualization Technologies
Number of Coding References	17
Number of Codes Coding	6
Coverage	11.52%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	07/02/2022 14:10
Coded Text	Building a ML model by nonexpert users is a complex, time consuming and error prone process. According to the no-free-lunch theorem [31], no single learning algorithm has always the lowest performance error on a broad problem domain. As a result, for a given usage scenario, a dedicated learning algorithm must be selected. The selection process is de ned as an Algorithm Selection Problem (ASP) [18],
Modified on	11/02/2022 12:42

Name	A Meta Learning Approach for Automating Model Selection in Big Data Environments using Microservice and Container Virtualization Technologies
Number of Coding References	17
Number of Codes Coding	6
Coverage	11.52%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	07/02/2022 14:10
Coded Text	For a given speci c machine learning task, very often several machine learning algorithms and their right con gurations are tested in a trial-and-error approach, until an adequate solution is found. This wastes human resources for constructing multiple models, requires a data analytics expert and is time-consuming, since a variety of learning algorithms are proposed in literature and the non-expert users do not know which one to use in order to obtain good performance results. Meta learning addresses these problems and supports non-expert users by recommending a promising learning algorithm based on meta features computed from a given dataset. In the present paper, a new generic microservice-based framework for realizing the concept ofmeta learning in Big Data environments is introduced. This framework makes use of a powerful Big Data software stack, container visualization, modern web technologies and a microservice architecture for a fully manageable and highly scalable solution. In this demonstration and for evaluation purpose, time series model selection is taken into account. The performance and usability of the new framework is evaluated on state-of-the-art machine learning algorithms for time series forecasting: it is shown that the proposed microservice-based meta learning framework introduces an excellent performance in assigning the adequate forecasting model for the chosen time series datasets.
Modified on	25/02/2022 10:07

Name	A Meta Learning Approach for Automating Model Selection in Big Data Environments using Microservice and Container Virtualization Technologies
Number of Coding References	17
Number of Codes Coding	6
Coverage	11.52%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	07/02/2022 14:10
Coded Text	Meta learning solves the problem of automated model selection by formulating it as a supervised learning task incorporating training and testing phases. The main task of the training is that an algorithm –referred to meta learner– learns the mapping between available learning algorithms and measurable properties –referred to meta features– of the task itself. To this end, a set of meta examples is needed. Each example is tagged by predictors and labels. The predictors correspond to the meta features extracted to describe the datasets. The labels indicate the most appropriate algorithms. In the testing phase, given a new dataset, the meta features will be extracted to be used by the trained meta learner for suggesting the most appropriate model.
Modified on	11/02/2022 12:42

Name	A Meta Learning Approach for Automating Model Selection in Big Data Environments using Microservice and Container Virtualization Technologies
Number of Coding References	17
Number of Codes Coding	6
Coverage	11.52%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	For a given speci c machine learning task, very often several machine learning algorithms and their right con gurations are tested in a trial-and-error approach, until an adequate solution is found. This wastes human resources for constructing multiple models, requires a data analytics expert and is time-consuming, since a variety of learning algorithms are proposed in literature and the non-expert users do not know which one to use in order to obtain good performance results. Meta learning addresses these problems and supports non-expert users by recommending a promising learning algorithm based on meta features computed from a given dataset. In the present paper, a new generic microservice-based framework for realizing the concept ofmeta learning in Big Data environments is introduced. This framework makes use of a powerful Big Data software stack, container visualization, modern web technologies and a microservice architecture for a fully manageable and highly scalable solution. In this demonstration and for evaluation purpose, time series model selection is taken into account. The performance and usability of the new framework is evaluated on state-of-the-art machine learning algorithms for time series forecasting: it is shown that the proposed microservice-based meta learning framework introduces an excellent performance in assigning the adequate forecasting model for the chosen time series datasets. Moreover, the recommendation of the most appropriate forecasting model results in a well acceptable low overhead demonstrating that the framework can provide an e cient approach to solve the problem of model selection in context of Big Data.
Modified on	11/02/2022 12:41

Name	A Meta Learning Approach for Automating Model Selection in Big Data Environments using Microservice and Container Virtualization Technologies
Number of Coding References	17
Number of Codes Coding	6
Coverage	11.52%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	The third layer, namely persistence and processing layer, pro- vides the basic model and data storage capabilities according to the underlying runtime computer infrastructure and provides generic interfaces for executing and managing ML jobs on this infrastructure independent of the used low level ML framework. While the current implementation only supports Apache Spark as ML framework, the persistence and processing layer is designed in a way that supports plugging in additional ML frameworks in the future. In the following, the layers will be described in more detail.
Modified on	15/06/2022 13:26

Name	A Meta Learning Approach for Automating Model Selection in Big Data Environments using Microservice and Container Virtualization Technologies
Number of Coding References	17
Number of Codes Coding	6
Coverage	11.52%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	whereby models can be seen as special data objects. The other service focuses on the management ofrunning ML jobs e.g. for training and testing. The framework described in this paper extends the existing microservice architecture with three new services added to the service layer, namely, data preprocessing, meta knowledge extraction and meta learning services to support non-expert users in selecting an adequate MLmodel for a given ML task. The services provide RESTful APIs which are used by the web applications in the UI layer to interact with the runtime environment.
Modified on	15/06/2022 13:25

Name	A Meta Learning Approach for Automating Model Selection in Big Data Environments using Microservice and Container Virtualization Technologies
Number of Coding References	17
Number of Codes Coding	6
Coverage	11.52%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	3.0.2 Service layer. The main advantage of using microservices, apart from their modularity, is that each component of the framework can be developed separately using e.g. d erent technologies and programming languages. This layer abstracts the interface of the UI applications to the ML runtime environment (e.g. computing cluster or single computer, etc.) by providing generic interfaces to the runtime environment via ve microservices, namely the Job Management Service (J.M.-Service), the Data Management Service (D.M.-Service), Data Preprocessing Service (D.P.-Service), Meta Knowledge Extraction Service (M.K.E.-Service) and Meta Learning Service (M.L.-Service) as shown in Figure 1. Our framework instruments Kubernetes, container orchestration technique for automating deployment, management and scaling of our microservices.
Modified on	11/02/2022 12:46

Name	A Meta Learning Approach for Automating Model Selection in Big Data Environments using Microservice and Container Virtualization Technologies
Number of Coding References	17
Number of Codes Coding	6
Coverage	11.52%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	3.0.3 Persistence and Processing Layer. It communicates with the service layer to perform ML tasks on cluster hiding the low level details of the runtime environment from the implementation ofthe services [23]. The services use generic functions implemented in this layer to interface with the job runtime directory in HDFS, the database infrastructure and the other components installed in Big Data environment.
Modified on	11/02/2022 12:47

Name	A Meta Learning Approach for Automating Model Selection in Big Data Environments using Microservice and Container Virtualization Technologies
Number of Coding References	17
Number of Codes Coding	6
Coverage	11.52%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03

Coded Text	Data Management UI: It allows the uploading, management, and con□guration of data sources which provide data to ML jobs. In this UI, an interactive visualization besides statistical analysis can be performed on the datasets to achieve a better understanding of their characteristics and properties. Model Management UI: It provides a tabular overview to list and manage the models that are (eventually) already pre-trained in the framework. Each model is described by some metadata (e.g. id, creation date, model name, a textual description of what the model does, etc.). Each row in the tabular
Modified on	11/02/2022 12:46

Name	A Meta Learning Approach for Automating Model Selection in Big Data Environments using Microservice and Container Virtualization Technologies
Number of Coding References	17
Number of Codes Coding	6
Coverage	11.52%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	In the following, some meta learning approaches developed as frameworks with wizard [12][26][8][5][11] are discussed. A parallelized, component-based, modular and easily extendable meta learning system for univariate and multivariate time series load forecasting can be found in [12]. Matijaš et. al. [12] built the meta learner as an ensemble method. As meta features, minimum, maximum, Standard Deviation (SD), skewness, to name a few, were considered. Auto-WEKA [26] is a framework for automatically selecting classi□ers and hyperparameters implemented in WEKA. In the updated version Auto-WEKA 2.0 [8], they also supported regression algorithms and a more tightly integration with WEKA. Auto-Sklearn [5] is a meta learning framework based on scikitlearn which uses the same principles as Auto-WEKA. To solve the Combined Algorithm Selection and Hyperparameter optimization (CASH) problem, they built on the research from Auto-WEKA and used the same Sequential Model based Algorithm Con□guration (SMAC) algorithm as Bayesian optimizer for hyperparameter tuning. The drawback in Auto-WEKA and Auto-Sklearn is that they are implemented as monolithic applications which limit the scalability and increase the di□culty of maintenance. Moreover, they did not provide the possibility to handle model selection for large amount of data. SmartML [11] is a meta learning framework based on the R language. It is implemented as web application with REST APIs. SmartML can recommend a classi□cation algorithm, including hyperparameter tuning based on a total of 25 meta features. The limitation here is also that SmartML does not support Big Data environment for large scale processing. In contrast to the aforementioned works, the current framework in the present paper is implemented as a microservice architecture to increase the scalability and facilitate maintainability. Moreover, the utilization of a powerful Big Data stack gives the ability to perform model selection for large amount of data.
Modified on	11/02/2022 12:44
Name	A Meta Learning Approach for Automating Model Selection in Big Data Environments using Microservice and Container Virtualization Technologies
Number of Coding References	17
Number of Codes Coding	6
Coverage	11.52%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	The proposed framework carries out these tasks based on □ve decoupled and cohesive microservices. Each microservice is a small and self contained application that can be deployed independently, e.g. on the runtime cluster with a single responsibility. In [23], Shahoud et al. developed two microservices to perform ML tasks on cluster. One service focused on data and model management,
Modified on	11/02/2022 12:45

Name	A Meta Learning Approach for Automating Model Selection in Big Data Environments using Microservice and Container Virtualization Technologies
Number of Coding References	17
Number of Codes Coding	6
Coverage	11.52%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	The third layer, namely persistence and processing layer, pro- vides the basic model and data storage capabilities according to the underlying runtime computer infrastructure and provides generic interfaces for executing and managing ML jobs on this infrastructure independent of the used low level ML framework. While the current implementation only supports Apache Spark as ML framework, the persistence and processing layer is designed in a way that supports plugging in additional ML frameworks in the future. In the following, the layers will be described in more detail.
Modified on	11/02/2022 12:45

Name	A Meta Learning Approach for Automating Model Selection in Big Data Environments using Microservice and Container Virtualization Technologies
Number of Coding References	17
Number of Codes Coding	6
Coverage	11.52%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	whereby models can be seen as special data objects. The other service focuses on the management ofrunning ML jobs e.g. for training and testing. The framework described in this paper extends the existing microservice architecture with three new services added to the service layer, namely, data preprocessing, meta knowledge extraction and meta learning services to support non-expert users in selecting an adequate MLmodel for a given ML task. The services provide RESTful APIs which are used by the web applications in the UI layer to interact with the runtime environment.
Modified on	11/02/2022 12:45

Name	A Meta Learning Approach for Automating Model Selection in Big Data Environments using Microservice and Container Virtualization Technologies
Number of Coding References	17
Number of Codes Coding	6
Coverage	11.52%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56
Coded Text	in which the selection of the best learning algorithm for a given application scenario can be long lasting and has a high computational complexity due to the size of the search space of possible algorithm candidates. To face this challenge, meta learning approaches and frameworks were proposed for solving ASPs. The majority of these approaches studied the selection of classi□cation algorithms [2, 5, 17], to name a few. With growing popularity of regression, the □rst use of meta learning for time series datasets was by Ludmir et. al. in [16] who proposed a methodology for time series model selection using NOEMON approach. As meta features, a set of10 meta features including simple, statistics and time series meta features were extracted to describe time series datasets. Wang et. al. in [29], proposed a meta learning framework to recommend the most appropriate forecasting method. As forecasting algorithms, random walk, exponential smoothing, neural networks and ARIMA were chosen as candidates to be selected. Only nine time series meta features were extracted to describe time series datasets. As meta learner, a decision tree algorithm was used. The same group ofmeta features has been later used by Widodo in [30]. The di□erence is that the author tried to reduce the dimensionality by applying Principal Component Analysis (PCA). Kück et. al. [9] used feedforward neural networks as meta learner to select the best time series forecasting model for 78 time series from the NN3 competition. As algorithm candidates, single, seasonal, seasonal-trend and trend exponential smoothing were used. For characterizing time series datasets, error-based features and statistical tests were used as meta features.
Modified on	11/02/2022 12:43

Name	A Novel Concept Drift Detection Method for Incremental Learning in Nonstationary Environments
Number of Coding References	5
Number of Codes Coding	1
Coverage	8.27%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 20:46
Coded Text	A. Brief Introduction to OS-ELM ELM [43] is a training method originally proposed for a single hidden layer feedforward neural networks. Since the input weights and biases of an ELM model, (ai , bi), i = 1, 2, ... , Nh, with Nh indicating the number of neurons in the hidden layer, are randomly generated and the output weights β =[βi, j], i = 1, 2, ... , Nh, j = 1, 2, ... , S, are analytically determined, the time and computational burden necessary to train an ELM model have been shown to be significantly lower than those for training a BPNN [46]. OS-ELM [48] is an ELM variant for online applications, characterized by the fact that model updating can be performed each time a new batch of data becomes available, without any constraint on its size. The effect of OS-ELM updating is to modify only the output weights β, whereas it keeps unmodified input weights ai and biases bi , i = 1, 2, ... , Nh. Since OS-ELM updating is fast and computationally not demanding, it is quite suitable for online concept drift detection. B. Measure of Dissimilarity Between OS-ELM Models Since an OS-ELM model is characterized by its output weights β a measure of dissimilarity between the baseline model M*(t1) trained using the data collected until time t1 and the model M*(t1 +~t) updated considering the data collected between times t1 and t1 + ~t, is the distance d(βt1 between the matrices βt1 , βt1+~t) and βt1+~t (e.g., Euclidean, Mahalanobis, Chebychev, and Cosine distance) [49]. In this paper, the Euclidean distance
Modified on	28/02/2023 14:44

Name	A Novel Concept Drift Detection Method for Incremental Learning in Nonstationary Environments
Number of Coding References	5
Number of Codes Coding	1
Coverage	8.27%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 20:46
Coded Text	Approaches for adapting ML models to concept drifts are typically categorized into passive and active [14]. Passive approaches adapt the model every time new data become available. Despite the fact that these approaches are effective, they are difficult to apply due to the large computational efforts required for model updating. Contrarily, active approaches aim at balancing the accuracy and updating burden by modifying the ML model only when the occurrence of a concept drift is detected. Most of the proposed active approaches are unable to deal with all the types of concept drifts, but only with a subset of them [16]. Active approaches are typically classified into three broad categories [16]: 1) sequential analysis-based; 2) data distribution-based; and 3) learner output-based. Sequential analysis-based approaches analyze the newly acquired patterns one by one until the probability of observing the data sequence under a new distribution p1 is significantly larger than that under the original distribution p0 [17], [18]. For example, the sequential probability ratio test [6], [19], [20] uses as drift detection index the logarithm of the ratio of the likelihood of the two distributions p0 and p1. Similarly, cumulative sum (CUSUM) [21] detects a drift when the CUSUM of the data sequence significantly deviates from its mean value. The Page-Hinkley test [21] uses as an index for the drift detection the cumulative difference between the observed values and their mean until the current time. In [22], the PageHinkley test is used to detect sudden drifts in the average of a Gaussian signal. The hierarchical intersection of confidence intervals-based change detection test (H-ICI-based CDT) and its variants are designed with a two-level hierarchical structure [23]–[25]. At the first level, the ICI-based CDT monitors the stationarities of the features extracted from the data stream (e.g., mean and variance) and detects concept drifts by using ICI rules. At the second level, the concept drifts detected at the first level are validated using hypothesis tests, such as the Hotelling's T-square statistic [25], to reduce false alarms. The H-ICI-based CDT has been successfully applied to detect sudden and incremental drifts in [24] and [25]. Common drawbacks of most sequential analysis-based drift detection approaches are: 1) the detection delay caused by the fact that the variation of the considered statistical metrics becomes significant only when enough data from the new concept are collected and 2) the necessity of applying them to univariate series, which limits their use in case ofmultivariate time series. Data distribution-based drift detection approaches typically consider distributions of raw data patterns from two different time windows: a fixed window containing information of the past time series behavior and a sliding window containing the most recent data [26]. Hypothesis tests are used to compare the data distributions in the two windows and detect concept drifts when they are significantly different. They are further classified into parametric and nonparametric tests. The former assumes that the data stream probability distribution functions (PDFs) before and after the concept drift occurrence are known, but their parameters are unknown. Contrarily, multivariate nonparametric tests do not require the knowledge of the data stream PDFs, which can be difficult to know in real-world problems, but they are usually less sensitive to small differences among the two PDFs [27]. The semiparametric log-likelihood (SPLL) detector uses the k-means method to cluster raw data patterns and models the obtained clusters using a mixture of Gaussian distributions. It has been successfully applied to sudden drift detection [28]. Lu et al. [29], [30] have modeled the raw data distribution using a competence model and have shown significant detection accuracy improvement in an application involving a case-base maintenance system. Liu et al. [31] proposed the nearest neighbor-based density variation identification to consider the regional density changes. In fuzzy competence model (FCM) drift detection [32], an FCM is used to represent the data empirical distribution, which provides a degree of concept drift. In [33], the QuantTree algorithm is proposed to model the empirical distribution of high-dimensional raw data using histograms and computing statistics from obtained nonparametric representation. Common drawbacks of these approaches are the need to store data in the two windows and the setting of the window lengths.

Modified on	28/02/2023 14:43
Name	A Novel Concept Drift Detection Method for Incremental Learning in Nonstationary Environments
Number of Coding References	5
Number of Codes Coding	1
Coverage	8.27%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 20:46
Coded Text	Learner output-based drift detection approaches are based on the development of a learner (classifier) and the tracking of its error rate fluctuations. In the drift detection method (DDM) [34], the classification error is represented by a Bernoulli random variable, and the concept drift is detected when the error rate exceeds a properly set threshold. The method has provided satisfactory performance in case of sudden drifts [35]. The early drift detection method (EDDM) [36], which extends DDM by considering the time distance between errors, provides better performances in case of slow gradual concept drifts. Statistical test of equal proportions (STEPD) [37] computes the accuracy of the learner on a recent window and compares it to its overall accuracy from the beginning. STEPD is shown very accurate in the detection of sudden concept drifts in [35]. Adaptive window (ADWIN) [38] monitors the distribution change of the learner output using a variable-length window obtained by partitioning the data sequence into two disjoint subsets and, then, evaluating possible statistical discrepancies between them. When a concept drift is detected, ADWIN shrinks the window to include only the data containing the new concept for the next detection. ADWIN has been shown able to provide satisfactory performances in cases of gradual and incremental drifts [35]. Exponentially weighted moving average (EWMA) for concept drift detection (ECDD) [39] uses two estimations computed by performing EWMA with different weights: one gives more weight to recent examples, while the other gives similar weights to recent and old data. A drift is detected when the difference between the two EWMA estimations exceeds a certain threshold. ECDD has been shown able to detect gradual concept drifts in [35]. DDMs Based on Hoeffding's (HDDMA and HDDMW) bounds [40] do not require assumptions on the probability density function of the learner classification error and perform well for sudden (HDDMA) and gradual (HDDMW) drifts detection, respectively. Detailed comparisons among various learner output-based drift detection approaches can be found in [35]. Learner output-based detection methods have shown more suitable for concept drift detection than sequential analysisbased and distribution-based methods in industrial applications, such as fault detection, diagnostics, and weather prediction [41], [42], in which it is required to deal with multivariate time series characterized by complex, nonlinear relationships. Since sequential analysis-based methods detect concept drifts on single time series, they cannot be used for real drifts. Although distribution-based methods can deal with multivariate data, they may detect concept drifts of variables irrelevant to the learner output, and, therefore, cause unnecessary ML model adaptations. A criticality of learner outputbased methods is that they have typically been developed considering classification problems only and not regression problems. Another limitation of learner output-based methods is that when the occurrence of a concept drift is detected, they do not provide information about the amount of data necessary to adapt the learner to the new environment.
Modified on	28/02/2023 14:44

Name	A Novel Concept Drift Detection Method for Incremental Learning in Nonstationary Environments
Number of Coding References	5
Number of Codes Coding	1
Coverage	8.27%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 20:46
Coded Text	The main steps of the method are (Fig. 1) as follows. 1) The development of two OS-ELM models, which receive input xt and provide output yt . The first model, which is referred to as the baseline model M*(t1),is trained using data collected until time t1 and represents the old system behavior. The second model, which is referred to as an updated model M*(t1 + ~t), is obtained by updating the first model with the information acquired in the time interval [t1 + 1, t1 +~t]. Note that the training of models M*(t1) and M*(t1 +~t) requires much lower computational burden than that of the ML model M. 2) The quantification of the degree of dissimilarity between the OS-ELM models M*(t1) and M*(t1 +~t). 3) The setting of a dynamic threshold on the degree of dissimilarity for concept drift detection. 4) The identification of the proper amount of data necessary for updating M and M*, in case of the concept drift detection. Section III-A briefly recalls the basics of OS-ELMs and Section III-B defines the dissimilarity measure among OS-ELM models, whereas Section III-C describes the procedure used for setting the threshold used for the concept drift detection. Section III-D illustrates the procedure for determining the amount of data to be collected from the new concept to update the ML model. Section III-E analyzes the computational cost of the proposed method. Finally, Section III-F discusses how to set the parameters of the proposed method.
Modified on	28/02/2023 14:44

Name	A Novel Concept Drift Detection Method for Incremental Learning in Nonstationary Environments
Number of Coding References	5
Number of Codes Coding	1
Coverage	8.27%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 20:46
Coded Text	We present a novel method for concept drift detec- tion, based on: 1) the development and continuous updating of online sequential extreme learning machines (OS-ELMs) and 2) the quantification of how much the updated models are modified by the newly collected data. The proposed method is verified on two synthetic case studies regarding different types of concept drift and is applied to two public real-world data sets and a real problem of predicting energy production from a wind plant. The results show the superiority of the proposed method with respect to alternative state-of-the-art concept drift detection methods. Furthermore, updating the prediction model when the concept drift has been detected is shown to allow improving the overall accuracy of the energy prediction model and, at the same time, minimizing the number of model updatings.
Modified on	28/02/2023 14:43

Name	A Novel Concept Drift Detection Method for Incremental Learning in Nonstationary Environments Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	A Quality-driven Machine Learning Governance Architecture for Self-adaptive Edge Clouds
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	A Quality-driven Machine Learning Governance Architecture for Self-adaptive Edge Clouds Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	A Survey of Software Quality for Machine Learning Applications
Number of Coding References	8
Number of Codes Coding	1
Coverage	6.24%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	2) Fault Localization: Wong et al. surveyed software fault localization [44]. Software fault localization, the act of identifying the locations of faults in a program, is widely recognized as one of the most tedious, time consuming, and expensive activities in program debugging. Fault localization is challenging for ML applications [59]. Le et al. proposed a new fault localization approach that employs a learning-torank strategy. The approach includes ranking methods based on their likelihood of being a root cause of a failure. Sun et al. investigated several coverage-based statistical fault localization metrics and reported on an empirical study they conducted to assess the relative importance of those metrics elements [37].
Modified on	11/02/2022 12:27

Name	A Survey of Software Quality for Machine Learning Applications
Number of Coding References	8
Number of Codes Coding	1
Coverage	6.24%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	3) Prediction: Prediction and inference are important func- tions of applications with ML applications. As software systems increase in complexity and operate with less human supervision, it becomes more difficult to use traditional techniques to detect when software is not behaving as intended [62]. For instance, predicting human behavior in front of software systems helps better design software. Katz et al. proposed techniques to model the behavior of executing programs
Modified on	11/02/2022 12:27

Name	A Survey of Software Quality for Machine Learning Applications
Number of Coding References	8
Number of Codes Coding	1
Coverage	6.24%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	4) MLaaS: Ribeiro et al. defined MLaaS as a ready-to-use ML functions. They proposed an architecture to create a flexible and scalable MLaaS [32]. VanDerHerten et al. developed a proof of concept for adaptive modeling and sampling methodologies for Internet-of-Things applications with MLaaS [69]. Assem et al. developed a methodology to build an application with train, compare, decide, and change approach with MLaaS [79]. This is more practical than theoretical; however, it seems to be the most helpful reference in the real world. For concrete
Modified on	11/02/2022 12:27

Name	A Survey of Software Quality for Machine Learning Applications
Number of Coding References	8
Number of Codes Coding	1
Coverage	6.24%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	5) Multi Agent: Multi agent is a key approach to adapt soft- ware engineering into ML applications such as autonomous vehicles and robots. Verifying a data-aware multi-agent system (DAMAS) is challenging because of the infinite state models generated by their infinite-domain variables. Belardinelli et al. proposed a method of parameterized DAMAS (P-DAMAS) as a system with an unbounded number of homogenous agents, each assumed to be data-aware, i.e., endowed with possibly infinite domains and interacting with an environment composed of partially shared data [53]. Wooldridge et al. presented formal models through which rational verification can be studied, and surveyed the complexity of key decision problems in multi-agent systems [4].
Modified on	11/02/2022 12:27

Name	A Survey of Software Quality for Machine Learning Applications
Number of Coding References	8
Number of Codes Coding	1
Coverage	6.24%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	6) Search-Based: Search-Based Software Engineering is the name given to a body of work in which search-based optimization is applied to software engineering [34]. McMinn
Modified on	11/02/2022 12:27

Name	A Survey of Software Quality for Machine Learning Applications
Number of Coding References	8
Number of Codes Coding	1
Coverage	6.24%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	7) Model Checking: Model checking is a traditional tech- nique of software engineering. However, describing applications models with ML allows the application model checking to verify the applications. Alechina et al. proposed a method for verifying existence of resource-bounded coalition uniform strategies to be able to automatically verify properties of such systems using model-checking [47]. Tappler et al. presented learning-based approach to detecting failures by model-based testing [40]
Modified on	11/02/2022 12:28

Name	A Survey of Software Quality for Machine Learning Applications
Number of Coding References	8
Number of Codes Coding	1
Coverage	6.24%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	From this survey, we raised problems with ML applications and discovered software engineering approaches and software testing research areas to solve these problems. We classified survey targets into Academic Conferences, Magazines, and Communities. We targeted 16 academic conferences on artificial intelligence and software engineering, including 78 papers. We targeted 5 Magazines, including 22 papers. The results indicated key areas, such as deep learning, fault localization, and prediction, to be researched with software engineering and testing.
Modified on	11/02/2022 12:23

Name	A Survey of Software Quality for Machine Learning Applications
Number of Coding References	8
Number of Codes Coding	1
Coverage	6.24%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	We summarize the software-quality methods and techniques for ML applications from the top 7 ranked tags in the survey results. 1) Deep Learning: Pei et al. discussed how to find a ”corner case” (case rarely occurs, a troublesome case) in a deep learning system and a test method using it [80]. Hsu mentioned the method in the article ”A NewWay to Find Bugs in Self-Driving AI Could Save Lives” [96]. Bau et al. proposed a technique for interpreting what internal firing of a deep learning model corresponds to using semantic segmentation [28]. Schulam et al. proposed a method of generating counterfactual predictions [73]. The method targeted a circumstances that if prediction can be executed by using experience-based data the predictions have good results, but experience-based data cannot be always used due to ethical problems. Zhang et al., Sangkloy et al. proposed verification method of visual pattern in deep learning [14], [19]. Kokkinos et al. discussed an innovative method of deep learning. They proposed a method that train in an end-to-end manner a convolutional neural network (CNN) that jointly handles low-, mid-, and high-level vision tasks in a unified architecture [22].
Modified on	11/02/2022 12:27

Name	A Survey on Distributed Machine Learning
Number of Coding References	16
Number of Codes Coding	2
Coverage	10.93%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	Communication As previously discussed, the need for more sophisticated machine learning-based setups quickly outgrows the capabilities of a single machine. There are several ways to partition the data and/or the program and to distribute these evenly across all machines. The choice ofdistribution, however, has direct implications on the amount of communication required to train the model. 3.5.1 Computation Time vs. Communication vs. Accuracy. When Distributed Machine Learn- ing is used, one aims for the best accuracy at the lowest computation and communication cost. However, for complex ML problems, the accuracy usually increases with processing more training data, and sometimes by increasing the ML model size, hence increasing the computation cost. Parallelizing the learning can reduce computation time, as long as the communication costs are not becoming dominant. This can become a problem if the model being trained is not sufficiently large in comparison to the data. If the data are already distributed (e.g., cloud-native data), then there is no alternative to either moving the data or the computation. Splitting up the dataset across different machines and training a separate model on a separate part of the dataset avoids communication, but this reduces the accuracy of the individual models trained on each machine. By ensembling all these models, the overall accuracy can be improved, However, the computation time is typically not much lower, since the individual models still have to take the same number ofmodel update steps to converge. By already synchronizing the different models during training (e.g., by combining the calculated gradients on all machines in case of gradient descent), the computation time can be reduced by converging faster to a local optimum. This, however, leads to an increase of communication cost as the model size increases. Therefore, practical deployments require seeking the amount of communication needed to achieve the desired accuracy within an acceptable computation time. 3.5.2 Bridging Computation and Communication. To schedule and balance the workload, there are three concerns that have to be taken into account [161]: • Identifying which tasks can be executed in parallel. • Deciding the task execution order. • Ensuring a balanced load distribution across the available machines. After deciding on these three issues, the information between nodes should be communicated as efficiently as possible. There are several techniques that enable the interleaving of parallel computation and inter-worker communication. These techniques trade off fast/correct model convergence (at the top ofthe list found below) with faster/fresher updates (at the bottom ofthe list found below). • Bulk Synchronous Parallel (BSP) is the simplest model in which programs ensure consistency by synchronizing between each computation and communication phase [161]. An example of a program following the BSP bridging model is MapReduce. An advantage is that serializable BSP ML programs are guaranteed to output a correct solution. A disadvantage is that finishedworkers must wait at every synchronization barrier until all other workers are finished, which results in overhead in the event of some workers progressing slower than others [34]. • Stale Synchronous Parallel (SSP) relaxes the synchronization overhead by allowing the faster workers to move ahead for a certain number of iterations. If this number is exceeded, then all workers are paused. Workers operate on cached versions of the data and only commit changes at the end of a task cycle, which can cause other workers to operate on stale ACM Computing Surveys, Vol. 53, No. 2, Article 30. Publication date: March 2020. A Survey on Distributed Machine Learning 30:15 data. The main advantage of SSP is that it still enjoys strong model convergence guarantees. A disadvantage, however, is that when the staleness becomes too high (e.g., when a significant number ofmachines slows down), the convergence rates quickly deteriorate. The algorithm can be compared to Conits [166], used in distributed systems, because it specifies the data on which the workers are working and consistency is to be measured. • Approximate Synchronous Parallel (ASP) limits how inaccurate a parameter can be. This contrasts with SSP, which limits how stale a parameter can be. An advantage is that, whenever an aggregated update is insignificant, the server can delay synchronization indefinitely. A disadvantage is that it can be hard to

	choose the parameter that defines which updates are significant and which are not [73]. • Barrierless Asynchronous Parallel [65]/Total Asynchronous Parallel [73] (BAP/TAP) lets worker machines communicate in parallel without waiting for each other. Theadvantageisthatitusually obtains thehighest possiblespeedup.Adisadvantage is that the model can converge slowly or even develop incorrectly, because, unlike BSP and SSP, the error grows with the delay [65].
Modified on	28/02/2023 12:19
Name	A Survey on Distributed Machine Learning
Number of Coding References	16
Number of Codes Coding	2
Coverage	10.93%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	Hinton et al. [70] address the challenge that accessing the weights of neurons from DRAM is a costly operation and can dominate the energy profile ofprocessing. Leveraging a deep compression technique, they are able to put the weights into SRAM and accelerate the resulting sparse matrixvector multiplications through efficient weight sharing. The result is a 2.9× higher throughput and a19× improved energy efficiency compared to DianNao. Even general-purpose CPUs have increased the availability and width of vector instructions in recent product generations to accelerate the processing ofcomputationally intensive problems like machine learning algorithms. These instructions are vector instructions, part of the AVX-512 family [126] with enhanced word-variable precision and support for single precision floating-point operations. In addition to the mainstream players, there are more specialized designs available such as the Epiphany [110]. This special-purpose CPU is designed with a MIMD architecture that uses an array of processors, each of which accessing the same memory, to speed up execution of floating-point operations. This is faster than giving every processor its ownmemory, because communicating between processors is expensive. The newest chip ofthe majormanufacturer Adapteva is the Epiphany V, which contains 1,024 cores on a single chip [109]. Although Adapteva has not published power consumption specifications of the Epiphany V yet, it has released numbers suggesting a power usage of only 2 Watts [4].
Modified on	28/02/2023 12:15

Name	A Survey on Distributed Machine Learning
Number of Coding References	16
Number of Codes Coding	2
Coverage	10.93%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	In general, the problem ofmachine learning can be separated into the training and the prediction phase (Figure 1). The Training phase involves training a machine learning model by feeding it a large body of training data and updating it using an ML algorithm. An overview of applicable and commonly used algorithms is given in Section 3.1. Aside from choosing a suitable algorithm for a given problem, we also need to find an optimal set of hyperparameters for the chosen algorithm, which is described in Section 3.2. The final outcome ofthe training phase is a TrainedModel, which can then be deployed. The Prediction phase is used for deploying the trained model in practice. The trained model accepts new data as input and produces a prediction as output. While the training phase of the model is typically computationally intensive and requires the availability of large datasets, the inference can be performed with less computing power. The training phase and prediction phase are not mutually exclusive. Incremental learning com- bines the training phase and inference phase and continuously trains the model by using new data from the prediction phase.
Modified on	28/02/2023 12:15
Name	A Survey on Distributed Machine Learning
Number of Coding References	16
Number of Codes Coding	2
Coverage	10.93%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	In some cases, the long runtime of training the models steers solution designers towards using distributed systems for an increase of parallelization and total amount of I/O bandwidth, as the training data required for sophisticated applications can easily be in the order of terabytes [29]. In other cases, a centralized solution is not even an option when data are inherently distributed or too big to store on single machines. Examples include transaction processing in larger enterprises on data that are stored in different locations [19] or astronomical data that are too large to move and centralize [124]. To make these types ofdatasets accessible as training data for machine learning problems, algo- rithms have to be chosen and implemented that enable parallel computation, data distribution, and resilience to failures. A rich and diverse ecosystem of research has been conducted in this field, which we categorize and discuss in this article. In contrast to prior surveys on distributed machine learning [119, 123] or related fields [87, 121, 122, 143, 152, 170], we apply a wholistic view to the problem and discuss the practical aspects of state-of-the-art machine learning from a distributed systems angle.
Modified on	28/02/2023 12:14

Name	A Survey on Distributed Machine Learning
Number of Coding References	16
Number of Codes Coding	2
Coverage	10.93%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	Machine Learning in the Cloud Several cloud operators have added machine learning as a service to their cloud offerings. Most providers offer multiple options of executing machine learning tasks in their clouds, ranging from IaaS-level services (VM instances with pre-packaged ML software) to SaaS-level solutions (Machine Learning as a Service). Much of the technology offered are standard distributed machine learning systems and libraries. Among other things, Google's Cloud Machine Learning Engine offers support for TensorFlow and even provides TPU instances [60]. Microsoft Azure Machine Learning allows model deployment through Azure Kubernetes, through a batch service, or by using CNTK VMs [101]. As a competitor to Google's TPUs, Azure supports accelerating ML applications through FPGAs [114]. Amazon AWS has introduced SageMaker, a hosted service for building and training machine learning models in the cloud. The service includes support for TensorFlow, MXNet, and Spark [7]. IBM has bundled their cloud machine learning offerings under the Watson brand [74]. Services include Jupyter notebooks, Tensorflow, and Keras. The cloud-based delivery model is becoming more important, as it reduces the burden of entry into designing smart applications that facilitate machine learning techniques. However, the cloud is not only a consumer of distributed machine learning technology but is also fueling the development of new systems and approaches back to the ecosystem to handle the large scale of the deployments.
Modified on	28/02/2023 12:20

Name	A Survey on Distributed Machine Learning
Number of Coding References	16
Number of Codes Coding	2
Coverage	10.93%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	MACHINE LEARNING—A HIGH-PERFORMANCE COMPUTING CHALLENGE? Recent years have seen a proliferation of machine learning technology in increasingly complex applications. While various competing approaches and algorithms have emerged, the data representations used are strikingly similar in structure. The majority of computation in machine learning workloads amounts to basic transformations on vectors, matrices, or tensors—well-known problems from linear algebra. The need to optimize such operations has been a highly active area of research in the high-performance computing community for decades. As a result, some techniques and libraries from the HPC community (e.g., BLAS [89]orMPI [62]) have been successfully adopted and integrated into systems by the machine learning community. At the same time, the HPC community has identified machine learning to be an emerging high-value workload and has started to apply HPC methodology to them. Coates et al. [38] were able to train a 1B parameter network on their Commodity Off-The-Shelf High Performance Computing (COTS HPC) system in just three days. You et al. [165] optimized the training of a neural network on Intel's Knights Landing, a chip designed for HPC applications. Kurth et al. [84] demonstrated how deep learning problems like extracting weather patterns can be optimized and scaled efficiently on large parallel HPC systems. Yan et al. [162] have addressed the challenge of scheduling deep neural network applications on cloud computing infrastructure by modeling the workload demand with techniques like lightweight profiling, which are borrowed from HPC. Li et al. [91] investigated the resilience characteristics of deep neural networks with regard to hardware errors when running on accelerators, which are frequently deployed in major HPC systems. Like for other large-scale computational challenges, there are two fundamentally different and complementary ways of accelerating workloads: adding more resources to a single machine (vertical scaling or scaling up) and adding more nodes to the system (horizontal scaling or scaling out).
Modified on	28/02/2023 12:14

Name	A Survey on Distributed Machine Learning
Number of Coding References	16
Number of Codes Coding	2
Coverage	10.93%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	Scaling Out While there are many different strategies to increase the processing power of a single machine for large-scale machine learning, there are reasons to prefer a scale-out design or combine the two approaches, as often seen in HPC. The first reason is the generally lower equipment cost, both in terms of initial investment and maintenance. The second reason is the resilience against failures because, when a single processor fails within an HPC application, the system can still continue operating by initiating a partial recovery (e.g., based on communication-driven checkpointing [46] or partial re-computation [168]). The third reason is the increase in aggregate I/O bandwidth compared to a single machine [49]. Training ML models is a highly data-intensive task, and the ingestion of data can become a serious performance bottleneck [67]. Since every node has a dedicated I/O subsystem, scaling out is an effective technique for reducing the impact of I/O on the workload performance by effectively parallelizing the reads and writes over multiple machines. A major challenge of scaling out is that not all ML algorithms lend themselves to a distributed computing model, which can thus only be used for algorithms that can achieve a high degree of parallelism. 2.3 Discussion The lines between traditional supercomputers, grids, and the cloud are increasingly getting blurred when it comes to the best execution environment for demanding workloads like machine learning. For instance, GPUs and accelerators are now more common in major cloud datacenters [134, 135]. As a result, parallelization of the machine learning workload has become paramount to achieving acceptable performance at large scale. When transitioning from a centralized solution to a distributed system, however, the typical challenges of distributed computing in the form of performance, scalability, failure resilience, or security apply [40]. The following section presents a systematic discussion of the different aspects of distributed machine learning and develops a reference architecture by which all existing systems can be categorized.
Modified on	28/02/2023 12:15

Name	A Survey on Distributed Machine Learning
Number of Coding References	16
Number of Codes Coding	2
Coverage	10.93%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	Scaling Up Among the scale-up solutions, adding programmable GPUs is the most common method and various systematic efforts have shown the benefits ofdoing so [18, 78, 125]. GPUs feature a high number of hardware threads. For example, the Nvidia Titan V and Nvidia Tesla V100 have a total of 5,120 cores, which makes them approximately 47× faster for deep learning than a regular server CPU (namely an Intel Xeon E5-2690v4) [107]. Originally the applications of GPUs for machine learning were limited because GPUs used a pure SIMD (Single Instruction, Multiple Data) [51] model that did not allow the cores to execute a different branch of the code; all threads had to perform the exact same program. Over the years GPUs have shifted to more flexible architectures where the overhead ofbranch divergence is reduced, but diverging branches is still inefficient [66]. The proliferation of GPGPUs (General-Purpose GPUs, i.e., GPUs that can execute arbitrary code) has lead the vendors to design custom products that can be added to conventional machines as accelerators and no longer fulfill any role in the graphics subsystem of the machine. For example, the Nvidia Tesla GPU series is meant for highly parallel computing and designed for deployment in supercomputers and clusters. When a sufficient degree of parallelism is offered by the workload, GPUs can significantly accelerate machine learning algorithms. For example, Meuth [100] reported a speed-up up to 200× over conventional CPUs for an image recognition algorithm using a Pretrained Multilayer Perceptron (MLP). An alternative to generic GPUs for acceleration is the use of Application Specific Integrated Circuits (ASICs), which implement specialized functions through a highly optimized design. In recent times, the demand for such chips has risen significantly [99]. When applied to, e.g., Bitcoin mining, ASICs have a significant competitive advantage over GPUs and CPUs due to their high performance and power efficiency [144]. Since matrix multiplications play a prominent role in many machine learning algorithms, these workloads are highly amenable to acceleration through ASICs. Google applied this concept in their Tensor Processing Unit (TPU) [128], which, as the name suggests, is an ASIC that specializes in calculations on tensors (n-dimensional arrays), and is designed to accelerate their Tensorflow [1, 2] framework, a popular building block for machine learning models. The most important component of the TPU is its Matrix Multiply unit based on a systolic array. TPUs use a MIMD (Multiple Instructions, Multiple Data) [51] architecture that, unlike GPUs, allows them to execute diverging branches efficiently. TPUs are attached to the server system through the PCI Express bus. This provides them with a direct connection with the CPU, which allows for a high aggregated bandwidth of 63 GB/s (PCI-e5x16). Multiple TPUs can be used in a data center, and the individual units can collaborate in a distributed setting. The benefit of the TPU over regular CPU/GPU setups is not only its increased processing power but also its power efficiency, which is important in large-scale applications due to the cost of energy and the limited availability in large-scale data centers. When running benchmarks, Jouppi et al. [80]found that the performance per watt of a TPU can approach 200× that of a traditional system. Further benchmarking by Sato et al. [128] indicated that the total processing power of a TPU or GPU can be up to 70× higher than a CPU for a typical neural network, with performance improvements varying from 3.5×–71×, depending on the task at hand. Chen et al. [32] developed DianNao, a hardware accelerator for large-scale neural networks with a small area footprint. Their design introduces a Neuro-Functional Unit (NFU) in a pipeline that multiplies all inputs, adds the results, and, in a staggered manner after all additions have been performed, optionally applies an activation function like a sigmoid function. The experimental evaluation using the different layers of several large neural network structures [48, 70, 90, 132, 133] shows a performance speedup of three orders of magnitude and an energy reduction ofmore than 20× compared to using a general-purpose 128-bit 2 GHz SIMD CPU.
Modified on	28/02/2023 12:15

Name	A Survey on Distributed Machine Learning
Number of Coding References	16
Number of Codes Coding	2
Coverage	10.93%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	
Modified on	28/02/2023 12:21

Name	A Survey on Distributed Machine Learning
Number of Coding References	16
Number of Codes Coding	2
Coverage	10.93%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03

Coded Text	Another consideration for the design ofa distributedmachine learning deployment is the structure in which the computers within the cluster are organized. A deciding factor for the topology is the degree of distribution that the system is designed to implement. Figure 3 shows four possible topologies, in accordance with the general taxonomy of distributed communication networks by Baran [15]. Centralized systems (Figure 3(a)) employ a strictly hierarchical approach to aggregation, which happens in a single central location. Decentralized systems allow for intermediate aggregation, either with a replicated model that is consistently updated when the aggregate is broadcast to all nodes such as in tree topologies (Figure 3(b)) or with a partitioned model that is sharded over multiple parameter servers (Figure 3(c)). Fully distributedsystems (Figure 3(d)) consist ofa network ofindependent nodes that ensemble the solution together and where no specific roles are assigned to certain nodes. There are several distinct topologies that have become popular choices for distributed machine learning clusters: • Trees. Tree-like topologies have the advantage that they are easy to scale and manage, as each node only has to communicate with its parent and child nodes. For example, in the AllReduce [5] paradigm, nodes in a tree accumulate their local gradients with those from their children and pass this sum to their parent node to calculate a global gradient. • Rings. In situations where the communication system does not provide efficient support for broadcast or where communication overhead needs to be kept to a minimum, ring topologies for AllReduce patterns simplify the structure by only requiring neighbor nodes to synchronize through messages. This is, e.g., commonly used between multiple GPUs on the same machine [76]. • Parameter Server. The Parameter Server paradigm (PS) [155] uses a decentralized set of workers with a centralized set of masters that maintain the shared state. All model parameters are stored in a shard on each parameter server, from which all clients read and write as a key-value store. An advantage is that all model parameters (within a shard) are in a global shared memory, which makes it easy to inspect the model. A disadvantage of the
Modified on	28/02/2023 12:17

Name	A Survey on Distributed Machine Learning
Number of Coding References	16
Number of Codes Coding	2
Coverage	10.93%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Communication Strategies. Communication is an important contributor to defining the performance and scalability of distributed processing [27]. Several communication management strategies [161] are used to spread and reduce the amount of data exchanged between machines: • To prevent bursts ofcommunication over the network (e.g., after a mapper is finished), continuous communication is used, such as in the state-of-the-art implementation Bösen [155]. • Neural networks are composed out of layers, the training of which (using the backpropagation gradient descent algorithm) is highly sequential. Because the top layers of neural networks contain the most parameters while accounting for only a small part of the total computation, Wait-free Backpropagation (WFBP) [171] was proposed. WFBP exploits the neural network structure by sending out the parameter updates of the top layers while still computing the updates for the lower layers, hence hiding most of the communication latency. • Because WFBP does not reduce the communication overhead, hybrid communication (HybComm) [171] was proposed. Effectively, it combines Parameter Servers (PS) [155] with Sufficient Factor Broadcasting (SFB) [159], choosing the best communication method depending on the sparsity of the parameter tensor. See below for more information about PS (under Centralized Storage) and SFB (under Decentralized Storage).
Modified on	28/02/2023 12:19

Name	A Survey on Distributed Machine Learning
Number of Coding References	16
Number of Codes Coding	2
Coverage	10.93%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Parallelism in distributed machine learning. Data parallelism trains multiple instances of the same model on different subsets of the training dataset, while model parallelism distributes parallel paths of a single model to multiple nodes. One option is to train different instances of the same or similar model, and aggregate the outputs of all trained models using methodologies like ensembling (Section 3.3). The final architectural decision is the topology of the distributed machine learning system. The different nodes that form the distributed system need to be connected through a specific architectural pattern to fulfill a common task. However, the choice of pattern has implications on the role that a node can play, the degree ofcommunication between nodes, and the failure resilience ofthe whole deployment. A discussion of commonly used topologies is presented in Section 3.4. In practice, the three layers of architecture (machine learning, parallelism, topology) are not independent. The combining factor is their impact on the amount of communication required to train the model, which is discussed in Section 3.5.
Modified on	28/02/2023 12:15

Name	A Survey on Distributed Machine Learning
Number of Coding References	16
Number of Codes Coding	2
Coverage	10.93%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Peer-to-Peer. In contrast to centralized state, in the fully distributed model, every node has its own copy of the parameters and the workers communicate directly with each other. This has the advantage of typically higher scalability than a centralized model and the elimination of single points of failure in the system [52]. An example implementation of this model is a peer-to-peer network, in which nodes broadcast updates to all other nodes to form a data-parallel processing framework. Since full broadcast is typically prohibitive due to the volume of communication, Sufficient Factor Broadcasting (SFB) [94] has been proposed to reduce the communication overhead. The parameter matrix in SFB is decomposed into so-called sufficient factors, i.e., two vectors that are sufficient to reconstruct the update matrix. SFB only broadcasts these sufficient factors and lets the workers reconstruct the updates. Other models limit the degree of communication to less-frequent synchronization points while allowing the individual models to temporarily diverge. Gossip Learning [138] is built around the idea that models are mobile and perform independent random walks through the peer-to-peer network. Since this forms a data- and model-parallel processing framework, the models evolve differently and need to be combined through ensembling. In Gossip Learning, this happens continuously on the nodes by combining the current model with a limited cache of previous visitors.
Modified on	28/02/2023 12:18

Name	A Survey on Distributed Machine Learning
Number of Coding References	16
Number of Codes Coding	2
Coverage	10.93%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	The demand for artificial intelligence has grown significantly over the past decade, and this growth has been fueled by advances in machine learning techniques and the ability to leverage hardware acceleration. However, to increase the quality of predictions and render machine learning solutions feasible for more complex applications, a substantial amount of training data is required. Although small machine learning models can be trained with modest amounts of data, the input for training larger models such as neural networks grows exponentially with the number ofparameters. Since the demand for processing training data has outpaced the increase in computation power ofcomputing machinery, there is a need for distributing the machine learning workload across multiple machines, and turning the centralized into a distributed system. These distributed systems present new challenges: first and foremost, the efficient parallelization ofthe training process and the creation ofa coherent model. This article provides an extensive overview ofthe current state-of-the-art in the field by outlining the challenges and opportunities of distributed machine learning over conventional (centralized) machine learning, discussing the techniques used for distributed machine learning, and providing an overview of the systems that are available.
Modified on	28/02/2023 12:13

Name	A Survey on Distributed Machine Learning
Number of Coding References	16
Number of Codes Coding	2
Coverage	10.93%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	topology is that the parameter servers can form a bottleneck, because they are handling all communication. To partially alleviate this issue, the techniques for bridging computation and communication mentioned in Section 3.5.2 are used.
Modified on	28/02/2023 12:18

Name	A Survey on Distributed Machine Learning
Number of Coding References	16
Number of Codes Coding	2
Coverage	10.93%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	When it comes to distribution, there are two fundamentally different ways of partitioning the problem across all machines: parallelizing the data or the model [119] (Figure 2). These two methods can also be applied simultaneously [161]. In the Data-Parallel approach, the data are partitioned as many times as there are worker nodes in the system and all worker nodes subsequently apply the same algorithm to different datasets. The same model is available to all worker nodes (either through centralization or through replication) so a single coherent output emerges naturally. The technique can be used with every ML algorithm with an independent and identical distribution (i.i.d.) assumption over the data samples (i.e., most ML algorithms [161]). In the Model-Parallel approach, exact copies of the entire datasets are processed by the worker nodes that operate on different parts ofthe model. The model is therefore the aggregate ofall model parts. The model-parallel approach cannot automatically be applied to every machine learning algorithm, because the model parameters generally cannot be split up.
Modified on	28/02/2023 12:15

Name	A Survey on Distributed Machine Learning Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	A Taxonomy of Software Engineering Challenges for Machine Learning Systems~ An Empirical Investigation
Number of Coding References	15
Number of Codes Coding	2
Coverage	15.99%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	
Modified on	11/02/2022 12:15

Name	A Taxonomy of Software Engineering Challenges for Machine Learning Systems~ An Empirical Investigation
Number of Coding References	15
Number of Codes Coding	2
Coverage	15.99%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	
Modified on	11/02/2022 12:17

Name	A Taxonomy of Software Engineering Challenges for Machine Learning Systems~ An Empirical Investigation
Number of Coding References	15
Number of Codes Coding	2
Coverage	15.99%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	Among the challenges identified from practitioners of Case E are difficulties with complex and poor logging mechanisms as well as in designing experiments, including interleaving experiments often done in ML components and interpreting experiment results. If product teams want to have good informative experiments they need to log the correct things. Logging in the past was done to understand if a product has crashed or not, or why it has crashed. This is not sufficient if you want to compute good business metrics in the end of the day
Modified on	11/02/2022 12:15

Name	A Taxonomy of Software Engineering Challenges for Machine Learning Systems~ An Empirical Investigation
Number of Coding References	15
Number of Codes Coding	2
Coverage	15.99%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	Data analysis and validation is an initial and critical activity for designers of ML components. Absence of critical analysis on training data results to training and serving skew, which describes differences in performance of ML model at training and deploy. The discrepancies are largely caused by differences in the handling of data distributions and pipelines during training and operations. Data sources or different fields in data e.g., in logs, may come from different components owned by other teams. Major changes to the values invalidate models trained on older data. Techniques and tools for monitoring and tracking data are crucial for developing ML systems, as supported in literature.
Modified on	11/02/2022 12:16

Name	A Taxonomy of Software Engineering Challenges for Machine Learning Systems~ An Empirical Investigation
Number of Coding References	15
Number of Codes Coding	2
Coverage	15.99%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	Databricks tool is used to build and deploy the models, which are typically saved as a single library and are version controlled. Prior to their recent use of the Databricks tool, the team faced challenges related to the lack of standardized approaches for reproducing model selection experiments quickly and scaling models in production. We also needed to worry about scaling because the volume of messages differs a lot with time. Generally, on Monday sales people send hundreds of thousands of emails to new prospects. We had to either do it manually by deploying more copies of the model, and then bringing them down to not use-up resources, or leave the model and then there will be a queue. We did not do this manual approach and the queue would get to almost 24 hours long. So those emails will only get processed on Tuesday because of the volume.
Modified on	11/02/2022 12:15

Name	A Taxonomy of Software Engineering Challenges for Machine Learning Systems~ An Empirical Investigation
Number of Coding References	15
Number of Codes Coding	2
Coverage	15.99%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	Depending on the use case, and whether the team is allowed to move data, datasets of varying sizes are used to train models. In extreme scenario with a datasets of 3TB per day and where data is not allowed to be moved outside a country, federated learning is used. In federated learning an initial model is built locally and then it gets trained and improved at the edge. The training dataset is curated and features engineered by data scientists prior to training. ML models are trained while also residing in the CI-CD pipeline since the company supports many customers across different locations. When training the models care is taken not to mix data of different clients. The ML models are packaged as Docker images that are deployed on Kubernetes in the cloud and monitored for model usage and accuracy, in addition to CPU usage and memory, using a tool called Prometheus. The main engineering challenges for Case D are related to data collection and model localization particularly in areas where data movement is constrained, as elaborated in quote below by the Tech Lead. I think really the challenge is actually getting data and that is why we are investing so much in federated learning because in some cases the data cannot leave the country. And also in some cases the links that you have are not strong enough to carry the data that you want because they are used by other things. So that is really the key challenge here and that is why we are looking into the techniques such as federated learning and reinforcement learning so that we can improve on it.
Modified on	11/02/2022 12:14
Name	A Taxonomy of Software Engineering Challenges for Machine Learning Systems~ An Empirical Investigation
Number of Coding References	15
Number of Codes Coding	2
Coverage	15.99%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	Existing data collection mechanisms and storage of software-intensive sys- tem are originally not set-up for ML systems. For example, poor logging and limited data cleaning mechanisms exist prior to the ML initiative. As a result, potentially large efforts are spent on data exploration, in addition to determining and formulating the problem of ML in the respective application domain. Difficulty in formulating the problem for ML is accounted for, among other, by the need to determine beforehand a benchmark or baseline against which ML model will be evaluated for accuracy and performance optimizations. While a variety of big data tools are used in data aggregation and structuring, different design decisions and trade-offs in model creation rely on inputs from domain experts, such as useful features. At this stage, models are not deployed but do provide valuable feedback to the experts about the direct impact of suggested features.
Modified on	11/02/2022 12:16

Name	A Taxonomy of Software Engineering Challenges for Machine Learning Systems~ An Empirical Investigation
Number of Coding References	15
Number of Codes Coding	2
Coverage	15.99%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	Experimentation platform consists of mainly four components namely experi- mentation portal, experiment execution service, log processing service, and analysis service. A good logging system that captures correct events, at correct time and identify targets is important for running experiments on the platform. This is because every product user is every single point in time in several experiments and logs need to be annotated with information of which experiments users in addition to using the data to (re)train models. Users can run their experiments on platform as per their requirement either with the help of some predefined templates or without templates by which they can eventually find a better performing trained model. This is important because the models are compared using measures that the business care for because users are using system functionality and not the models.
Modified on	11/02/2022 12:15

Name	A Taxonomy of Software Engineering Challenges for Machine Learning Systems~ An Empirical Investigation
Number of Coding References	15
Number of Codes Coding	2
Coverage	15.99%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	For automatic sentiment tagging, the selected final DL model is deployed to a Kubernetes cluster in cloud infrastructure allowing among other, scaling. It exposes a REST API that can be called via JavaScript fetch from the online music catalogue application. At the time of interview, trained ML models of the output quality predictions of pulp processing had not been deployed in production but yielded feedback in form of report given to clients about features indicative of quality. The later was among factors considered in decision to buy a new machine. In addition to challenges of developing AI platform, such as managing design trade-offs in customization of platform functionalities, other challenges concerned handling of data drifts in uploaded data, invalidation of models e.g., due to changes in data sources, and the need to monitor models in production for staleness. You’re trying to simultaneously build reproducibility, collaboration and ease of use at the same time you’re trying to give people as much customization as possible. It’s the difference between giving somebody a notebook where they can do anything they want and giving a higher level tool that has a lot of built-in functionality. It’s there that I see most challenges
Modified on	11/02/2022 12:10

Name	A Taxonomy of Software Engineering Challenges for Machine Learning Systems~ An Empirical Investigation
Number of Coding References	15
Number of Codes Coding	2
Coverage	15.99%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	For critical deployment of ML components, challenges in the implementation of the end-to-end ML pipeline comes with the need and difficulty in implementing an effective experimentation infrastructure. The experimentation infrastructure is used to evaluate performance improvements and effects of ML models with the use of metrics that are business-centric rather than algorithmic-centric. The ability to design and conduct several experiments on continuous basis is nontrivial. While experiments that are conducted online are exploring and exploiting the models, end-users are not to be affected and need to adhere to the stringent requirements of latency and throughput.
Modified on	11/02/2022 12:17

Name	A Taxonomy of Software Engineering Challenges for Machine Learning Systems~ An Empirical Investigation
Number of Coding References	15
Number of Codes Coding	2
Coverage	15.99%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	For the final model in the cascading deployment strategy, the challenge comes from the difficulty in tracking changes in models giving the input features and in performing a sliced analysis to the evaluation results. It is apparent that as the system scales to handle more models, it becomes difficult to identify the cause of poor performance for final system, for example due to undeclared consumers [20]. When final model performance results are not sliced, such as merely focus on accuracy on validation training set, according to [15] important effects are masked and can result in quality improving in one part but degrading in another.
Modified on	11/02/2022 12:17

Name	A Taxonomy of Software Engineering Challenges for Machine Learning Systems~ An Empirical Investigation
Number of Coding References	15
Number of Codes Coding	2
Coverage	15.99%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	However, efficient software engineering principles and processes need to be considered and extended when developing AI- enabled systems. The objective of this study is to identify and classify software engineering challenges that are faced by different companies when developing software-intensive systems that incorporate machine learning components. Using case study approach, we explored the development of machine learning systems from six different companies across various domains and identified main software engineering challenges.
Modified on	11/02/2022 12:05

Name	A Taxonomy of Software Engineering Challenges for Machine Learning Systems~ An Empirical Investigation
Number of Coding References	15
Number of Codes Coding	2
Coverage	15.99%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	Main SE challenges identified from Case C is the need for processes and tools for forming accurate and consistent annotations in large dataset, especially when the system has no self-labelling instrumentation. Furthermore, there are difficulties in negotiating interpretations and dealing with poor inter-rater agreement across a large group of annotators. Customers using the annotated dataset, often do not have other mechanisms to know if the annotations were done correctly. “So the challenging part of creating large amounts of examples is that it’s usually ambiguous. You have a distributed group of people and you need very low error tolerance, because if you’re going to have production grade machine learning systems, their performance will be governed by the quality of the data”
Modified on	11/02/2022 12:11

Name	A Taxonomy of Software Engineering Challenges for Machine Learning Systems~ An Empirical Investigation
Number of Coding References	15
Number of Codes Coding	2
Coverage	15.99%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	While Case A poses issues and extreme requirements for storage and pre- processing of large data sets for creating DL models in automotive domain, one main engineering challenge perceived by the DL organization manager was difficulty in building DL infrastructure, as expressed below. This is in addition to problems related evolving requirement definition for AD vehicles, which affect model training and evaluations. At the time of the interview, there was limited support for quickly recreating the different training results. There are no tool-chains you can download in an infrastructure with deep learning like this. And we realized after the mistakes and discussions with our new IT that they didn't really have the expertise to be able to deliver this to us. So we had to create new teams, which took the responsibility of creating both the infrastructure, but also the software tool-chain to be able to train deep learning networks within a reasonable amount of time.
Modified on	11/02/2022 12:09

Name	A Taxonomy of Software Engineering Challenges for Machine Learning Systems~ An Empirical Investigation
Number of Coding References	15
Number of Codes Coding	2
Coverage	15.99%
Folder Location	Codes\\Snowballing
Parent Name	Codes\\Snowballing\\Snowballing
Created on	21/06/2022 14:00
Coded Text	The research area of this study is applied ML, wherein the focus is to create verifiable knowledge pertaining to the design of software systems that incorporate ML techniques [14]. In our study, the considered software systems not only incorporate ML techniques to real world problems but are in operational use in commercial settings. This is in contrast to application of ML techniques to activities of software development process in field of SE [23], such as fault prediction and localization in software testing, which also gives numerous benefits in practice [16]. There exists empirical studies [6] and experience reports [7,15,19,21] pub- lished across different disciplines that present an end-to-end development process and challenges of operational AI-enabled applications. In a field study of how intelligent systems are developed, Hill et al. [6] describe a high-level process SE Challenges for AI-Enabled Systems 229 that includes the following activities that are not necessarily sequential: defining problem, collecting data, establishing ground truth, selecting algorithm, selecting features and creating and evaluating ML model. Most of the challenges identified by the authors [6] at each activity of the ML process as well as cross-cutting issues are reported in other empirical reports [1,19]. For instance, the use of informal methods to manage dataset and common artifacts (trained models, feature sets, training jobs) during ML model selection experiments is a challenge that is commonly observed and presents difficulties to quickly reproduce and compare different experiments [18]. In addition to using agile approach for quick iterations [19], among the solutions proposed include using versioning in ML pipelines [22] and automating tracking of metadata and provenance information of the common artifacts [18]. However, some challenges are yet to be addressed, such as tracking provenance of complex final model that combines variety ofmodels trained on different dataset [18] and data generated and processed through highly-heterogeneous infrastructure [13]. Concerning ML infrastructure several challenges are encountered, such as the ability to train models with large data volumes [5]. Using technical debt metaphor of SE, Sculley et al. [20] bring to awareness the different trade-offs involved, and require careful consideration, when maintaining ML systems overtime in real-world industrial settings. According to the authors [20], technical debt in real-world ML systems is attributed to maintenance problems of application code and issues specific to ML, such as data dependencies. ML systems have various sources of variability that need to be stabilized otherwise they can cause significant differences between ML models [8]. On the other hand, difficulties in debugging DL systems is currently one of the challenging topic that is gaining much focus in research [4]. Our study seeks to provide a taxonomy that can be used to consolidate the different challenges reported in prior empirical reports
Modified on	22/06/2022 11:30

Name	A view of programming scalable data analysis~ from clouds to exascale
Number of Coding References	26
Number of Codes Coding	4
Coverage	15.77%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	As the data scale increases, we must address new chal- lenges and attack ever-larger problems. New discoveries will be achieved and more accurate investigations can be carried out due to the increasingly widespread availability of large amounts of data. Scientific sectors that fail to make full use of the huge amounts of digital data available today risk losing out on the significant opportunities that big data can offer. To benefit from the big data availability, specialists and researchers need advanced data analysis tools and applications running on scalable architectures allowing for the extraction of useful knowledge from such huge data sources.
Modified on	10/02/2022 12:55

Name	A view of programming scalable data analysis~ from clouds to exascale
Number of Coding References	26
Number of Codes Coding	4
Coverage	15.77%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	As we discussed in the previous sections, data analysis gained a primary role because of the very large availability of datasets and the continuous advancement of methods and algorithms for finding knowledge in them. Data analysis solutions advance by exploiting the power of data mining and machine learning techniques and are changing several scientific and industrial areas. For example, the amount of data that social media daily generate is impressive and continuous. Some hundreds of terabyte of data, including several hundreds of millions of photos, are uploaded daily to Facebook and Twitter. Therefore it is central to design scalable solutions for processing and analysis such massive datasets. As a general forecast, IDC experts estimate data generated to reach about 45 zettabytes worldwide by 2020 [6]. This impressive amount of digital data asks for scalable high performance data analysis solutions. However, today only one-quarter of digital data available would be a
Modified on	10/02/2022 14:09

Name	A view of programming scalable data analysis~ from clouds to exascale
Number of Coding References	26
Number of Codes Coding	4
Coverage	15.77%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	fail and assuring its completion. Reliability is an issue much more important in the Exascale domain where the number of processing elements is massive and fault occurrence increases making detection and recovering vital. ~ Application reproducibility. Reproducibility is another open research issue for designers of complex applications running on parallel systems. Reproducibility in scalable data analysis must face, for example, with data communication, data parallel manipulation and dynamic computing environments. Reproducibility demands that current data analysis frameworks (like those based on MapReduce and on workflows) and the future ones, especially those implemented on Exascale systems, must provide additional information and knowledge on how data are managed, on algorithm characteristics and on configuration of software and execution environments. ~ Data and tool integration and openness. Code coordination and data integration are main issues in large-scale applications that use data and computing resources. Standard formats, data exchange models and common APIs are needed to support interoperability and ease cooperation among design teams using different data formats and tools. ~ Interoperability ofbig data analytics frameworks. The service-oriented paradigm allows running largescale distributed applications on cloud heterogeneous platforms along with software components developed using different programming languages or tools. Cloud service paradigms must be designed to allow worldwide integration of multiple data analytics frameworks.
Modified on	10/02/2022 14:14

Name	A view of programming scalable data analysis~ from clouds to exascale
Number of Coding References	26
Number of Codes Coding	4
Coverage	15.77%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	However, additional research work in this field must be done and the development of new models, solutions and tools is needed [13, 24]. Just to mention a few, active and promising research topics are listed here ordered by importance factors: ~ Programming models for big data analytics. New abstract programming models and constructs hiding the system complexity are needed for big data analytics tools. The MapReduce model and workflow models are often used on HPC and clouds, but more research effort is needed to develop other scalable, adaptive, general-purpose higher-level models and tools. Research in this area is even more important for Exascale systems; in the next section we will discuss some of these topics in Exascale computing. ~ Reliability in scalable data analysis. As the number of processing elements increases, reliability of systems and applications decreases, therefore mechanisms for detecting and handling hardware and software faults are needed. Although in [7] has been proved that no reliable communication protocol can tolerate crashes of processors on which the protocol runs, as stated in the same paper some ways in which systems cope with the impossibility result can be found. Among them, at programming level it is necessary to design constructs for handling communication, data access, and computing failures and for recovering from them. Programming models, languages and APIs must provide general and data-oriented mechanisms for failure detection and isolation, avoiding that an entire application can
Modified on	10/02/2022 14:14

Name	A view of programming scalable data analysis~ from clouds to exascale
Number of Coding References	26
Number of Codes Coding	4
Coverage	15.77%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	importance in programming models for data analysis on massively parallel systems. ~ Locality-based data selection and classification for limiting the latency of basic data analysis operations running in parallel on large scale machines in a way that the subset of data needed together in a given phase are locally available (in a subset of nearby cores).
Modified on	10/02/2022 14:16

Name	A view of programming scalable data analysis~ from clouds to exascale
Number of Coding References	26
Number of Codes Coding	4
Coverage	15.77%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	Requirements of exascale runtime for data analysis One of the most important aspect to ponder in applications that run on Exascale systems and analyze big datasets is the tradeoff between sharing data among processing elements and computing things locally to reduce communication and energy costs, while keeping performance and fault-tolerance levels. A scalable programming model founded on basic operations for data intensive/data-driven applications must include mechanisms and operations for ~ Parallel data access that allows increasing data access bandwidth by partitioning data into multiple chunks, according to different methods, and accessing several data elements in parallel to meet high throughput requirements. ~ Fault resiliency that is a major issue as machines expand in size and complexity. On Exascale systems with huge amount of processes, non-local communication must be prepared for a potential failure of one of the communication sides; runtimes must features failure handing mechanisms for recovering from node and communication faults. ~ Data-driven local communication that is useful to limit the data exchange overhead in massively parallel systems composed of many cores; in this case data availability among neighbor nodes dictates the operations taken by those nodes. ~ Data processing on limited groups of cores allows concentrating data analysis operations involving limited sets of cores and large amount of data on localities of Exascale machines facilitating a type of data affinity co-locating related data and computation. ~ Near-data synchronization to limit the overhead generated by synchronization mechanisms and protocols that involve several far away cores in keeping data up-to-date. ~ In-memory querying and analytics needed to reduce query response times and execution of analytics operations by caching large volumes of data in the computing node RAMs and issuing queries and other operation in parallel on the main memory of computing nodes. ~ Group-level data aggregation in parallel systems is useful for efficient summarization, graph traversal and matrix operations, therefore it is of great
Modified on	10/02/2022 14:16

Name	A view of programming scalable data analysis~ from clouds to exascale
Number of Coding References	26
Number of Codes Coding	4
Coverage	15.77%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	Scalability is a key feature for big data analysis and machine learning frameworks and for applications that need to analyze very large and real-time data available from data repositories, social media, sensor networks, smartphones, and the Web. Scalable big data analysis today can be achieved by parallel implementations that are able to exploit the computing and storage facilities of high performance computing (HPC) systems and clouds, whereas in the near future Exascale systems will be used to implement extreme-scale data analysis. Here is discussed how clouds currently support the development of scalable data mining solutions and are outlined and examined the main challenges to be addressed and solved for implementing innovative data analysis applications on Exascale systems.
Modified on	10/02/2022 12:52

Name	A view of programming scalable data analysis~ from clouds to exascale
Number of Coding References	26
Number of Codes Coding	4
Coverage	15.77%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	22/07/2022 12:21
Coded Text	Data analysis on clouds Clouds implement elastic services, scalable performance and scalable data storage used by a large and everyday increasing number of users and applications [2, 12]. In fact, clouds enlarged the arena of distributed computing systems by providing advanced Internet services that complement and complete functionalities of distributed computing provided by the Web, Grid systems and peer-to-peer networks. In particular, most cloud computing applications use big data repositories stored within the cloud itself, so in those cases large datasets are analyzed with low latency to effectively extract data analysis models. Big data is a new and over-used term that refers to massive, heterogeneous, and often unstructured digital content that is difficult to process using traditional data
Modified on	22/07/2022 12:21

Name	A view of programming scalable data analysis~ from clouds to exascale
Number of Coding References	26
Number of Codes Coding	4
Coverage	15.77%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	22/07/2022 12:21
Coded Text	platforms for addressing both the computational and data storage needs of big data mining and parallel knowledge discovery applications. These computing architectures are needed to run data analysis because complex data mining tasks involve data- and compute-intensive algorithms that require large, reliable and effective storage facilities together with high performance processors to get results in appropriate times. Now that data sources became very big and pervasive, reliable and effective programming tools and applications for data analysis are needed to extract value and find useful insights in them. New ways to correctly and proficiently compose different distributed models and paradigms are required and interaction between hardware resources and programming levels must be addressed. Users, professionals and scientists working in the area of big data need advanced data analysis programming models and tools coupled with scalable architectures to support the extraction of useful information from such massive repositories. The scalability of a parallel computing system is a measure of its capacity to reduce program execution time in proportion to the number of its processing elements (The Appendix introduces and discusses in detail scalability in parallel systems). According to scalability definition, scalable data analysis refers to the ability of a hardware/software parallel system to exploit increasing computing resources effectively in the analysis of (very) large datasets.
Modified on	22/07/2022 12:21

Name	A view of programming scalable data analysis~ from clouds to exascale
Number of Coding References	26
Number of Codes Coding	4
Coverage	15.77%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	22/07/2022 12:21
Coded Text	Scalability is a key feature for big data analysis and machine learning frameworks and for applications that need to analyze very large and real-time data available from data repositories, social media, sensor networks, smartphones, and the Web. Scalable big data analysis today can be achieved by parallel implementations that are able to exploit the computing and storage facilities of high performance computing (HPC) systems and clouds, whereas in the near future Exascale systems will be used to implement extreme-scale data analysis. Here is discussed how clouds currently support the development of scalable data mining solutions and are outlined and examined the main challenges to be addressed and solved for implementing innovative data analysis applications on Exascale systems.
Modified on	22/07/2022 12:21

Name	A view of programming scalable data analysis~ from clouds to exascale
Number of Coding References	26
Number of Codes Coding	4
Coverage	15.77%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	22/07/2022 12:21
Coded Text	Solving problems in science and engineering was the first motivation for inventing computers. After a long time since then, computer science is still the main area in which innovative solutions and technologies are being developed and applied. Also due to the extraordinary advancement of computer technology, nowadays data are generated as never before. In fact, the amount of structured and unstructured digital datasets is going to increase beyond any estimate. Databases, file systems, data streams, social media and data repositories are increasingly pervasive and decentralized. As the data scale increases, we must address new chal- lenges and attack ever-larger problems. New discoveries will be achieved and more accurate investigations can be carried out due to the increasingly widespread availability of large amounts of data. Scientific sectors that fail to make full use of the huge amounts of digital data available today risk losing out on the significant opportunities that big data can offer. To benefit from the big data availability, specialists and researchers need advanced data analysis tools and applications running on scalable architectures allowing for the extraction of useful knowledge from such huge data sources. High performance computing (HPC) systems and cloud computing systems today are capable
Modified on	22/07/2022 12:21
Name	A view of programming scalable data analysis~ from clouds to exascale
Number of Coding References	26
Number of Codes Coding	4
Coverage	15.77%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	22/07/2022 12:21
Coded Text	Today complex analysis of real-world massive data sources requires using high-performance computing systems such as massively parallel machines or clouds. However in the next years, as parallel technologies advance, Exascale computing systems will be exploited for implementing scalable big data analysis in all the areas of science and engineering [23]. To reach this goal, new design and programming challenges must be addressed and solved. The focus of the paper is on discussing current cloud-based designing and programming solutions for data analysis and suggesting new programming requirements and approaches to be conceived for meeting big data analysis challenges on future Exascale platforms.
Modified on	22/07/2022 12:21

Name	A view of programming scalable data analysis~ from clouds to exascale
Number of Coding References	26
Number of Codes Coding	4
Coverage	15.77%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	
Modified on	10/02/2022 14:15

Name	A view of programming scalable data analysis~ from clouds to exascale
Number of Coding References	26
Number of Codes Coding	4
Coverage	15.77%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	22/07/2022 12:22
Coded Text	
Modified on	22/07/2022 12:25

Name	A view of programming scalable data analysis~ from clouds to exascale
Number of Coding References	26
Number of Codes Coding	4
Coverage	15.77%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	22/07/2022 12:22
Coded Text	Cloud-based data analysis tools Using the DASaaS methodology we designed a cloud-based system, the Data Mining Cloud Framework (DMCF) [17], which supports three main classes of data analysis and knowledge discovery applications: ~ Single-task applications,in whichasingle data mining task such as classification, clustering, or association rules discovery is performed on a given dataset; ~ Parameter-sweeping applications, in which a dataset is analyzed by multiple instances of the same data mining algorithm with different parameters; and ~ Workflow-based applications, in which knowledge discovery applications are specified as graphs linking together data sources, data mining tools, and data mining models. DMCF includes a large variety of processing patterns to express knowledge discovery workflows as graphs whose nodes denote resources (datasets, data analysis
Modified on	22/07/2022 12:22
Name	A view of programming scalable data analysis~ from clouds to exascale
Number of Coding References	26
Number of Codes Coding	4
Coverage	15.77%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	22/07/2022 12:22
Coded Text	code-based (or script-based) formalism allows users to program complex applications more rapidly, in a more concise way, and with higher flexibility [16]. Script-based applications can be designed in different ways (see Fig. 3): ~ With a complete language or a language extension that allows to express parallelism in applications, according to a general purpose or a domain specific approach. This approach requires the design and implementation of a new parallel programming language or a complete set of data types and parallel constructs to be fully inserted in an existing language. ~ With annotations in the application code that allow the compiler to identify which instructions will be executed in parallel. According to this approach, parallel statements are separated from sequential constructs and they are clearly identified in the program code because they are denoted by special symbols or keywords. ~ Using a library in the application code that adds parallelism to the data analysis application
Modified on	22/07/2022 12:24

Name	A view of programming scalable data analysis~ from clouds to exascale
Number of Coding References	26
Number of Codes Coding	4
Coverage	15.77%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	22/07/2022 12:22
Coded Text	Concluding remarks and future work Cloud-based solutions for big data analysis tools and systems are in an advanced phase both on the research and the commercial sides. On the other hand, new Exascale hardware/software solutions must be studied and designed to allow the mining of very large-scale datasets on those new platforms. Exascale systems raise new requirements on applica- tion developers and programming systems to target architectures composed of a very large number of homogeneous and heterogeneous cores. General issues like energy consumption, multitasking, scheduling, reproducibility, and resiliency must be addressed together with other data-oriented issues like data distribution and mapping, data access, data communication and synchronization. Programming constructs and runtime systems will play a crucial role in enabling future data analysis programming models, runtime models and hardware platforms to address these challenges, and in supporting the scalable implementation of real big data analysis applications. In particular, here we summarize a set of open design challenges that are critical for designing Exascale programming systems and for their scalable implementation. The following design choices, among others, must be taken into account: ~ Application reliability: Data analysis programming models must include constructs and/or mechanisms for handling task and data access failures and for recovering. As new data analysis platforms appear ever larger, the fully reliable operations cannot be implicit and this assumption becomes less credible, therefore explicit solutions must be proposed. ~ Reproducibility requirements. Big data analysis running on massively parallel systems demands for reproducibility. New data analysis programming frameworks must collect and generate metadata and provenance information about algorithm characteristics, software configuration and execution environment for supporting application reproducibility on large-scale computing platforms. ~ Communication mechanisms: Novel approaches must be devised for facing network unreliability [7] and network latency, for example by expressing asynchronous data communications and localitybased data exchange/sharing. ~ Communication patterns: A correct paradigm design should include communication patterns allowing application dependent features and data access models, limiting data movement and simplify the burden on Exascale runtimes and interconnection. ~ Data handling and sharing patterns: Data locality mechanisms/constructs, like near-data computing
Modified on	22/07/2022 12:25

Name	A view of programming scalable data analysis~ from clouds to exascale
Number of Coding References	26
Number of Codes Coding	4
Coverage	15.77%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	22/07/2022 12:22
Coded Text	Exascale and big data analysis As we discussed in the previous sections, data analysis gained a primary role because of the very large availability of datasets and the continuous advancement of methods and algorithms for finding knowledge in them. Data analysis solutions advance by exploiting the power of data mining and machine learning techniques and are changing several scientific and industrial areas. For example, the amount of data that social media daily generate is impressive and continuous. Some hundreds of terabyte of data, including several hundreds of millions of photos, are uploaded daily to Facebook and Twitter. Therefore it is central to design scalable solutions for processing and analysis such massive datasets. As a general forecast, IDC experts estimate data generated to reach about 45 zettabytes worldwide by 2020 [6]. This impressive amount of digital data asks for scalable high performance data analysis solutions. However, today only one-quarter of digital data available would be a
Modified on	22/07/2022 12:23

Name	A view of programming scalable data analysis~ from clouds to exascale
Number of Coding References	26
Number of Codes Coding	4
Coverage	15.77%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	22/07/2022 12:22
Coded Text	Extreme data sources and scientific computing Scalability and performance requirements are challenging conventional data storages, file systems and database management systems. Architectures of such systems have reached limits in handling very large processing tasks involving petabytes of data because they have not been built for scaling after a given threshold. This condition claims for new architectures and analytics platform solutions that must process big data for extracting complex predictive and descriptive models [30]. Exascale systems, both from the hardware and the software side, can play a key role to support solutions for these problems [23]. An IBM study reports that we are generating around 2.5 exabytes of data per day.1 Because of that continuous and explosive growth of data, many applications require the use of scalable data analysis platforms. A well-known example is the ATLAS detector from the Large Hadron Collider at CERN in Geneva. The ATLAS infrastructure has a capacity of 200 PB of disk and 300,000 cores, with more than 100 computing centers connected via 10 Gbps links. The data collection rate is very high and only a portion of the data produced by the collider is stored. Several teams of scientists run complex applications to analyze subsets of those huge volumes of data. This analysis would be impossible without a high-performance infrastructure that supports data storage, communication and processing. Also computational astronomers are collecting and producing larger and larger datasets each year that without scalable infrastructures cannot be stored and processed. Another significant case is represented by the Energy Sciences Network (ESnet) is the USA Department of Energy’s high-performance network managed by Berkeley Lab that in late 2012 rolled out a 100 gigabits-per-second national network to accommodate the growing scale of scientific data.
Modified on	22/07/2022 12:23

Name	A view of programming scalable data analysis~ from clouds to exascale
Number of Coding References	26
Number of Codes Coding	4
Coverage	15.77%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	22/07/2022 12:22
Coded Text	However, additional research work in this field must be done and the development of new models, solutions and tools is needed [13, 24]. Just to mention a few, active and promising research topics are listed here ordered by importance factors: ~ Programming models for big data analytics. New abstract programming models and constructs hiding the system complexity are needed for big data analytics tools. The MapReduce model and workflow models are often used on HPC and clouds, but more research effort is needed to develop other scalable, adaptive, general-purpose higher-level models and tools. Research in this area is even more important for Exascale systems; in the next section we will discuss some of these topics in Exascale computing. ~ Reliability in scalable data analysis. As the number of processing elements increases, reliability of systems and applications decreases, therefore mechanisms for detecting and handling hardware and software faults are needed. Although in [7] has been proved that no reliable communication protocol can tolerate crashes of processors on which the protocol runs, as stated in the same paper some ways in which systems cope with the impossibility result can be found. Among them, at programming level it is necessary to design constructs for handling communication, data access, and computing failures and for recovering from them. Programming models, languages and APIs must provide general and data-oriented mechanisms for failure detection and isolation, avoiding that an entire application can Talia Journal of Cloud Computing: Advances, Systems and Applications (2019) 8:4 Page 6 of 16 fail and assuring its completion. Reliability is an issue much more important in the Exascale domain where the number of processing elements is massive and fault occurrence increases making detection and recovering vital. ~ Application reproducibility. Reproducibility is another open research issue for designers of complex applications running on parallel systems. Reproducibility in scalable data analysis must face, for example, with data communication, data parallel manipulation and dynamic computing environments. Reproducibility demands that current data analysis frameworks (like those based on MapReduce and on workflows) and the future ones, especially those implemented on Exascale systems, must provide additional information and knowledge on how data are managed, on algorithm characteristics and on configuration of software and execution environments. ~ Data and tool integration and openness. Code coordination and data integration are main issues in large-scale applications that use data and computing resources. Standard formats, data exchange models and common APIs are needed to support interoperability and ease cooperation among design teams using different data formats and tools. ~ Interoperability ofbig data analytics frameworks. The service-oriented paradigm allows running largescale distributed applications on cloud heterogeneous platforms along with software components developed using different programming languages or tools. Cloud service paradigms must be designed to allow worldwide integration of multiple data analytics frameworks.
Modified on	22/07/2022 12:23

Name	A view of programming scalable data analysis~ from clouds to exascale
Number of Coding References	26
Number of Codes Coding	4
Coverage	15.77%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	22/07/2022 12:22
Coded Text	importance in programming models for data analysis on massively parallel systems. ~ Locality-based data selection and classification for limiting the latency of basic data analysis operations running in parallel on large scale machines in a way that the subset of data needed together in a given phase are locally available (in a subset of nearby cores).
Modified on	22/07/2022 12:25

Name	A view of programming scalable data analysis~ from clouds to exascale
Number of Coding References	26
Number of Codes Coding	4
Coverage	15.77%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	22/07/2022 12:22
Coded Text	Like DMCF, other innovative cloud-based systems de- signed for programming data analysis applications are: Apache Spark, Sphere, Swift, Mahout, and CloudFlows. Most of them are open source. Apache Spark is an open-source framework developed at UC Berkeley for in-memory data analysis and machine learning [34]. Spark has been designed to run both batch processing and dynamic applications like streaming, interactive queries, and graph analysis. Spark provides developers with a programming interface centered on a data structure called the resilient distributed dataset (RDD), that is a read-only multi-set of data items distributed over a cluster of machines, that is maintained in a fault-tolerant way. Differently from other systems and from Hadoop, Spark stores data in memory and queries it repeatedly so as to obtain better performance. This feature can be useful for a future implementation of Spark on Exascale systems. Swift is a workflow-based framework for implementing functional data-driven task parallelism in data-intensive applications.
Modified on	22/07/2022 12:23

Name	A view of programming scalable data analysis~ from clouds to exascale
Number of Coding References	26
Number of Codes Coding	4
Coverage	15.77%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	22/07/2022 12:22
Coded Text	must be designed and evaluated on big data applications when subsets of data are stored in nearby processors and by avoiding that locality is imposed when data must be moved. Other challenges concern data affinity control data querying (NoSQL approach), global data distribution and sharing patterns. ~ Data-parallel constructs: Useful models like datadriven/data-centric constructs, dataflow parallel operations, independent data parallelism, and SPMD patterns must be deeply considered and studied. ~ Grain ofparallelism: from very fine-grain to process-grain parallelism must be analyzed also in combination with the different parallelism degree that Exascale hardware supports. Perhaps different grain size should be considered in a single model to address hardware needs and heterogeneity. Finally, since big data mining algorithms often require the exchange of raw data or, better, of mining parameters and partial models, to achieve scalability and reliability on thousands of processing elements, metadata-based information, limited-communication programming mechanisms, and partition-based data structures with associated parallel operations must be proposed and implemented.
Modified on	22/07/2022 12:25

Name	A view of programming scalable data analysis~ from clouds to exascale
Number of Coding References	26
Number of Codes Coding	4
Coverage	15.77%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	22/07/2022 12:22
Coded Text	Requirements of exascale runtime for data analysis One of the most important aspect to ponder in applications that run on Exascale systems and analyze big datasets is the tradeoff between sharing data among processing elements and computing things locally to reduce communication and energy costs, while keeping performance and fault-tolerance levels. A scalable programming model founded on basic operations for data intensive/data-driven applications must include mechanisms and operations for ~ Parallel data access that allows increasing data access bandwidth by partitioning data into multiple chunks, according to different methods, and accessing several data elements in parallel to meet high throughput requirements. ~ Fault resiliency that is a major issue as machines expand in size and complexity. On Exascale systems with huge amount of processes, non-local communication must be prepared for a potential failure of one of the communication sides; runtimes must features failure handing mechanisms for recovering from node and communication faults. ~ Data-driven local communication that is useful to limit the data exchange overhead in massively parallel systems composed of many cores; in this case data availability among neighbor nodes dictates the operations taken by those nodes. ~ Data processing on limited groups of cores allows concentrating data analysis operations involving limited sets of cores and large amount of data on localities of Exascale machines facilitating a type of data affinity co-locating related data and computation. ~ Near-data synchronization to limit the overhead generated by synchronization mechanisms and protocols that involve several far away cores in keeping data up-to-date. ~ In-memory querying and analytics needed to reduce query response times and execution of analytics operations by caching large volumes of data in the computing node RAMs and issuing queries and other operation in parallel on the main memory of computing nodes. ~ Group-level data aggregation in parallel systems is useful for efficient summarization, graph traversal and matrix operations, therefore it is of great
Modified on	22/07/2022 12:25

Name	A view of programming scalable data analysis~ from clouds to exascale
Number of Coding References	26
Number of Codes Coding	4
Coverage	15.77%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	22/07/2022 12:22
Coded Text	The growing use of service-oriented computing is accelerating the use of cloud-based systems for scalable big data analysis. Developers and researchers are adopting the three main cloud models, software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS), to implement big data analytics solutions in the cloud [27, 31]. According to a specialization of these three models, data analysis tasks and applications can be offered as services at infrastructure, platform or software level and made available every time form everywhere. A methodology for implementing them defines a new model stack to delivery data analysis solutions that is a specialization of the XaaS (Everything as a Service) stack and is called Data Analysis as a Service (DAaaS). It adapts and specifies the three general service models (SaaS, PaaS and IaaS), for supporting the structured development of Big Data analysis systems, tools and applications according to a service-oriented approach. The DAaaS methodology is then based on the three basic models for delivering data analysis services at different levels as described here (see also Fig. 1): ~ Data analysis infrastructure as a service (DAIaaS). This model provides a set of hardware/software virtualized resources that developers can assemble and use as a an integrated infrastructure where storing large datasets, running data mining applications and/or implementing data analytics systems from scratch; ~ Data analysis platform as a service (DAPaaS). This model defines a supporting software platform that developers can use for programming and running their data analytics applications or extending existing ones without concerning about the underlying infrastructure or specific distributed architecture issues; and ~ Data analysis software as a service (DASaaS). This is a higher-level model that offers to end users data mining algorithms, data analysis suites or ready-touse knowledge discovery applications as Internet services that can be accessed and used directly through
Modified on	22/07/2022 12:22

Name	A view of programming scalable data analysis~ from clouds to exascale
Number of Coding References	26
Number of Codes Coding	4
Coverage	15.77%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	22/07/2022 12:22
Coded Text	tools, mining models) and whose edges denote dependencies among resources. A Web-based user interface allows users to compose their applications and submit them for execution to the Cloud platform, following the data analysis software as a service approach. Visual workflows can be programmed in DMCF through a language called VL4Cloud (Visual Language for Cloud), whereas script-based workflows can be programmed by JS4Cloud (JavaScript for Cloud), a JavaScript-based language for data analysis programming.
Modified on	22/07/2022 12:23

Name	Achiever or explorer~ gamifying the creation process of training data for machine learning
Number of Coding References	11
Number of Codes Coding	2
Coverage	5.35%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	10/02/2022 12:05
Coded Text	2.2.2 Tags You Don’t Forget: GamifiedTagging ofPersonal Images. Another approach was created by [20] whose scope was the creation of a game, used to annotate personal photos. Two mobile applications were developed (one single, one multiplayer) and evaluated as well as compared to a simple tagging app without any gami□cation.
Modified on	10/02/2022 12:07
Name	Achiever or explorer~ gamifying the creation process of training data for machine learning
Number of Coding References	11
Number of Codes Coding	2
Coverage	5.35%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	10/02/2022 12:05
Coded Text	2.2.3 Crowdsourcing. Lastly, we analyzed crowdsourcing tools, which often include game elements to engage users. Google Crowdsource [11] is a desktop platform as well as a mobile app, which makes use of humans to improve Google tools such as Google Photos or Google Translate and can be used by anyone who has a Google account.
Modified on	10/02/2022 12:07

Name	Achiever or explorer~ gamifying the creation process of training data for machine learning
Number of Coding References	11
Number of Codes Coding	2
Coverage	5.35%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	10/02/2022 12:05
Coded Text	for certain psychological outcomes such as motivation, enjoyment, and ow. Previous research shows that a gami ed environment for data annotation has the potential to increase user engagement and grati cation [12]. Improved user experience is a goal ofgami cation, as are increased participation, the attraction of a younger audience, optimization ofwork ows and increased engagement of users, as well as immediate feedback for the users on their performance [23]. Gami cation of company workplaces has just recently gained in importance – not only for the training but also to encourage employees in their daily work routine. A tool with well-designed game elements at the workplace can keep employees motivated to perform their tasks [16]. This paper presents the results ofour work aiming at integrating game elements into an existing annotation tool for the creation of training data at the AI product company AI4BD 1. We describe our multi-step development process, thereby laying the foundation for future user studies to investigate the e ect of the implemented game elements.
Modified on	10/02/2022 12:06
Name	Achiever or explorer~ gamifying the creation process of training data for machine learning
Number of Coding References	11
Number of Codes Coding	2
Coverage	5.35%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	10/02/2022 12:05
Coded Text	Gamification in video labeling. A game for video annotation was designed in [21]. They thought out three di erent game approaches: a label vote game, an entity annotation where users were asked to assign a certain category to a video segment, a click game, where users had to locate a certain object inside the video and click on it, and a bounding box game, which asked users to draw a box around a speci c object. The last one was implemented and evaluated with the aid of 20
Modified on	10/02/2022 12:07

Name	Achiever or explorer~ gamifying the creation process of training data for machine learning
Number of Coding References	11
Number of Codes Coding	2
Coverage	5.35%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	10/02/2022 12:05
Coded Text	The creation of necessary labels is usually performed with the aid of humans. Due to the necessary amount of training data the creation process is typically highly repetitive and quickly turns into a rather unexciting, demotivating task for the annotator.
Modified on	10/02/2022 12:06

Name	Achiever or explorer~ gamifying the creation process of training data for machine learning
Number of Coding References	11
Number of Codes Coding	2
Coverage	5.35%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	10/02/2022 12:05

Coded Text	The development ofarti□cial intelligence, e. g., for Computer Vision, through supervised learning requires the input of large amounts of annotated or labeled data objects as training data. The creation of high-quality training data is usually done manually which can be repetitive and tiring. Gami□cation, the use of game elements in a non-game context, is one method to make tedious tasks more interesting. This paper proposes a multi-step process for gamifying the manual creation of training data for machine learning purposes. We choose a user-adapted approach based on the results ofa preceding user study with the target group (employees of an AI software development company) which helped us to identify annotation use cases and the users' player characteristics. The resulting concept includes levels of increasing di□culty, tutorials, progress indicators and a narrative built around a robot character which at the same time is a user assistant.
Modified on	10/02/2022 12:05

Name	Achiever or explorer~ gamifying the creation process of training data for machine learning
Number of Coding References	11
Number of Codes Coding	2
Coverage	5.35%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	03/02/2022 10:47
Coded Text	• "I□nd labeling tasks tiresome" (65% agreed, M=0.7, SD=1.117) • "Iwould like to be able to see howwell Iam doing in labeling, compared to my coworkers" (55% agreed, M=0.2, SD=1.348) • "Iflabeling included game elements, the label results would be better" (50% agreed, M=0.4, SD=0.993) • "Iflabeling included game elements it would be much more fun" (65% agreed, M=0.9, SD=0.999) • "Iwouldnotlike itifotherswere able to seemylabeling progress on a leaderboard" (45% agreed, M=0.35, SD=1.27) • "Using game elements at work makes a company seem less serious" (30% agreed, 55% disagreed, M=-0.65, SD=1.306)
Modified on	10/02/2022 12:08

Name	Achiever or explorer~ gamifying the creation process of training data for machine learning
Number of Coding References	11
Number of Codes Coding	2
Coverage	5.35%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	03/02/2022 10:47
Coded Text	by Callan et al. [6], where ten □ctive scenarios of gami□cation are presented which have been wrongly established in businesses. Recurring problems were a lack of goal-orientation, unsuitable game elements and rewarding, and the danger of revealing too much information to the employees which they might attempt to use for their bene□t. Furthermore, the term addiction is mentioned in this context [2].
Modified on	10/02/2022 12:07

Name	Achiever or explorer~ gamifying the creation process of training data for machine learning
Number of Coding References	11
Number of Codes Coding	2
Coverage	5.35%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	03/02/2022 10:47
Coded Text	Dangers of Gami□cation As gami□cation makes use of game elements, it is necessary to keep in mind that with these elements some of their risks might also be adopted. One way to approach this topic has been executed
Modified on	10/02/2022 12:07

Name	Achiever or explorer~ gamifying the creation process of training data for machine learning
Number of Coding References	11
Number of Codes Coding	2
Coverage	5.35%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	03/02/2022 10:47
Coded Text	The company's existing annotation tool is a multi-user web ap- plication prototype which o□ers registered users a sophisticated annotation environment for collections ofimages (typically scanned documents). The annotation tasks are of four di□erent types: • handwriting annotation, where annotators are given an image of a handwritten sequence of letters and numbers which they have to type, • document classi□cation, where annotators need to classify parts of a document, e. g., to mark tables inside a form using semantic bounding boxes, • classi□cation, where annotators are asked to identify a given object, e. g., if an image contains a number, • natural language processing (NLP), where annotators are asked to assign semantic meaning to words, for example, to mark all persons in a given text.
Modified on	10/02/2022 12:08

Name	Achiever or explorer~ gamifying the creation process of training data for machine learning
Number of Coding References	11
Number of Codes Coding	2
Coverage	5.35%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	03/02/2022 10:47
Coded Text	The creation of necessary labels is usually performed with the aid of humans. Due to the necessary amount of training data the creation process is typically highly repetitive and quickly turns into a rather unexciting, demotivating task for the annotator.
Modified on	10/02/2022 12:06

Name	ADCME MPI~ Distributed machine learning for computational engineering
Number of Coding References	12
Number of Codes Coding	4
Coverage	15.23%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	one challenge with this approach is that for large-scale problems, the memory and computational costs for the numerical solver are prohibitive. The de-facto standard for solving such large-scale problems on modern distributed memory high performance computing (HPC) architectures is the Message Passing Interface (MPI) (Gabriel et al. 2004; Gropp, Thakur, and Lusk 1999). The TensorFlow backend used by ADCME was originally designed for deep learning/machine learning. Despite that there are much work on extending TensorFlow for distributed training of machine learning models, some of the key capabilities, such as distributed linear algebra and domain decomposition, for solving scientific computing problems are still lacking. This paper is about incorporating MPI functionalities into ADCME to achieve scalability and flexibility for distributed memory.
Modified on	11/02/2022 14:32

Name	ADCME MPI~ Distributed machine learning for computational engineering
Number of Coding References	12
Number of Codes Coding	4
Coverage	15.23%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	The main idea is to parallelize numerical solvers by split- ting the mesh or matrices onto different MPI ranks (or processors). Then, data communication nodes are inserted into the computational graph. Because for our computational engineering applications DNNs are typically small, they are duplicated on each processor. Each computational graph includes a set of “communication” nodes (Fig. 1-top), which are absent in a single processor computational graph. These operators invoke MPI calls and are in charge of data communication between different computer nodes. During the gradient back-propagation, we need to reverse the data-flow direction and operation of the data communication operators
Modified on	11/02/2022 14:32

Name	ADCME MPI~ Distributed machine learning for computational engineering
Number of Coding References	12
Number of Codes Coding	4
Coverage	15.23%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 23:01
Coded Text	The main idea is to parallelize numerical solvers by split- ting the mesh or matrices onto different MPI ranks (or processors). Then, data communication nodes are inserted into the computational graph. Because for our computational engineering applications DNNs are typically small, they are duplicated on each processor. Each computational graph includes a set of “communication” nodes (Fig. 1-top), which are absent in a single processor computational graph. These operators invoke MPI calls and are in charge of data communication between different computer nodes. During the gradient back-propagation, we need to reverse the data-flow direction and operation of the data communication operators
Modified on	11/02/2022 14:32

Name	ADCME MPI~ Distributed machine learning for computational engineering
Number of Coding References	12
Number of Codes Coding	4
Coverage	15.23%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	
Modified on	11/02/2022 14:32

Name	ADCME MPI~ Distributed machine learning for computational engineering
Number of Coding References	12
Number of Codes Coding	4
Coverage	15.23%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	
Modified on	11/02/2022 14:33

Name	ADCME MPI~ Distributed machine learning for computational engineering
Number of Coding References	12
Number of Codes Coding	4
Coverage	15.23%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	In the strong scaling experiments, we consider a fixed problem size 1,800 × 1,800 (mesh size, which implies the matrix size is around 32 million × 32 million). In the weak scaling experiments, each MPI processor owns a 300 × 300 block. For example, a problem with 3,600 processors has the problem size 90,000 × 3,600 ≈ 0.3 billion. We first consider the weak scaling case. We consider two cases: each MPI rank has 1 core or 4 cores. In the latter case, the TensorFlow backend enjoys the benefit of inter-parallelism, where independent operators in the computational graph can be executed simultaneously. However, 4 cores do not guarantee a 4 times acceleration; the performance depends on the availability of independent tasks, scheduling conflicts, resource contention, etc. Fig. 5 shows the runtime for the forward computation as well as the gradient back-propagation. There are two important observations: 1. By using more cores per processor, the runtime is reduced significantly. For example, the runtime for the backward is reduced to around 10 seconds from 30 seconds by switching from 1 core to 4 cores per processor. 2. The runtime for the backward pass is typically less than twice the forward computation. Although the backward pass requires solving two linear systems (one of them is in the forward computation), the AMG (algebraic multigrid) linear solver in the back-propagation may converge faster, and therefore may cost less than during the forward pass. Additionally, we show the overhead in Fig. 6, which is defined as the difference between total runtime and Hypre linear solver time, for both the forward and backward calculation. We see that the overhead is quite small compared to the total time, especially when the problem size is large. This indicates that the ADCME MPI implementation is very effective. In Fig. 7, we consider the strong scaling. In this case, we fixed the whole problem size and split the mesh onto different MPI processors. Fig. 7 shows the runtime for the forward computation and the gradient back-propagation. We can reduce the runtime by more than 20 times for the expensive gradient back-propagation by utilizing more than 100 MPI processors. Fig. 8 shows the speedup and efficiency. We can see that the 4 cores have smaller runtime compared to 1 core
Modified on	11/02/2022 14:33

Name	ADCME MPI~ Distributed machine learning for computational engineering
Number of Coding References	12
Number of Codes Coding	4
Coverage	15.23%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	Methodology ADCME MPI aims at providing a modular, efficient, and flexible implementation for distributed computing in inverse modeling. Conceptually, we can treat data communication operations as a node in the computational graph: they are similar to computational nodes (e.g., a linear solver), except that their responsibility is to invoke MPI calls and backpropagate the gradients in the reverse mode automatic differentiation (Baydin et al. 2017). This solution provides an elegant enhancement to the ADCME library because to convert a single processor program to multiple processor one, users only need to insert data communication nodes as needed and most parts of the original codes are unchanged. In this section, we briefly describe our contributions in ADCME MPI to extend its distributed computing capabilities to couple DNNs and PDE solvers. MPI APIs ADCME MPI provides a set of commonly used MPI primitives, such as mpi bcast, mpi gather, mpi send, etc. These operators are wrappers for standard MPI APIs. However, these operators are also “differentiable,” in the sense that they can handle gradient back-propagation. The gradient back-propagation functionality uses the fact that there exists one-to-one correspondence between forward and backward MPI calls. For example, the forward “send” corresponds to the backward “receive” (Cheng 2006; Towara, Schanen, and Naumann 2015).
Modified on	11/02/2022 14:33
Name	ADCME MPI~ Distributed machine learning for computational engineering
Number of Coding References	12
Number of Codes Coding	4
Coverage	15.23%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	We propose a framework for training deep neural networks (DNNs) that are coupled with partial differential equations (PDEs) in a parallel computing environment. Unlike most distributed computing frameworks for DNNs, our focus is to parallelize both numerical solvers and DNNs in forward and adjoint computations. Our parallel computing model views data communication as a node in the computational graph for numerical simulations. The advantage of our model is that data communication and computing are cleanly separated, which enables better flexibility, modularity, and testability of the software. We demonstrate our approach on a large-scale problem and show that we can achieve substantial acceleration by using parallel numerical PDE solvers while training DNNs that are coupled with PDEs.
Modified on	11/02/2022 14:30

Name	ADCME MPI~ Distributed machine learning for computational engineering
Number of Coding References	12
Number of Codes Coding	4
Coverage	15.23%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	
Modified on	11/02/2022 14:32

Name	ADCME MPI~ Distributed machine learning for computational engineering
Number of Coding References	12
Number of Codes Coding	4
Coverage	15.23%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	In the strong scaling experiments, we consider a fixed problem size 1,800 × 1,800 (mesh size, which implies the matrix size is around 32 million × 32 million). In the weak scaling experiments, each MPI processor owns a 300 × 300 block. For example, a problem with 3,600 processors has the problem size 90,000 × 3,600 ≈ 0.3 billion. We first consider the weak scaling case. We consider two cases: each MPI rank has 1 core or 4 cores. In the latter case, the TensorFlow backend enjoys the benefit of inter-parallelism, where independent operators in the computational graph can be executed simultaneously. However, 4 cores do not guarantee a 4 times acceleration; the performance depends on the availability of independent tasks, scheduling conflicts, resource contention, etc. Fig. 5 shows the runtime for the forward computation as well as the gradient back-propagation. There are two important observations: 1. By using more cores per processor, the runtime is reduced significantly. For example, the runtime for the backward is reduced to around 10 seconds from 30 seconds by switching from 1 core to 4 cores per processor. 2. The runtime for the backward pass is typically less than twice the forward computation. Although the backward pass requires solving two linear systems (one of them is in the forward computation), the AMG (algebraic multigrid) linear solver in the back-propagation may converge faster, and therefore may cost less than during the forward pass. Additionally, we show the overhead in Fig. 6, which is defined as the difference between total runtime and Hypre linear solver time, for both the forward and backward calculation. We see that the overhead is quite small compared to the total time, especially when the problem size is large. This indicates that the ADCME MPI implementation is very effective. In Fig. 7, we consider the strong scaling. In this case, we fixed the whole problem size and split the mesh onto different MPI processors. Fig. 7 shows the runtime for the forward computation and the gradient back-propagation. We can reduce the runtime by more than 20 times for the expensive gradient back-propagation by utilizing more than 100 MPI processors. Fig. 8 shows the speedup and efficiency. We can see that the 4 cores have smaller runtime compared to 1 core
Modified on	11/02/2022 14:33

Name	ADCME MPI~ Distributed machine learning for computational engineering
Number of Coding References	12
Number of Codes Coding	4
Coverage	15.23%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Methodology ADCME MPI aims at providing a modular, efficient, and flexible implementation for distributed computing in inverse modeling. Conceptually, we can treat data communication operations as a node in the computational graph: they are similar to computational nodes (e.g., a linear solver), except that their responsibility is to invoke MPI calls and backpropagate the gradients in the reverse mode automatic differentiation (Baydin et al. 2017). This solution provides an elegant enhancement to the ADCME library because to convert a single processor program to multiple processor one, users only need to insert data communication nodes as needed and most parts of the original codes are unchanged. In this section, we briefly describe our contributions in ADCME MPI to extend its distributed computing capabilities to couple DNNs and PDE solvers. MPI APIs ADCME MPI provides a set of commonly used MPI primitives, such as mpi bcast, mpi gather, mpi send, etc. These operators are wrappers for standard MPI APIs. However, these operators are also “differentiable,” in the sense that they can handle gradient back-propagation. The gradient back-propagation functionality uses the fact that there exists one-to-one correspondence between forward and backward MPI calls. For example, the forward “send” corresponds to the backward “receive” (Cheng 2006; Towara, Schanen, and Naumann 2015).
Modified on	11/02/2022 14:33

Name	ADCME MPI~ Distributed machine learning for computational engineering
Number of Coding References	12
Number of Codes Coding	4
Coverage	15.23%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	We propose a framework for training deep neural networks (DNNs) that are coupled with partial differential equations (PDEs) in a parallel computing environment. Unlike most distributed computing frameworks for DNNs, our focus is to parallelize both numerical solvers and DNNs in forward and adjoint computations. Our parallel computing model views data communication as a node in the computational graph for numerical simulations. The advantage of our model is that data communication and computing are cleanly separated, which enables better flexibility, modularity, and testability of the software. We demonstrate our approach on a large-scale problem and show that we can achieve substantial acceleration by using parallel numerical PDE solvers while training DNNs that are coupled with PDEs.
Modified on	11/02/2022 14:30

Name	AI Explainability 360~ Impact and Design
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	AI Explainability 360~ Impact and Design Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	AI lifecycle models need to be revised
Number of Coding References	31
Number of Codes Coding	7
Coverage	15.46%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	
Modified on	10/02/2022 11:57

Name	AI lifecycle models need to be revised
Number of Coding References	31
Number of Codes Coding	7
Coverage	15.46%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	
Modified on	10/02/2022 11:57

Name	AI lifecycle models need to be revised
Number of Coding References	31
Number of Codes Coding	7
Coverage	15.46%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	6.1 Implications We see the following implications of our results for the fintech industry and for research. 6.1.1 Implications for Machine Learning Practitioners Machine Learning practitioners have to be aware of extra steps and challenges in their process of developing Machine Learning applications. Although not mentioned in existing lifecycle models, the undertaking of feasibility assessments, documentation, and model monitoring, are crucial while developing Machine Learning applications.
Modified on	10/02/2022 11:59

Name	AI lifecycle models need to be revised
Number of Coding References	31
Number of Codes Coding	7
Coverage	15.46%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	6.1.2 Implications for Process Architects Existing lifecycle models provide a canonical overview of the multiple stages in the lifecycle of a Machine Learning application. However, when being applied to a particular context, such as fintech, these models need to be adapted. From our findings, we suspect that this is also the case for other fields where AI is getting increasing importance.
Modified on	10/02/2022 11:59

Name	AI lifecycle models need to be revised
Number of Coding References	31
Number of Codes Coding	7
Coverage	15.46%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	6.1.3 Implications for Researchers Researchers could focus on solving the reported challenges in the Machine Learning lifecycle with additional tool support and reveal challenges of the ML lifecycle in other domains by extending the case study to more organizations and different types of industries. More automation is required for exploratory data analysis and data integration techniques (Mitchell et al. 2019; Damiani and Frati 2018). Moreover, there are minimal advancements in documentation of Machine Learning projects. Techniques ought to be studied to help trace documentation back to the codebase and vice versa.
Modified on	10/02/2022 11:59
Name	AI lifecycle models need to be revised
Number of Coding References	31
Number of Codes Coding	7
Coverage	15.46%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	6.1.4 Implications for Tool Developers Although a number of tools are emerging to aid ML engineering, these solutions fail to address the singularities of different projects. Thus, practitioners are adopting their own customized solutions. For example, spreadsheets are still being used to manually log experiments regardless of the existing automated solutions, such as MLFlow, DVC, Replicate, and so on. It is important to understand what is missing in the current solutions and how we can propose a solution that effectively solves version control to keep track of changes in data, changes in scoring metrics, and executions of different experiments.
Modified on	10/02/2022 11:59

Name	AI lifecycle models need to be revised
Number of Coding References	31
Number of Codes Coding	7
Coverage	15.46%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	6.1.5 Implications for Educators Page 25 of 29 95 Education of Machine Learning should focus on the whole lifecycle of Machine Learning development, including exploratory analysis with a focus on statistics, data analysis and data visualization. Moreover, practitioners with background on both data science and software engineering are a valuable resource for organizations.
Modified on	10/02/2022 11:59

Name	AI lifecycle models need to be revised
Number of Coding References	31
Number of Codes Coding	7
Coverage	15.46%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	6.1.6 Implications for Organizations Embracing AI The embrace of AI stretches the adequacy of well-established processes at organizations. Multi-disciplinary teams are essential to embrace AI: AI experts have the knowledge to try innovative approaches, but will likely have little expertise to identify business value. Thus, knowledge transfer between stakeholders is challenging and might hinder the motivation of developers. New strategies must be outlined to reduce the amount of effort required to document AI projects.
Modified on	10/02/2022 12:00

Name	AI lifecycle models need to be revised
Number of Coding References	31
Number of Codes Coding	7
Coverage	15.46%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	Moreover, model risk experts are now required to have a strong background in two disjoints fields: 1) Governance, Risk Management, and Compliance and 2) AI. We conjecture
Modified on	10/02/2022 11:58

Name	AI lifecycle models need to be revised
Number of Coding References	31
Number of Codes Coding	7
Coverage	15.46%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	Technology Access All AI technologies, tools, and libraries need to be audited to make sure they are safe to be used in fintech applications. Only then, practitioners are able to design their Machine Learning systems around the latest technology. This is a challenge that needs to tackled by any organization akin to ING. As presented in Section 4.4, this process can be limiting since new AI technologies are appearing every day. Practitioners willing to try the latest AI technology may feel less motivated since it may take some time before they are approved. As referred in Section 4.1, many problems at ING are triggered by the Technology push. Hence, new business opportunities might be missed if practitioners are not able to experiment the latest AI technologies. We do not know to what extent Technology Access is also a challenge to software orga- nizations operating in other domains. Previous work suggests that only 8% of software developers consider an organization’s culture and policies highly-influential when selecting third-party software libraries (Larios Vargas et al. 2020).
Modified on	10/02/2022 11:59

Name	AI lifecycle models need to be revised
Number of Coding References	31
Number of Codes Coding	7
Coverage	15.46%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13
Coded Text	
Modified on	10/02/2022 11:54

Name	AI lifecycle models need to be revised
Number of Coding References	31
Number of Codes Coding	7
Coverage	15.46%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13
Coded Text	All projects must goover a feasibility study in their early stages. Until then, projects do not fit thetypicalsprint-based agile planning.Anagile approachhelps practitioners prioritize tasks andengagestakeholders.
Modified on	10/02/2022 11:53

Name	AI lifecycle models need to be revised
Number of Coding References	31
Number of Codes Coding	7
Coverage	15.46%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13
Coded Text	Although Model Risk Assessmentis not new to thefintech industry,MachineLearn- ing is requiring a revised approach.Currently,developers endure considerable efforts to create therequired documentation.
Modified on	10/02/2022 11:52

Name	AI lifecycle models need to be revised
Number of Coding References	31
Number of Codes Coding	7
Coverage	15.46%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13
Coded Text	Although practitioners are eager to learnautomated testing practices, this is not part oftheir skillset. Hence, projects are struggling to adopt unit andintegration testing strategies.
Modified on	10/02/2022 11:53

Name	AI lifecycle models need to be revised
Number of Coding References	31
Number of Codes Coding	7
Coverage	15.46%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13
Coded Text	Collecting, understanding, andpreparing dataare the most time-consuming stages ofMachineLearning projects. There is ameticulousdataaccess control that, despite being quintessential, sets major obstacles inunderstanding thedataand performing exploratory analyses. Practitioners emphasize, dataunderstanding implies being able to communicate it to other stakeholders. Finally,the differences between development andproduction environments pose challenges for data preparation.
Modified on	10/02/2022 11:52

Name	AI lifecycle models need to be revised
Number of Coding References	31
Number of Codes Coding	7
Coverage	15.46%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13

Coded Text	Documentation is a first-class artifact for regulatory compliance, knowledge transfer, andreproducibility.Hence, a peer-review process is in place to ensure documentation quality.
Modified on	10/02/2022 11:52

Name	AI lifecycle models need to be revised
Number of Coding References	31
Number of Codes Coding	7
Coverage	15.46%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13
Coded Text	MachineLearning projects start witha problem statement which is used to discuss whether a MachineLearning solution is necessary.This step requires high engagement from problem domain experts.
Modified on	10/02/2022 11:52

Name	AI lifecycle models need to be revised
Number of Coding References	31
Number of Codes Coding	7
Coverage	15.46%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13
Coded Text	More automation is needed for model monitoring.Teams havecreated their own automation tools, but making themavailable to other teamsrequires unfeasible efforts thatdo not meet their priorities.
Modified on	10/02/2022 11:53

Name	AI lifecycle models need to be revised
Number of Coding References	31
Number of Codes Coding	7
Coverage	15.46%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13
Coded Text	Requirements are not alwaysdefined beforehand. DataandModel requirements become more clear while working withan initial model. Requirements relatedtotraceability,interpretability, andexplainabilityare typically defined a theorganizational level.
Modified on	10/02/2022 11:52

Name	AI lifecycle models need to be revised
Number of Coding References	31
Number of Codes Coding	7
Coverage	15.46%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13
Coded Text	Thechallenges in Modeling summarize as follows: 1) thelatest MachineLearning technologies are not alwayseligible for use;2)baseline models are essential artifacts for model development; 3) teams keep track of all experiments, which often revolves around keeping a customized spreadsheet; and4)defining performance metrics is problem-specific, posing a challengeto thedefinition ofstandards attheorganizational level.
Modified on	10/02/2022 11:52

Name	AI lifecycle models need to be revised
Number of Coding References	31
Number of Codes Coding	7
Coverage	15.46%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13
Coded Text	There are deploymentpatternsinwhicha separate team needs to reimplementthe model to meet production settings.
Modified on	10/02/2022 11:52

Name	AI lifecycle models need to be revised
Number of Coding References	31
Number of Codes Coding	7
Coverage	15.46%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13
Coded Text	There is practical valueonhaving a strong backgroundon bothSoftware Engi- neering and DataScience. Education should put more focuson theprocess insteadof model-training techniques.
Modified on	10/02/2022 11:53

Name	AI lifecycle models need to be revised
Number of Coding References	31
Number of Codes Coding	7
Coverage	15.46%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13
Coded Text	We have found that the following stages have been overlooked by previous lifecycle models: data collection, feasibility study, documentation, model monitoring,and model risk assessment.Our work shows that the real challenges of applying Machine Learning go much beyond sophisticated learning algorithms – more focus is needed on the entire lifecycle. In particular, regardless of the existing development tools for Machine Learning, we observe that they are still not meeting the particularities of this field.
Modified on	10/02/2022 11:49

Name	AI lifecycle models need to be revised
Number of Coding References	31
Number of Codes Coding	7
Coverage	15.46%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	06/01/2022 15:15
Coded Text	4.7 Model Deployment We observed three deployment patterns at ING: 1. A specialized team creates a prototype with a validated methodology, and an engineering team takes care of reimplementing it in a scalable, ready-to-deploy fashion. In some cases, this is a necessity due to the technical requirements of the model, e.g., when models are developed in Python, but should be deployed in Java (P08, P09, P13). 2. A specialized team creates a model and exports its configuration (e.g., a pickle9 and required dependencies) to a system that will semi-automatically bundle it and deploy it without changing the model (P01, P09). 3. The same team takes care of creating the model and taking it into production. This mostly means that software engineers are part of the team and a structured and strict software architecture is ensured. Similar to the training environments, Machine Learning systems are deployed to on- premises environments. A reported challenge regarding the deployment environment is that different hardware and platform parameters (e.g., Spark parameters) can result in different model behavior or errors (P16). For example, the deployment environment may have less memory than the training environment. Furthermore, the resources for a Machine Learning system are dynamically allocated whenever needed. However, it is not trivial understanding when a system is no longer needed and should be scaled down to zero (P01). There are deploymentpatternsinwhicha separate team needs to reimplementthe model to meet production settings.
Modified on	24/02/2022 13:11

Name	AI lifecycle models need to be revised
Number of Coding References	31
Number of Codes Coding	7
Coverage	15.46%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	04/02/2022 21:53
Coded Text	4.8 Model Monitoring After having a model in production, it is necessary to keep track of its behavior to make sure it operates as expected. It implies testing the model while the model is deployed online. The main advantage is that it uses real data. Previous work refers to this stage as online testing (Zhang et al. 2020).
Modified on	24/02/2022 13:09

Name	AI lifecycle models need to be revised
Number of Coding References	31
Number of Codes Coding	7
Coverage	15.46%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 11:04
Coded Text	4.6 Model Evaluation An essential step in the evaluation of the model is communicating how well the model performs according to the defined metrics. It is about demonstrating that the model meets business and regulatory needs and assessing the design of the model. One key difference between the metrics used in this step and the metrics used for Model Scoring is that these metrics are communicated to different stakeholders that do not necessarily have a Machine Learning or data science background. Thus, the set of metrics needs to be extended to a general audience. One complementary strategy used by practitioners is having live demos of the model with business stakeholders (P03, P15, P16). These demos allow stakeholders to try out different inputs and try corner cases. 4.6.1 Model Risk Assessment An important aspect of evaluating a model at ING is making sure it complies with regulations, ethics, and organizational values (P15, P06). This is a common task for any type of model built within the organization – i.e., not only Machine Learning models but also economic models, statistical forecasting models, and so on. In the interviews, Model Risk Assessment was mentioned as mandatory within the model governance strategy, undertaken in collaboration with an independent specialized team (P06, P14). This is a long-stablished stage which is now being challenged by the specifics of Machine Learning. For example, traditional risk assessment teams did not initially have the right Machine Learning expertise to evaluate the models with confidence. Depending on the criticality level of the model, the intensity of the review may vary. Each model owner is responsible for the risk management of their model, but colleagues from the risk department help and challenge the model owner in this process. During the periodic risk assessment process, assessors inspect the documentation pro- vided by the Machine Learning team to assess whether all regulations and minimum standards are followed. The documentation used in this stage is considered to be overly time-consuming, as emphasized by P07: “70% percent of the time people are writing Word documents to explain their code is compliant.”. Although the process is still under development within ING, the following key points are being covered (P06): 1) model identification 95 Page 16 of 29 Empir Software Eng (2021) 26: 95 (identify if the candidate is a model which needs risk management), 2) model boundaries (define which components are part of the model), 3) model categorization (categorize the model into the group of models with a comparable nature, e.g. anti-money-laundering), 4) model classification (classify the model into in the class of models which require a comparable level of model risk management), and 5) assess the model by a number of sources of risk. Although Model Risk Assessmentis not new to thefintech industry,MachineLearn- ing is requiring a revised approach.Currently,developers endure considerable efforts to create therequired documentation.
Modified on	24/02/2022 13:10

Name	AI lifecycle models need to be revised
Number of Coding References	31
Number of Codes Coding	7
Coverage	15.46%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	10/02/2022 11:25
Coded Text	4.4 Modeling Model training is mostly done in on-premises environments such as Hadoop2 and Spark3 clusters (P09) or in generic systems using, for example, the scikit-learn4 library (P01). These private platforms are connected with the data lakes where data is stored, so training can be done on (a copy of) real production data (P01, P03). The on-premises environment has no outgoing connection to the internet, so a connection to other cloud services such as Microsoft Azure5 or Google Cloud6 is not possible (P08). This means that data scientists are limited to the tools and platforms available within the organization when dealing with sensitive data. Also, all project dependencies need to be previously approved, after which they are made available in a private package repository (P04, P12), which contains whitelisted packages that have been internally audited. This can be frustrating, when new ground-breaking AI technologies appear, practitioners have to wait before they can explore the potential of those technologies at ING (P12) – we later refer to this challenge as Technology Access (cf. Section 5). Fewer restrictions are in place if Machine Learning is applied to public data, for example on stock prices. In that case, external cloud services and packages may be used (P09). 2Hadoop enables distributed processing of large data sets across clusters of computers https://hadoop.apache. org 3Spark is a unified analytics engine for large-scale data processing. https://spark.apache.org 4Scikit-learn is a Machine Learning library for Python. https://scikit-learn.org 5Microsoft Azure is a cloud computing service. https://azure.microsoft.com/en-us 6Google Cloud is a cloud computing service. https://cloud.google.com 95 Page 14 of 29 Empir Software Eng (2021) 26: 95 Model training is an iterative process. Usually, multiple models are created for the same problem. First, a simple model is created (e.g., a linear regression model) to set as a baseline (P09). In the following iterations, more advanced models are compared to this baseline model. If an approach other than Machine Learning already exists (e.g., rule-based software), the models are also compared with this. To keep track of different versions of models, different teams use different strategies. For example, the team of P08 keeps track of an experiment log using a spreadsheet, in which the training set, validation set, model, and pre-processing steps are specified for each version. This approach for versioning is preferred over solutions like MLFlow7 for the sake of simplicity (P08, P15). 4.4.1 Model Scoring An implicit sub stage of modeling is assessing model performance to measure how well the predictions of the model represent ground truth data. We define Model Scoring as assessing the performance of the model based on scor- ing metrics (e.g., f1-score for supervised learning). It is also known as Validation by the Machine Learning community, which should not be confused with the definition by the Software Engineering community8 (Ryan and Wheatcraft 2017; 15288 2015). The main remarks for this stage are related to defining the right set of metrics (P03, P06, P12, P14, P15, P16). The problem is two-fold: 1) identify the right metrics and 2) communicate why the selected metrics are right. Practitioners report that this is very problem-specific. Thus, it requires a good understanding of the business, data, and learning algorithms being used. From an organization’s point of view, these different perspectives are a big barrier to defining validation standards. Thechallenges in Modeling summarize as follows: 1) thelatest MachineLearning technologies are not alwayseligible for use;2)baseline models are essential artifacts for model development; 3) teams keep track of all experiments, which often revolves around keeping a customized spreadsheet; and4)defining performance metrics is problem-specific, posing a challengeto thedefinition ofstandards attheorganizational level.
Modified on	24/02/2022 13:12

Name	AI lifecycle models need to be revised
Number of Coding References	31
Number of Codes Coding	7
Coverage	15.46%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56
Coded Text	Based on CRISP-DM, a number of lifecycle models have been proposed (Mart´ınez- Plumed et al. 2019; Mariscal et al. 2010) to address varying objectives. Derived models extend CRISP-DM for projects with geographically dispersed teams (Moyle and Jorge 2001), with large amounts of data and more focus on automation (Wu et al. 2013; Rollins 2015), or targeting the model reuse across different contexts (Mart´ınez-Plumed et al. 2017). TDSP is “an agile, iterative data science methodology” proposed by Microsoft, to deliver Machine Learning solutions efficiently (Ericson et al. 2017). The original methodology includes four major stages, as can be seen in Fig. 2: Business Understanding, Data Acquisition, Modeling and Deployment. As depicted by the arrows in the figure, TDSP proposes stronger dependencies but does not enforce a particular order between stages, emphasizing that different stages can be iteratively repeated at almost any time in the project. Amershi et al. (2019) describe the nine stages followed by software engineering teams at Microsoft who are integrating machine learning into application and platform development. The workflow is presented in Fig. 3, with nine stages: Model Requirements, Data Collection, Data Cleaning, Data Labeling, Feature Engineering, Model Training, Model Evaluation, Model Deployment,and Model Monitoring. The large feedback arrows in the figure depict stages that can be followed by any of their precedent stages. It is the case of Model Evaluation and Model Monitoring. The smaller feedback arrow shows that Model Training and Feature Engineering are typically revisited iteratively. Despite the number of advancements proposed in previous work, we argue that they do not tackle AI systems that target challenges faced by the fintech industry. Our work pinpoints the changes that needed to be accommodated for AI systems operating under heavy-regulated scenarios and bringing value over pre-existing non-data-driven approaches.
Modified on	10/02/2022 11:50

Name	AI lifecycle models need to be revised
Number of Coding References	31
Number of Codes Coding	7
Coverage	15.46%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56
Coded Text	In this study, we consider three reference models for the lifecycle of Machine Learning applications: Cross-Industry Standard Process for Data Mining (CRISP-DM) (Shearer 2000), the Team Data Science Process (TDSP) (Ericson et al. 2017), and the Microsoft model described by Amershi et al. (2019). We chose CRISP-DM, as although it is twenty years old, it is still the de facto standard for developing data mining and knowledge discovery projects (Mart´ınez-Plumed et al. 2019). We selected TDSP as modern industry methodology, which has at a high level much in common with CRISP-DM. Finally, we also include the model described by Amershi et al., which is based on CRISP-DM and TDSP and addresses the workflow of software engineering teams (Amershi et al. 2019). There are other methodologies, but most are similar to these three. Findings in our paper can be extrapolated to those other lifecycle models. CRISP-DM aims to provide anyone with “a complete blueprint for conducting a data mining project” (Shearer 2000). Although data mining is not the common term used nowadays, it is valid for any project applying scientific methods to extracting value from data, including Machine Learning (Mart´ınez-Plumed et al. 2019). CRISP-DM breaks down a project into six phases, as presented in Fig. 1. It typically starts with Business Understanding to determine business objectives, going back and forward with Data Understanding.It is followed by Data Preparation to make data ready for Modeling. The produced model goes through an Evaluation in which it is decided whether the model can go for Deployment or it needs another round of improvement. The arrows between stages indicate the most relevant and recurrent dependencies, while the arrows in the outer circle indicate the evolution of Machine Learning systems after being deployed and their iterative nature.
Modified on	10/02/2022 11:50

Name	AI lifecycle models need to be revised
Number of Coding References	31
Number of Codes Coding	7
Coverage	15.46%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56
Coded Text	practitioner working on a machine learning system. Moreover, we argue that Microsoft as a long history in developing machine learning systems, which might neglect some of the challenges that organizations shifting to AI have to endure. Finally, we compare our observations with existing Machine Learning lifecycle models, including the one proposed by Amershi et al.. Another case study from industry has been performed at Booking.com by Bernardi et al. (2019). In contrast with academic research in which Machine Learning models are validated by means of an error measurement, models at Booking.com are validated through business metrics such as conversion or cancellations. The paper describes process stages such as model designing, deployment, monitoring, and evaluation, but no formal lifecycle model is defined. Moreover, we hypothesize that the fintech domain poses extra challenges that stem from having to adhere to heavy regulations. Hill et al. (2016) studied how people develop intelligent systems in practice. The study leverages a high-level model of the process and identifies the main challenges. Results show that developers struggle with establishing repeatable processes and that there is a basic mismatch between the tools available versus the practical needs. In this study, we extend the work by Hill et al. by looking more closely at what happens after the Machine Learning model has been evaluated, for example regarding its deployment and monitoring. The paper by Lin and Ryaboy (2013) describes the big data mining cycle at Twitter, based on the experience of the two authors. The main points made are that for data-driven projects, most time goes to preparatory work before, and engineering work after the actual model training and that a significant amount of tooling and infrastructure is required. In our study, we validate the recommendations of these two experts with a case study with seventeen participants. Concrete challenges data scientists face are elaborated upon in the study by Kim et al. (2017). They have surveyed 793 professional data scientists at Microsoft. An example of a challenge found is that the proliferation of data science tools makes it harder to reuse work across teams. This challenge is also reinforced in the study by Ahmed et al. (2019). As models are mostly implemented without standard API, input format, or hyperparameter notation, data scientists spend considerable effort on implementing glue code and wrappers around different algorithms and data formats to employ them in their pipelines. Ahmed et al. (2019) show evidence that most models need to be rewritten by a different engineering team for deployment. The root of this challenge lies on runtime constraints, such as a different hardware or software platform, and constraints on the pipeline size or prediction latency. More studies looked at Machine Learning from a Software Engineering viewpoint. Scul- ley et al. (2015) identified a number of Machine Learning-specific factors that increase technical debt, such as boundary erosion and hidden feedback loops. Breck et al. (2017) have proposed 28 specific tests for assessing production readiness for Machine Learning applications. These tests include tests for features and data, model development, infrastructure, and monitoring. Arpteg et al. (2018) have identified Software Engineering challenges of building intelligent systems with deep learning components based on seven projects from companies of different types and sizes. These challenges include development, production, and organizational challenges, such as experiment management, dependency management, and effort estimation. In this current study, we will extend this line of research and identify where Software Engineering can help mitigate inefficiencies in the development and evolution of Machine Learning systems.
Modified on	22/07/2022 13:00

Name	AI lifecycle models need to be revised
Number of Coding References	31
Number of Codes Coding	7
Coverage	15.46%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56
Coded Text	The Machine Learning development lifecycle has been studied in practice in previous research. Amershi et al. (2019) have conducted a case study at Microsoft to study the differences between Software Engineering and Machine Learning. They interviewed 15 software engineers and a conducted a survey with 551 software engineers, yielding 4 main contributions: 1) a description of a machine learning workflow, that we use for comparison with our study; 2) a set of best practices for building applications with machine learning models, 3) a preliminary maturity model for teams developing machine learning applications, and 4) a discussion of the fundamental differences between developing software systems that integrate machine learning models and traditional software systems. According to Amershi et al. (2019), the main differences that set machine learning apart from traditional software engineering can be summarized as follows: a) handling the data needed for Machine Learning applications is considerably more complex, b) model customization and reuse require a wide set of skills that are not typically found in software teams, c) it is hard to isolate two different machine learning models that operate in the same system – often they ought to be developed and training together. We complement this study by not restricting our observations to software engineering teams. Hence, we include any Fig. 3 Microsoft's machine learning workflow described by Amershi et al. (2019)
Modified on	10/02/2022 11:51

Name	All versus one~ an empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software
Number of Coding References	26
Number of Codes Coding	4
Coverage	12.43%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	10/02/2022 11:26
Coded Text	
Modified on	10/02/2022 11:43

Name	All versus one~ an empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software
Number of Coding References	26
Number of Codes Coding	4
Coverage	12.43%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	10/02/2022 11:26
Coded Text	
Modified on	10/02/2022 11:43

Name	All versus one~ an empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software
Number of Coding References	26
Number of Codes Coding	4
Coverage	12.43%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	10/02/2022 11:26
Coded Text	
Modified on	10/02/2022 11:44

Name	All versus one~ an empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software
Number of Coding References	26
Number of Codes Coding	4
Coverage	12.43%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	10/02/2022 11:26
Coded Text	For RQ1, we obtained the following findings: Finding 1: The retrained version of a given learning algorithm does not always lead to higher accuracy than its incremental counterpart. In fact, the winner on accuracy can be considerably affected by the actual learning algorithm, i.e., incremental modeling is better with MLP while the retrained one is better with SVM, and the characteristics of subject adaptable software, i.e., the incremental modeling is more accurate for highly fluctuated adaptable software while the retrained one is better for stable software. Finding 2: Overall, the retrained modeling tends to be more robust accuracy than that of the incremental modeling. This would affect the choice for adaptable software where the stability is more important than having greater accuracy. Finding 3: For ensemble learning algorithms, the incremental modeling has consistently better accuracy on Bagging while the retrained one shows less error on Boosting.
Modified on	10/02/2022 11:45

Name	All versus one~ an empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software
Number of Coding References	26
Number of Codes Coding	4
Coverage	12.43%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	10/02/2022 11:26
Coded Text	For RQ2, we have the following findings: Finding 4: Although the incremental modeling has statistically shorter training time than that of the retrained one (from 15% to three order of magnitude), the practical improvement may be trivial depending on the learning algorithms, e.g., for MLP, this can be practically important but may be negligible for other learning algorithms. Finding 5: Training time of incremental modeling is more robust while that of the retrained one varies depending on the subject adaptable software: more stable software system can lead to robust training time while fluctuated ones can impose varied training time. This would affect the choice for adaptable software where any single spike of high training time can cause serious consequence.
Modified on	10/02/2022 11:45

Name	All versus one~ an empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software
Number of Coding References	26
Number of Codes Coding	4
Coverage	12.43%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	10/02/2022 11:26
Coded Text	For RQ3, we obtained the following findings: Finding 6: With all the learning algorithms studied, the incremental modeling yields better accuracy and training time for 3 out of the 5 performance indicators considered. For the remaining two indicators, there is a trade-off when considering all the learning algorithms studied: the incremental modeling could exhibit shorter training time but worse accuracy. Conversely, the retrained modeling tends to impose longer training time but lead to better accuracy. This means that it is possible for the incremental modeling to achieve the best on both accuracy and training time. Finding 7: Even for the same learning algorithm, the decision of using incremental or retrained modeling can be a trade-off, see for example the DT on throughput.
Modified on	10/02/2022 11:45
Name	All versus one~ an empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software
Number of Coding References	26
Number of Codes Coding	4
Coverage	12.43%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	10/02/2022 11:26
Coded Text	For RQ4, we obtained the following findings: Finding 8: For both the incremental and retrained modeling, their errors exhibit considerably positive monotonic correlations to the number of drifts, and non-trivial negative monotonic correlations to the deviations (mRSD) of data. Relatively, the accuracy of incremental modeling worse off faster when the number of drifts increase; and improve quicker when the mRSD becomes larger. Finding 9: For the incremental modeling, its training time has strong negative monotonic correlations to the number of drifts while the correlation between the training time of retrained modeling and the number of drifts is arbitrary. There is also no clear relationship between the training time of both modeling methods and the deviations (mRSD) of data, or such a relationship is rather arbitrary.
Modified on	10/02/2022 11:45

Name	All versus one~ an empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software
Number of Coding References	26
Number of Codes Coding	4
Coverage	12.43%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	10/02/2022 11:26
Coded Text	Incremental modeling is chosen for faster training [12] [13] [14] [8] while the retrained modeling is chosen when higher accuracy is preferred [15] [16] [17] [18] [9] [19] [20] [7] [14]. The choice is a tradeoff between accuracy and training time
Modified on	10/02/2022 11:40

Name	All versus one~ an empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software
Number of Coding References	26
Number of Codes Coding	4
Coverage	12.43%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	10/02/2022 11:26
Coded Text	Lesson 1: The original belief has flaws and is inaccu- rate. Findings 1 - 3 are clear contradictions to the general belief when a learning algorithm is considered, such that the retrained modeling do not always lead to better accuracy than its incremental counterpart. Our findings have revealed some patterns when choosing the method, for example, the incremental modeling is more accurate for highly fluctuated adaptable software while the retrained one is better for stable software. The retrained modeling also exhibits more robust accuracy overall. Despite that the incremental modeling is always trained faster with better robustness than its retrained counterpart (Finding 4 and 5), which is consistent with the belief, the distinction may be practically insignificant, e.g., they differ only in milliseconds. Lesson 2: Trade-off between accuracy and training time exists, but not always. When considering all learning algorithms, tread-off is needed based on preferences, but not always. The findings (Finding 6 and 7) reveal that it is possible for the incremental modeling to perform better on both accuracy and training time; This is partially comply with the general belief. Lesson 3: Runtime fluctuation (i.e., number of drifts and deviations of data) could indeed impose non-trivial monotonic impacts on the accuracy, but limited on training time of both modeling methods. Our empirical findings (Finding 8 and 9) reveal that, in contrast to the retrained modeling, the accuracy of incremental modeling exhibits generally stronger, monotonic correlations to the number of drifts
Modified on	10/02/2022 11:45

Name	All versus one~ an empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software
Number of Coding References	26
Number of Codes Coding	4
Coverage	12.43%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	10/02/2022 11:26
Coded Text	RQ1: Does the retrained version of a given learning algorithm always make more accurate model than its incremental counterparts when modeling adaptable software? No it does not, the incremental modeling can achieve statistically better accuracy under certain learning algorithms, the adaptable software and the fluctuations of the obtained data, which is clearly contradict to what the general belief claims.
Modified on	10/02/2022 11:40

Name	All versus one~ an empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software
Number of Coding References	26
Number of Codes Coding	4
Coverage	12.43%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	10/02/2022 11:26
Coded Text	RQ2: Does the incremental version of a given learning algorithm constantly leads to faster training than its retrained counterparts when modeling adaptable software? Yes it does, as the general belief stated. However, the gain on training time may be practically trivial.
Modified on	10/02/2022 11:40

Name	All versus one~ an empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software
Number of Coding References	26
Number of Codes Coding	4
Coverage	12.43%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	10/02/2022 11:26
Coded Text	RQ3: When choosing modeling methods considering different learning algorithms, do the trade-offs between accuracy and training time for modeling performance of adaptable software always needed? Trade-off is indeed required, in which the incremental modeling could train faster but with worse accuracy. However, this is not always the case—it is possible that the incremental modeling achieves the best for both properties. Therefore, the general belief is inaccurate.
Modified on	10/02/2022 11:40
Name	All versus one~ an empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software
Number of Coding References	26
Number of Codes Coding	4
Coverage	12.43%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	10/02/2022 11:26
Coded Text	RQ4: How the modeling methods can be affected by the runtime fluctuations of the adaptable software, i.e., the number of concept drifts and the deviations in the data? The errors of both modeling methods exhibit considerably positive monotonic correlations to the number of drifts, and non-trivial negative monotonic correlations to the deviations of data. We did not observe clear correlations of their training time to the number of concept drift and data deviations in general. The only exception is the strong correlation between training time of incremental modeling and the number of concept drift.
Modified on	10/02/2022 11:41

Name	All versus one~ an empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software
Number of Coding References	26
Number of Codes Coding	4
Coverage	12.43%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	10/02/2022 11:26
Coded Text	This paper is the first to report on a comprehensive empirical study that examines both modeling methods under distinct domains of adaptable software, 5 performance indicators, 8 learning algorithms and settings, covering a total of 1,360 different conditions. Our findings challenge the general belief, which is shown to be only partially correct, and reveal some of the important, statistically significant factors that are often overlooked in existing work, providing evidence-based insights on the choice.
Modified on	10/02/2022 11:26

Name	All versus one~ an empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software
Number of Coding References	26
Number of Codes Coding	4
Coverage	12.43%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	03/02/2022 14:16
Coded Text	Lesson 3: Runtime fluctuation (i.e., number of drifts and deviations of data) could indeed impose non-trivial monotonic impacts on the accuracy, but limited on training time of both modeling methods. Our empirical findings (Finding 8 and 9) reveal that, in contrast to the retrained modeling, the accuracy of incremental modeling exhibits generally stronger, monotonic correlations to the number of drifts
Modified on	10/02/2022 11:46

Name	All versus one~ an empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software
Number of Coding References	26
Number of Codes Coding	4
Coverage	12.43%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 20:46
Coded Text	C. Analysis of the Fluctuation in Subject Software Systems To analyze the fluctuation of the adaptable software, we use the following criteria to represents the changes at runtime: Concept Drift: the concept drift [46] refers to the statistical properties of the target performance indicator, which the model is trying to predict, change over time in unforeseen ways. In general, for real-world software and data as what we studied in this work, there is no exact understanding about when the concept drift occurs. Therefore, we leverage ADWIN [47], a well-known drift detector, to measure the number of drifts in the data stream. Since we can only count the number of drifts not the extents of drifts, we apply another metric below. Relative Standard Deviations (RSD): RSD measures the extents of change in the data stream by calculating the ratio between standard deviations and mean. This includes the data about the performance indicators and the related features of the software that can be used to train a model. The normalized nature of RSD allows us to report the mean value of the RSD, denoted as mRSD, for the features and performance indicators under all cases. A larger mRSD often imply that the overall extent of concept drifts is also more significant.
Modified on	10/02/2022 11:44

Name	All versus one~ an empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software
Number of Coding References	26
Number of Codes Coding	4
Coverage	12.43%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 20:46
Coded Text	RQ4: How the modeling methods can be affected by the runtime fluctuations of the adaptable software, i.e., the number of concept drifts and the deviations in the data? The errors of both modeling methods exhibit considerably positive monotonic correlations to the number of drifts, and non-trivial negative monotonic correlations to the deviations of data. We did not observe clear correlations of their training time to the number of concept drift and data deviations in general. The only exception is the strong correlation between training time of incremental modeling and the number of concept drift.
Modified on	10/02/2022 11:41

Name	All versus one~ an empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software
Number of Coding References	26
Number of Codes Coding	4
Coverage	12.43%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	10/02/2022 11:25
Coded Text	Accuracy (Error): We measure the accuracy of the model as the adaptable software runs and as the model evolves4. At each time point t, a model is firstly updated by the data samples up to t-1 (t-2 for environment features). Then in the validation phase, the model takes the adaptable features at t and the environment features at t-1 to predict the performance at t, which is then compared with the ground truth at t. Given a scenario, we adopt Mean Absolute Error (MAE) to show the accuracy over all the intervals and repeated runs of a case, as it can additionally reflect the practicality of the error in the original scale. Suppose yk,t and ˆyk,t are the predicted and actual performance of the kth run at time t respectively; the MAE over n intervals and m repeated runs is: MAE = 1 × m× n ~m k=1 ~n t=1 |yk,t − ˆyk,t| (3) Training Time: We collected the time taken for training, and analyzed the Mean Training Time (MTT) over all the time intervals and repeated runs of a case. Robustness: By analyzing the variance of the accuracy and training time, we aim to understand the robustness of
Modified on	10/02/2022 11:45

Name	All versus one~ an empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software
Number of Coding References	26
Number of Codes Coding	4
Coverage	12.43%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	10/02/2022 11:25
Coded Text	For modeling performance at runtime, the problem that a software engineer would face is: how to update the model when using a learning algorithm2 under evolving data? Literature from the Software Engineering and Machine Learning communities take two predominate modeling methods to achieve this: (i) either completely retraining the model by learning a new data sample in conjunction with the historical ones (i.e., the retrained modeling), or (ii) simply tuning the existing model using a new data sample as it arrives (i.e., the incremental modeling). The choice between those two methods does not change the interpretation of the model, but they make fundamentally different assumptions about how a model is learned and hence they lead to different variants of a learning
Modified on	10/02/2022 11:39

Name	All versus one~ an empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software
Number of Coding References	26
Number of Codes Coding	4
Coverage	12.43%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	10/02/2022 11:25
Coded Text	Given the ever-increasing complexity of adaptable software systems and their commonly hidden internal information (e.g., software runs in the public cloud), machine learning based performance modeling has gained momentum for evaluating, understanding and predicting software performance, which facilitates better informed self-adaptations. As performance data accumulates during the run of the software, updating the performance models becomes necessary. To this end, there are two conventional modeling methods: the retrained modeling that always discard the old model and retrain a new one using all available data; or the incremental modeling that retains the existing model and tunes it using one newly arrival data sample. Generally, literature on machine learning based performance modeling for adaptable software chooses either of those methods according to a general belief, but they provide insufficient evidences or references to justify their choice.
Modified on	10/02/2022 11:25

Name	All versus one~ an empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software
Number of Coding References	26
Number of Codes Coding	4
Coverage	12.43%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	10/02/2022 11:25
Coded Text	Incremental modeling: incremental modeling follows the online learning paradigm, which is truly incremental in the sense that instead of replacing the entire model, its internal structure is tuned using the new data sample. In other words, it learns each new data sample in isolation as they arrive. The good side of incremental modeling is the likely small computation effort. However, the fact that each data sample is learned individually may ignore some joint correlations that can only be discovered when data samples are learned in conjunction with each others, which may affect the accuracy.
Modified on	10/02/2022 11:42

Name	All versus one~ an empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software
Number of Coding References	26
Number of Codes Coding	4
Coverage	12.43%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	10/02/2022 11:25
Coded Text	One fundamental to effective application of machine learn- ing in performance modeling is the data, which determines the levels of knowledge that a model can learn and generalize. However, many real world scenarios do not have sufficient data, or the available data do not adequately represent what the adaptable software is likely to behave in changing and uncertain environments. Therefore, modeling software performance at runtime with evolving data stream has been increasingly important [8] [9]. Machine learning based performance modeling at runtime has the advantage that the model can be updated using the most up-to-date data samples, which inherently improves the effectiveness of the model.
Modified on	10/02/2022 11:39

Name	All versus one~ an empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software
Number of Coding References	26
Number of Codes Coding	4
Coverage	12.43%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	10/02/2022 11:25
Coded Text	Prior Incremental Performance Modeling The other direction of effort on performance modeling assumes truly incremental modeling. For example, incremental modeling has been used in relatively simpler learning algorithms, e.g., linear regression (e.g., in [12][14]) and ARMA (e.g., in [13]), when modeling performance under changing environment of an adaptable software. The linear nature of those models make incremental modeling much more straightforward and can be tuned using Recursive Least Squares (RLS)
Modified on	10/02/2022 11:43

Name	All versus one~ an empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software
Number of Coding References	26
Number of Codes Coding	4
Coverage	12.43%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	10/02/2022 11:25
Coded Text	Prior Retrained Performance Modeling To build machine learning based performance models under evolving data stream, a large amount of research has relied on retrained modeling. Among others, Kundu et al. [15][16] have relied on Multi-Layer Perceptron (MLP) [10] and Support Vector Machine (SVM) [25] to model the performance of cloudbased and service-oriented software. Their models are built in the retrained manner, where certain amount of historical data is used to train the MLP model at design time, then at runtime, such a model is retrained whenever new data sample is available. Similarly, Siegmund et al. [20], Sieber et al. [17] and Gerostathopoulos et al. [26] use Linear Regression (LR) [27] to build the performance model at runtime, but again, the model is retrained completely instead of being tuned when significant outliers are detected or as new data is collected. Another notable effort of retrained modeling based on the Decision Tree (DT) family (e.g., M5 decision tree [28]), such as FUSION [18] and Guo et al. [19], where the performance model is discarded and rebuilt using all the available data when the adaptable software collects new information.
Modified on	10/02/2022 11:43

Name	All versus one~ an empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software
Number of Coding References	26
Number of Codes Coding	4
Coverage	12.43%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	10/02/2022 11:25
Coded Text	The Comparison Procedure and Metrics To ensure generality, we investigated a wide range of combinations on scenarios and cases, which are defined as: — Scenario: A scenario refers to each pair of learning algorithm and performance indicator of a software, e.g., using LR to predict the throughput of ASOS.
Modified on	10/02/2022 11:44

Name	All versus one~ an empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software
Number of Coding References	26
Number of Codes Coding	4
Coverage	12.43%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	10/02/2022 11:25
Coded Text	The Retrained and Incremental Modeling Modeling the performance of adaptable software via ma- chine learning often require the model to learn whenever newly observed data sample becomes available as the software runs. However, the problem that a software engineer would face is: how to update the model when using machine learning under evolving data? According to the literature from both the Software Engineering and the Machine Learning community, there are two predominate modeling methods to achieve this: Retrained modeling: retrained modeling is similar to the traditional offline learning, where the old model is discarded and a new model is retrained using whatever data that is available, i.e., the new data samples and all the historical ones. The good side of retrained modeling is that it is able to capture the interrelation between different data samples given the fact that they are always learned in conjunction with each others.
Modified on	10/02/2022 11:42

Name	AMLBID~ An auto-explained Automated Machine Learning tool for Big Industrial Data
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	An Automated Framework for Distributed Deep Learning-A Tool Demo
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	An Automated Framework for Distributed Deep Learning–A Tool Demo
Number of Coding References	5
Number of Codes Coding	1
Coverage	21.30%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	Deep learning has revolutionized the state-of-the-art results in many domains, especially healthcare [11]. Creating highperformance models requires large amounts of diverse data that covers all possible scenarios. However, centralized sharing of multi-institutional data is undesirable and challenging due to technical [9], privacy [13], and legal concerns [12]. To address these challenges, several methods for distributed deep learning have emerged, which enable individual data owners (clients) to train a shared model over their joint data without having to exchange it with one another—most notably Federated Learning (FL) [10] and Split Learning (SL) [5]. FL methods and tools are maturing past the proof-of-concept phase and have been used in several healthcare applications[8]. However, FL imposes expensive computational burdens on the clients since it requires training the entire model locally at each participating client [2]. Given these challenges, combined with the benefits of SL, we limit the scope of this demo paper to the research body of SL. To mitigate the expensive computations of FL, Split Learn- ing emerged as a new distributed deep-learning approach with the core idea of splitting the training computations between the client and the server, and exchanging the outputs of a single 2575-8411/22/$31.00 ©2022 IEEE DOI 10.1109/ICDCS54860.2022.00142 Fig. 1. Overview of splitting the model structure in Split Learning (SL) layer between the two instead of the entire model. Generally, SL splits the shared (global) model, denoted f, into two submodels (see Fig. 1): a client model, fc, and a server model, fs. The training starts with a forward propagation pass at the client, which produces a set of outputs called smashed data. Smashed data is then sent to the server to continue the forward propagation, calculate the loss value, and then update its model weights using the computed gradients. Next, the server sends the gradients of the smashed data back to the client to compute the local gradients and update its model. When there exists more than one client, the original SL work suggested training one client with the server at a time in a sequential manner. Because the global model is split, this approach can reduce the computational and communication burdens on the clients [14] and opens the horizon for advanced distributed settings, such as training on vertically-partitioned data [1]. However, the sequential training paradigm makes SL impractical for realworld applications, which has led to new versions that support parallel training at the clients, namely SplitFed [7]. While the existing body of SL research have demonstrated significant results across many tasks, this research domain still largely ignores the design and implementation of proper tool support leading to a wider gap between the methods and their implementation—especially tools that cater to deep learning practitioners with non-technical background, such as physicians. Implementing distributed learning is challenging and requires interdisciplinary skills (data management, model training, networking, etc.), but is currently left to the end user. 1302 Authorized licensed use limited to: UNIVERSITY OF OSLO. Downloaded on February 15,2023 at 13:41:08 UTC from IEEE Xplore. Restrictions apply.
Modified on	16/02/2023 12:04

Name	An Automated Framework for Distributed Deep Learning–A Tool Demo
Number of Coding References	5
Number of Codes Coding	1
Coverage	21.30%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	Split learning (SL) is a distributed deep-learning approach that enables individual data owners to train a shared model over their joint data without exchanging it with one another. SL has been the subject of much research in recent years, leading to the development of several versions for facilitating distributed learning. However, the majority of this work mainly focuses on optimizing the training process while largely ignoring the design and implementation of practical tool support. To fill this gap, we present our automated software framework for training deep neural networks from decentralized data based on our extended version of SL, termed Blind Learning. Specifically, we shed light on the underlying optimization algorithm, explain the design and implementation details of our framework, and present our preliminary evaluation results. We demonstrate that Blind Learning is 65% more computationally efficient than SL and can produce better performing models. Moreover, we show that running the same job in our framework is at least 4.5× faster than PySyft. Our goal is to spur the development of proper tool support for distributed deep learning.
Modified on	16/02/2023 12:03

Name	An Automated Framework for Distributed Deep Learning–A Tool Demo
Number of Coding References	5
Number of Codes Coding	1
Coverage	21.30%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	System implementation: The Router is a cloud service for managing the organizations' accounts and their jobs. It also provides a web application to list and explore metadata about existing assets, which can be used for collaborative training of deep learning models. The web application interface is implemented using React while its back-end and APIs are implemented in Django. The authentication and authorization at the Router are based on the JSON Web Token (JWT) and use Secure Socket Layer (SSL) to verify the authenticity of all parties in the ecosystem and to encrypt the communications between them. The PostgreSQL database is used to store metadata about all existing users and their available assets. The Access Point is implemented as a docker container encapsulating the implementation of the BL protocol. APs of different organizations interact with each other using secure communication channels directly and never pass throught the Router. We utilize the SQLite database to store the organizations assets within the AP. The SDK is built in Python and provides a complete scripting control of the overall system. It is installed on the end user's device (e.g., a data scientist's workstation) to manage the organization's assets and to run training jobs. It includes a Python library and command-line utilities to interface with the rest of the framework.
Modified on	16/02/2023 12:05

Name	An Automated Framework for Distributed Deep Learning–A Tool Demo
Number of Coding References	5
Number of Codes Coding	1
Coverage	21.30%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	THE BLIND LEARNING FRAMEWORK Training workflow: We introduce in this section the overall framework architecture, which enables efficient and automated tool support for BL, made of three major components: Router, Access Point (AP), and the Software Development Kit (SDK), see Fig. 2 (the numbered steps in the figure match the numbered steps in the text below). But first, we explain the overall workflow for a training job. We assume two types of users: a data owner (client) and a data user (server), each with an AP installed on their organizations' machines. An AP is a docker container (Blue boxes in Fig. 2) that includes our toolset and enables organizations to communicate with each other, host their assets (data and models), run our protocols, and control access permissions– all enabled via a set of high-level APIs defined in our SDK. To run BL, the data user explores available datasets via a web application located at the Router 1◯, which is the primary management unit of our ecosystem. The Router is responsible for coordinating the users, jobs, permissions, and digital rights. It also provides a web application that allows different users to list and explore assets. The assets are listed by their owners when they position them inside their APs 2◯. Positioning an asset allows our protocols to operate within the AP without sending any raw data outside the organizations infrastructure. After identifying proper datasets, the data user can use our APIs, defined in the SDK, to configure and run a training job 3◯. Data owners will receive access requests, which they can review and accept/decline. Then, the training takes place between the data user (server) and data providers (clients) directly without passing through the Router. Fig. 3 illustrates the usage of the BL API to run an actual training job by a data user over two datasets, referenced in Lines 3 and 4 using their public IDs, obtained from the web application. The model is defined in Lines 6-16 while the training settings, e.g., number of epochs, test size, batch size, are defined in the bl.create_job dictionary (Lines 20 to 30). Refer to [3] and the demo site for more examples.
Modified on	16/02/2023 12:04

Name	An Automated Framework for Distributed Deep Learning–A Tool Demo
Number of Coding References	5
Number of Codes Coding	1
Coverage	21.30%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	To the best of our knowledge, only two tools have shed some light on this issue: PySyft 1 and FedML [6]. FedML was created for research purposes, focuses mainly on FL, and limits the user to a handful of models and datasets. In contrast, PySyft allows actual distributed training over the public Internet, but it still requires significant manual intervention from the end user. It also introduces significant communication overhead; for example, training the same model requires 17 minutes in PySyft versus less than 3.6 minutes in our framework. Moreover, both PySyft and FedML mainly focus on FL and provide support for the sequential version of SL only, which is impractical for real-world applications. Contributions. To this end, we present an automated soft- ware framework for distributed deep learning with the goal of facilitating and accelerating collaborative training. Specifically, (1) we shed light on the underlying learning approach, termed Blind Learning (BL), (2) present the design and implementation details of a complete framework for distributed learning, and (3) present preliminary evaluation results demonstrating the efficiency and accuracy of our framework.
Modified on	16/02/2023 12:04

Name	An Automated Framework for Distributed Deep Learning-A Tool Demo (2)
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	An Automated Framework for Distributed Deep Learning-A Tool Demo (2) Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	An Automated Framework for Distributed Deep Learning-A Tool Demo Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	An Automated Framework for Distributed Deep Learning–A Tool Demo Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	An Empirical Study of Refactorings and Technical Debt in Machine Learning Systems
Number of Coding References	16
Number of Codes Coding	2
Coverage	8.84%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	
Modified on	10/02/2022 11:14

Name	An Empirical Study of Refactorings and Technical Debt in Machine Learning Systems
Number of Coding References	16
Number of Codes Coding	2
Coverage	8.84%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	
Modified on	10/02/2022 11:15

Name	An Empirical Study of Refactorings and Technical Debt in Machine Learning Systems
Number of Coding References	16
Number of Codes Coding	2
Coverage	8.84%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	
Modified on	10/02/2022 11:16

Name	An Empirical Study of Refactorings and Technical Debt in Machine Learning Systems
Number of Coding References	16
Number of Codes Coding	2
Coverage	8.84%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	
Modified on	10/02/2022 11:16

Name	An Empirical Study of Refactorings and Technical Debt in Machine Learning Systems
Number of Coding References	16
Number of Codes Coding	2
Coverage	8.84%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	
Modified on	10/02/2022 11:16

Name	An Empirical Study of Refactorings and Technical Debt in Machine Learning Systems
Number of Coding References	16
Number of Codes Coding	2
Coverage	8.84%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	
Modified on	10/02/2022 11:17

Name	An Empirical Study of Refactorings and Technical Debt in Machine Learning Systems
Number of Coding References	16
Number of Codes Coding	2
Coverage	8.84%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	1) Automated refactorings especially designed for migrating “linear” ML algorithm and configuration code to use inheritance constructs may be advantageous in avoiding code duplication. 2) More ML-specific refactoring tool-support may encourage more refactoring of model and configuration code, potentially reducing technical debt. 3) More automated client-side matrix calculation refactorings may replace manual model code performance enhancements. Fig. 6: Recommendations.
Modified on	10/02/2022 11:17

Name	An Empirical Study of Refactorings and Technical Debt in Machine Learning Systems
Number of Coding References	16
Number of Codes Coding	2
Coverage	8.84%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	A. Code Duplication in Configuration & Model Code With duplicate code elimination being one of the top overall and crosscutting refactoring categories (finding 2), as well as the top refactoring performed on ML-related code (finding 7), ML systems seem to exhibit a significant amount of code duplication, particularly in configuration and model code regions (finding 13). Feasible explanations include (i) data scientists—potentially untrained as software engineers and thus not fully aware of advanced modularization techniques—may be responsible for model code, (ii) model code is highly-configurable—containing a substantial number of different yet related hyperparameters—which are configured in similar ways, and (iii) many different ML algorithms share a significant amount of commonality, giving way to code duplication.
Modified on	10/02/2022 11:17

Name	An Empirical Study of Refactorings and Technical Debt in Machine Learning Systems
Number of Coding References	16
Number of Codes Coding	2
Coverage	8.84%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	B. Combating Code Duplication Debt in ML Systems We identified the two ML system areas that exhibit the most duplication, i.e., configuration and model code. Amershi et al. [8] also note issues with model code reuse. Fortunately, per finding 5, inheritance was a centrally used technique in eliminating code duplication, particularly with algorithm variations, and finding 14 shows that it was especially useful to reduce duplicate configuration code.
Modified on	10/02/2022 11:17

Name	An Empirical Study of Refactorings and Technical Debt in Machine Learning Systems
Number of Coding References	16
Number of Codes Coding	2
Coverage	8.84%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	C. Generic vs. ML-specific Refactorings Generic refactorings (94.03%) vastly outnumbered those of our new ML-specific refactorings (5.97%). A feasible explanation is (i) model code is among the smallest ML subsystems [1]; thus, we would expect less ML-specific refactorings, (ii) data scientists—potentially not versed in refactoring—may be responsible for ML-related code maintenance and evolution, and (iii) a lack of ML-specific automated refactorings may deter developers as they must refactor manually, leading to recommendation 2, fig. 6. The lack of ML-specific refactoring occurrences—along with finding 11—does not necessarily indicate that technical debt is not present; it may be that it is simply not being addressed. Also, good solutions for these problems may not yet exist [1].
Modified on	10/02/2022 11:18

Name	An Empirical Study of Refactorings and Technical Debt in Machine Learning Systems
Number of Coding References	16
Number of Codes Coding	2
Coverage	8.84%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	In this section, we mainly summarize the study results using data—noting trends, exceptions, and unexpected outcomes. §IV, on the other hand, consolidates and comments on the main findings and connects the different parts of the results. Related discussion in §IV is referenced where appropriate.
Modified on	10/02/2022 11:14

Name	An Empirical Study of Refactorings and Technical Debt in Machine Learning Systems
Number of Coding References	16
Number of Codes Coding	2
Coverage	8.84%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	ML-related Code Performance Model code needs to be fast; thus, it is not surprising that 66.67% of performance enhancements occurred in MLrelated code (finding 7). We came across several refactorings that converted reference types to primitives for performance reasons. Our findings coincide with that of Kim [11] and Zhang et al. [18], i.e., performance is essential yet challenging in ML systems. Additional client-side tool-support focused on improving matrix calculations (e.g., [5]) may alleviate developers from making manual performance enhancements, leading to recommendation 3, fig. 6.
Modified on	10/02/2022 11:18

Name	An Empirical Study of Refactorings and Technical Debt in Machine Learning Systems
Number of Coding References	16
Number of Codes Coding	2
Coverage	8.84%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	New ML-specific refactorings & technical debt categories We introduce 14 and 7 new ML-specific refactorings and technical debt categories, respectively. Recommendations, best practices, & anti-patterns We propose preliminary recommendations, best practices, and anti-patterns for long-lasting ML system evolution from our statistical results, as well as an in-depth analysis. Complete results of our study are available in our dataset [17].
Modified on	10/02/2022 11:14

Name	An Empirical Study of Refactorings and Technical Debt in Machine Learning Systems
Number of Coding References	16
Number of Codes Coding	2
Coverage	8.84%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	Our contributions can be summarized as follows: Refactoring hierarchical taxonomy From 327 patches of 26 projects manually examined, we build a rich hierarchical, crosscutting taxonomy of common generic and MLspecific refactorings, whether they occur in ML-related code—code specific to ML-related tasks (e.g., classifiers, feature extraction, algorithm parameters)—and the MLspecific technical debt they address.
Modified on	10/02/2022 11:14

Name	An Empirical Study of Refactorings and Technical Debt in Machine Learning Systems
Number of Coding References	16
Number of Codes Coding	2
Coverage	8.84%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	Our study involved analyzing 26 projects, consisting of 4.2 MLOC, along with 327 manually examined code patches. Refactorings were taxonomized, labeled as being performed in ML code or not, and related to the ML-specific debt they alleviated. Our study indicates that (i) duplicate code elimination—largely performed by introducing inheritance—was a major crosscutting theme in ML system refactoring that mainly involved ML configuration and model code, which was also the most refactored code, (ii) subtle variation of different yet related ML algorithms and their configurations were a major force driving code duplication, (iii) code generalization, reusability, and external interoperability—essential SE concepts—were among the least performed refactorings, and (iv) configuration, duplicate model code, and plain-old-data types were the most addressed technical debt.
Modified on	10/02/2022 11:13

Name	An Empirical Study of Refactorings and Technical Debt in Machine Learning Systems
Number of Coding References	16
Number of Codes Coding	2
Coverage	8.84%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	03/02/2022 14:16
Coded Text	Machine Learning (ML), including Deep Learning (DL), systems, i.e., those with ML capabilities, are pervasive in today's data-driven society. Such systems are complex; they are comprised of ML models and many subsystems that support learning processes. As with other complex systems, ML systems are prone to classic technical debt issues, especially when such systems are long-lived, but they also exhibit debt specific to these systems. Unfortunately, there is a gap of knowledge in how ML systems actually evolve and are maintained.
Modified on	10/02/2022 11:12
Name	An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions
Number of Coding References	31
Number of Codes Coding	4
Coverage	10.54%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	
Modified on	10/02/2022 11:04

Name	An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions
Number of Coding References	31
Number of Codes Coding	4
Coverage	10.54%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	
Modified on	10/02/2022 11:06

Name	An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions
Number of Coding References	31
Number of Codes Coding	4
Coverage	10.54%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	Concept drift exists in the operation data, which can be explained by the fact that the relationship between the variables in the operation data evolves over time. Practitioners and researchers should proactively detect and address the problem ofconcept drift in their AIOps solutions.
Modified on	10/02/2022 11:07

Name	An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions
Number of Coding References	31
Number of Codes Coding	4
Coverage	10.54%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	Concept drift exists in the operation data. Figure 12 describes the concept drift in different time periods of the studied datasets. We observe that many of the time periods show a concept drift from its previous period. For example, on the Google dataset, the RF, NN, and CART models indicate that 70% (19 out of 27) time periods exhibit concept drift, and the CART and SVM indicate 18 and 17 periods with concept drift, respectively. On the Backblaze dataset, the CART model shows that all 11 time periods have concept drift from the previous time periods, while the other four models (i.e., RF, NN, RGF, SVM
Modified on	10/02/2022 11:07

Name	An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions
Number of Coding References	31
Number of Codes Coding	4
Coverage	10.54%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	Data leakage could exist in AIOps solutions that use a random splitting of training and validation datasets, as random splitting achieves a higher model performance than the baseline splitting strategy that leverages all the available past data. We observe that, overall, models that are trained and evaluated on a random splitting have a higher performance than the baseline splitting, which indicates that the random splitting could lead to over-estimation ofmodel performance than the baseline splitting
Modified on	10/02/2022 11:05

Name	An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions
Number of Coding References	31
Number of Codes Coding	4
Coverage	10.54%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	Due to the existence of concept drift, AIOps models should be updated periodically, as periodically updated models outperform stationary models. In general, increasing the frequency of updating AIOps models can lead to better performance while increasing the modeling cost. However, the performance benefit and modeling cost of increasing the update frequency show very different trends across models and datasets.
Modified on	10/02/2022 11:08

Name	An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions
Number of Coding References	31
Number of Codes Coding	4
Coverage	10.54%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	Handling data leakage. Prior studies [5, 24, 57, 66] usually randomly split the dataset into a training set and a validation set. For example, El-Sayed et al. [24] randomly split the whole Google cluster trace dataset [88] into 70% training data and 30% validation data. Botezatu et al. [5]and Mahdisoltani et al. [57] randomly split the Backblaze disk stats dataset into 80% training data and 20% validation data, and 75% training data and 25% validation data, respectively. In comparison, some prior studies use a time-based approach to split training and validation data, which ensures that the training data always occurs before the validation data [49, 51, 71, 89]. In this work, we analyze the existence of data leakage in the studied operation datasets (RQ1). Then, we evaluate the impact of using a time-based splitting (i.e., considering the temporal order in the data) instead of random splitting on model evaluation (RQ2).
Modified on	10/02/2022 11:03

Name	An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions
Number of Coding References	31
Number of Codes Coding	4
Coverage	10.54%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	Handling imbalanced data. Operation data is often very imbalanced [5, 24, 49, 51, 57]. Therefore, AIOps solutions usually apply data rebalancing techniques (e.g., over-sampling, undersampling, SMOTE, ROSE) to make the modeled classes more balanced and produce more accurate models [44, 80]. For example, Botezatu et al. [5] and Mahdisoltani et al. [57] use under-sampling approaches (i.e., randomly reducing the samples of the majority class) to balance the samples of failed disks and normal disks in their tasks of disk failure prediction. El-Sayed et al. [24]use an over-sampling approach (i.e., making random duplication ofthe minority class) to balance the samples offailed jobs and normally terminated jobs in their tasks ofjob failure prediction. Xu et al. [89] and Chen et al. [16] use the SMOTE over-sampling approach [13] to balance their classes in the tasks of disk failure prediction and service outage prediction, respectively. This work does not explore the impact of data rebalancing techniques on AIOps solutions, as data rebalancing has been extensively discussed in prior work (e.g., References [49, 51, 80]). Instead, we use an under-sampling approach to balance the classes in both our studied datasets, as done in Botezatu et al. [5] and Mahdisoltani et al. [57].
Modified on	10/02/2022 11:02
Name	An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions
Number of Coding References	31
Number of Codes Coding	4
Coverage	10.54%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	Prior work proposed many AIOps solutions to address various problems in the operations of largescale software and systems, such as incident prediction [5, 16, 24, 49, 51, 57, 70, 89], anomaly detection [31, 50], ticket management [90, 91], issue diagnosis [55], and self healing [18, 19, 52, 53]. For example, Lin et al. [51]and Li et al.[49] leverage temporal data (e.g., CPU and memory utilization metrics, alerts), spatial data (e.g., rack locations), and config data (e.g., memory size) to predict node failures in large-scale cloud computing platforms. El-Sayed et al. [24] and Rosa et al. [70] learned from the trace data to predict job failures in the Google cloud computing platform. Botezatu et al. [5], Mahdisoltani et al. [57], andXuetal. [89] leveraged disk-level sensor data and systemlevel events to predict disk failures in operations of large-scale cloud platforms. As illustrated in Figure 1, ML modeling, in particular, supervised learning, in the context of AIOps usually faces three data splitting-related challenges: the imbalanced data challenge in model training, the data leakage challenge in model training and evaluation, and the concept drift challenge in model maintenance. Table 1 and Table 2 list prior AIOps work that leverages supervised learning and unsupervised learning techniques, respectively. For the works using supervised learning, we summarize how they handle the three challenges in different ML modeling phases. Below, we discuss prior AIOps solutions that rely extensively on supervised ML models.
Modified on	15/02/2022 11:43

Name	An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions
Number of Coding References	31
Number of Codes Coding	4
Coverage	10.54%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	Random splitting of training and validation datasets has higher performance than time-based splitting. We observe
Modified on	10/02/2022 11:05

Name	An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions
Number of Coding References	31
Number of Codes Coding	4
Coverage	10.54%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	Randomly splitting operation data for the training and validation of a model may cause data leakage problems in AIOps solutions that impact a model’s realistic evaluation.
Modified on	15/02/2022 11:41

Name	An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions
Number of Coding References	31
Number of Codes Coding	4
Coverage	10.54%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	The contributions of this article are: (1) This is the first work that assesses the performance impact ofvarious data splitting decisions on AIOps solutions. The findings and the proposed techniques in this article can be useful to machine learning engineers or software engineering researchers interested in improving the quality and maintainability of AIOps solutions. (2) Our results show that problems such as data leakage (caused by decisions during model training and evaluation) and concept drift (caused by the evolution of data) can easily appear in AIOps solutions ifnot being careful while deciding on various data splitting strategies. Such problems may severely impact the performance ofAIOps solutions while being deployed in the field. (3) To mitigate the risks of the various problems arising from data splitting decisions, we also proposed suggested techniques and demonstrated their effectiveness in our case studies. In particular, we observe that using a time-based splitting of training and validation datasets can reduce data leakage and provide a more reliable evaluation. We also observe that periodically updating AIOps models can help mitigate the impact of concept drift, while the frequency of model updating should be cautiously considered.
Modified on	10/02/2022 11:00

Name	An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions
Number of Coding References	31
Number of Codes Coding	4
Coverage	10.54%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	The different model evaluation scenarios are illustrated in Figure 3 and detailed below. • Baseline splitting:We use a baseline splitting strategy that trains a model using all the past data before predicting each data sample, which intuitively should yield better performance than other valid splitting strategies. Prior work [48] uses a similar approach to evaluate the performance of predicting log changes. Specifically, we first randomly choose N samples as the testing data. For each testing sample, we build a model with all available samples (i.e., samples that have finished before the testing sample) and test the model on the current testing sample. We then combine the prediction results of all the testing samples to calculate the performance. For the Google cluster trace data, we set N as 15,000; for the Backblaze disk stats data, we set N as 150,000. We choose these values to ensure that we have enough samples from the minority classes. • Random splitting: In this scenario, we first randomly split the data into a training set and a validation set, then we train a model using the training set and evaluate the model on the validation set. We consider five training/validation split ratios that range from 50%/50% to 90%/10%. • Time-based splitting: In this scenario, we split the data into the training set and testing set based on the temporal order, then we train a model using the training set and evaluate the model on the validation set. We also consider the five training/validation split ratios that range from 50%/50% to 90%/10%.
Modified on	10/02/2022 11:05

Name	An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions
Number of Coding References	31
Number of Codes Coding	4
Coverage	10.54%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	The time-based splitting strategy provides a more realistic evaluation of an AIOps model when it is retrained on all the available data and applied to future unseen data. The estimated performance of a model (obtained on the validation dataset) is closer to the
Modified on	10/02/2022 11:07

Name	An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions
Number of Coding References	31
Number of Codes Coding	4
Coverage	10.54%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	The time-based splitting strategy shows a more consistent performance between the validation and the unseen testing datasets compared to random splitting. Figure 7 and Figure 8 show the performance of the models that are trained and evaluated using different data splitting strategies and splitting ratios. Under a random splitting, the evaluatedmodel performance (i.e., on the validation dataset) is less consistent with the performance of the same model on the unseen testing dataset;
Modified on	10/02/2022 11:07

Name	An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions
Number of Coding References	31
Number of Codes Coding	4
Coverage	10.54%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	While prior work relies on the random splitting of training and validation sets, their reported performance on the validation set could be higher than on the unseen testing data, which is a biased evaluation. The bias is particularly larger when specific models (e.g., CART) or a very large splitting ratio (e.g., 90%/10%) is used. On the contrary, the timebased splitting is more appropriate for AIOps model evaluation, since it produces more consistent performance between the validation and unseen testing data.
Modified on	10/02/2022 11:07

Name	An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions
Number of Coding References	31
Number of Codes Coding	4
Coverage	10.54%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 20:46
Coded Text	Concept drift exists in the operation data, which can be explained by the fact that the relationship between the variables in the operation data evolves over time. Practitioners and researchers should proactively detect and address the problem ofconcept drift in their AIOps solutions.
Modified on	10/02/2022 11:07

Name	An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions
Number of Coding References	31
Number of Codes Coding	4
Coverage	10.54%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 20:46
Coded Text	Handling concept drift. Existing AIOps studies usually train a static model regardless of po- tential concept drift [5, 16, 24, 57, 71, 91] without respecting that the operation data is constantly evolving [17, 49, 51]. However, concept drift may lead to the obsolescence of such static models trained on previous data. To mitigate the impact of concept drift, other prior works suggest that AIOps models need to be retrained periodically to ensure that the models are not outdated
Modified on	10/02/2022 11:03

Name	An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions
Number of Coding References	31
Number of Codes Coding	4
Coverage	10.54%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 20:46
Coded Text	In this work, we first analyze the existence of concept drift in the studied operation dataset (RQ3), then we evaluate the impact ofperiodically updating amodel instead ofusing a static model on the model performance, and how the model update frequency might impact the performance and cost of AIOps models (RQ4).
Modified on	10/02/2022 11:03

Name	An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions
Number of Coding References	31
Number of Codes Coding	4
Coverage	10.54%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 20:46
Coded Text	Increasing the frequency ofupdating the AIOps models can improve the performance, while the improvement shows difference across models and datasets. Figure 16 shows the overall performance ofthe models (in terms ofAUC)whenwe vary the number oftime periods (i.e., N). Other performance metrics show a similar trend. In most cases, increasing the model update frequency can gradually improve model performance. For example, the AUC of the CART model on the Backblaze dataset increases by 3.5% (i.e., from 0.85 to 0.88) when we increase the number of time periods from 4 to 24 (the AUC of the stationary model, i.e., when N = 2, is 0.84). However, in some cases, for example, when updating the SVM model on the Backblaze dataset, we did not observe any performance improvement. We infer that some models (e.g., RGF) can not learn the evolving patterns in the datasets, thus are less sensitive to the update frequency. Increasing themodel update frequency increase the overall cost ofAIOps models; how- ever, the cost increase varies significantly across models and datasets. Figure 17 shows the
Modified on	24/02/2022 09:08

Name	An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions
Number of Coding References	31
Number of Codes Coding	4
Coverage	10.54%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 20:46
Coded Text	Periodically updated AIOps models provide better performance than the stationary models, which suggest modelers update their AIOps models periodically. As shown in Figure 15(a) and Figure 15(b), periodically updating the models achieve a better performance in terms of the evaluated metrics than using a stationary model, and overall the difference becomes bigger when the distance between the training periods of the stationary model and the testing period becomes larger. We observe that the stationary models and the periodically updated models have a bigger difference on the Google dataset than on the Backblaze dataset, which may be explained by the fact that the Google dataset has a more severe concept drift issue (as discussed in Section 6).
Modified on	24/02/2022 09:08

Name	An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions
Number of Coding References	31
Number of Codes Coding	4
Coverage	10.54%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	31/01/2022 14:44
Coded Text	
Modified on	10/02/2022 10:59

Name	An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions
Number of Coding References	31
Number of Codes Coding	4
Coverage	10.54%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	31/01/2022 14:44

Coded Text

• Imbalanced data: Operation data is often very imbalanced [5, 24, 49, 51, 57], which challenges AIOps modeling, as the models tend to make a more accurate prediction on the majority class while performing poorly on the minority class [44, 80, 89]. Such a challenge requires the application of data rebalancing techniques (e.g., over-sampling, under-sampling, SMOTE [13], ROSE [54]) to make the modeled classes more balanced (i.e., splitting the data ofdifferent classes to achieve a better balance between the classes) [44, 80] or using cost-sensitive models [1, 15, 35].
• Data leakage: Prior studies (e.g., References [5, 24, 57]) in AIOps randomly split operation data into training and validation data. However, such a splitting strategy may risk data leakage, i.e., leak information in the validation data that should not be available for model training into the training data, which may introduce bias and result inmisleading evaluation results [39, 40, 65, 72]. For example, in a recent Kaggle competition [74], the leakage of the future information into the training features cause the model to make unrealistic good predictions that could not reflect the actual model performance in a practical setting.
• Concept drift: Over time, the distribution of the operation data and the relationship between the variables in the data may be constantly evolving [17, 49, 51] (a.k.a. concept drift [64, 84–86]). Concept drift may lead to obsolescence of the models trained on historical data, i.e., a model trained on outdated data may perform poorly on new data.

Modified on	10/02/2022 10:59

Name	An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions
Number of Coding References	31
Number of Codes Coding	4
Coverage	10.54%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	31/01/2022 14:44
Coded Text	Despite the breakthroughs in ML models and their applications in AIOps, there are still challenges preventing the integration of such ML models into software systems [41], such as the challenges in model evaluation and model evolution [3, 73]. One of the main reasons is that ML experts usually focus on tuning the MLmodel performance instead ofmaintaining model behavior after deploying in the field [41]. Hence, software engineering for machine learning has become an emerging topic that aims to manage the lifecycle ofmachine learning models (i.e., training, testing, deploying, evolving, etc.) [3, 41, 63]. Within the lifecycle of ML models, making appropriate decisions for data splitting (e.g., splitting data into training and validation sets) is particularly challenging, even for ML experts [38, 63]. For example, ML experts highlight the importance of data splitting in ML modeling [63] and advocate the introduction of engineering processes for data splitting [38]. In particular, in the context ofAIOps, ML modeling faces three data splitting (DS)-related challenges during the process of developing AIOps solutions, as shown in Figure 1.
Modified on	10/02/2022 10:58

Name	An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions
Number of Coding References	31
Number of Codes Coding	4
Coverage	10.54%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	31/01/2022 14:44
Coded Text	The Challenge of Concept Drift: In machine learning and data mining, the distribution of the data and the relationship between the variables may evolve over time, which is known as concept drift [64, 84–86]. Concept drift may lead to obsolescence of models trained on previous data and negatively impact the performance. To mitigate the impact of concept drift, prior works propose approaches for detecting concept drift [26, 30, 64, 87] and handling concept drift [9, 12, 21, 28, 32, 60, 61, 77, 85]. For example, Nishida et al. [64] propose a concept drift detection method using statistical testing. It assumes that the prediction accuracy on the data from a recent time window would be equal to the overall accuracy if the target concept is stationary, and a significant decrease in the recent accuracy suggests a concept drift. When there is concept drift, aside from retraining a model from scratch, online learning updates the current model using the most recent data incrementally. Such model process input examples one-by-one and update
Modified on	10/02/2022 11:01

Name	An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions
Number of Coding References	31
Number of Codes Coding	4
Coverage	10.54%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	31/01/2022 14:44
Coded Text	The Challenge of Data Leakage: Data leakage is the introduction of information in the training data that should not be available for model training and can lead to the bias of model evaluation [39, 40, 65, 72]. The creation of such unexpected additional information in the training data would enable the models to use the future data to predict the past data [49, 51, 89], and therefore cause it to make unrealistically good predictions that could not reflect the practical performance. Leakage is a pervasive challenge in applied machine learning, causing models to over-represent their generalization error and often rendering them useless in the real world. For example, leakage of the future information into the training features are reported in many Kaggle competitions, including a recent one in a prostate cancer dataset [74]. Prior works [39, 72] suggest that when there are risks of such data leakage, time-based splitting of training and validating data splitting (i.e., splitting the data based on their time sequence) should be used over a random-based splitting strategy. In this work, we study the existence of data leakage in the context of AIOps, which has not been explored before. We also evaluate the impact of different splitting strategies (e.g., time-based splitting) on data leakage.
Modified on	10/02/2022 11:01

Name	An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions
Number of Coding References	31
Number of Codes Coding	4
Coverage	10.54%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	31/01/2022 14:44
Coded Text	The Challenge of Imbalanced Data: Imbalanced data is a common problem in the ma- chine learning community. It arises when one of the classes is severely underrepresented in the dataset. [32]. Imbalanced data could cause models to focus on the majority class and ignore the rare events, which heavily compromises the process of learning [44]. The machine learning community usually addresses the issue in two ways. One way is to apply data rebalancing techniques, most simply, oversampling the minority class or under-sampling the majority class. There are also approaches that blend the two sampling strategy like SMOTE (Synthetic Minority Over-sampling TEchnique) [13], and approaches that combine oversampling with the generation of artificial data like ROSE (Random OverSampling Examples) [59]. As a result, the modeled classes are more balanced and may produce more predictive models. Other than resampling techniques that balance the sample classes, researchers also design ML models specially optimized for the imbalanced data issue by assigning distinct costs to the training samples. For example, Arya et al. [35] propose a cost-sensitive support vector machine algorithm that provides superior generalization performance compared to conventional SVM on imbalanced data; deep learning approaches can also tackle the imbalanced data problem with a weighted backpropagation [15] or a weighed form of categorical cross-entropy [1]. Besides, updatable classification algorithms can also be a viable approach in handling imbalanced data. Ming et al. [78]report that updatable classification algorithms, which update the training set incrementally to take advantage of the feedback from each run, improve the precision under some circumstances when handling imbalanced data.
Modified on	10/02/2022 11:01

Name	An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions
Number of Coding References	31
Number of Codes Coding	4
Coverage	10.54%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 20:46
Coded Text	• Concept drift: Over time, the distribution of the operation data and the relationship between the variables in the data may be constantly evolving [17, 49, 51] (a.k.a. concept drift [64, 84–86]). Concept drift may lead to obsolescence of the models trained on historical data, i.e., a model trained on outdated data may perform poorly on new data.
Modified on	10/02/2022 10:59

Name	An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions
Number of Coding References	31
Number of Codes Coding	4
Coverage	10.54%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 20:46
Coded Text	In machine learning and data mining, concept drift means the change in the relationships between the variables over time [84–86, 92]. Concept drift may negatively impact the performance of a model trained from the past data when applied to the new data [84–86]. Therefore, in this RQ, we analyze the studied datasets to understand whether concept drift issues exist in the context of AIOps. In particular, we leverage statistical analysis to measure the existence of concept drift in the studied datasets. 6.2 Approach Prior work [42, 45, 64] assumes that, given a stationary data distribution (i.e., no concept drift), a model trained on a previous time period would achieve a prediction performance (when evaluated on the next time period) that has no statistical difference from the prediction performance on the training period.We follow the same hypothesis to measure the concept drift in our studied datasets. If a model trained from the previous data shows a statistically significant performance difference on the new data, then a concept drift exists. In our study, we use the natural time intervals (i.e., one-day periods for the Google dataset and one-month periods for the Backblaze dataset) to split the data into different time periods. We choose such a time window size as prior works have applied similar update strategies. For example, Lin et al. [51] update their model deployed in a production cloud service system with data from a one-month window. Similarly, Li et al. [49] consider retraining their model periodically and they also apply a one-month window. Also, Xu et al. [89] perform a daily model update with the data in a 90-day sliding window. We conduct our experiment as follows: (1) For each time period, we train a model using the data from that time period and test the same model using the next time period's data to measure the prediction error rate. (2) We then compute the statistical difference between the model's prediction error rate on the training time period and its prediction error rate on the testing time period, similar to prior work [45, 64]. However, these studies [45, 64] do not explicitly explain how they measure the prediction error rate on the training time period. Thus, we follow prior work [42, 86] and use 10-fold cross-validation on the training time period to measure the prediction error rate on the training time period. (3) Similar to prior work [45, 64], we use a two-proportion Z-test to compute the statistical difference between the model's prediction error rates in the training and testing time periods, which is described as follows: Z = ■ pˆ(1 − ˆp) (ˆp2 − ˆp1) − 0 □ 1 n1 + 1 n2 ■ , (1) where pˆ1 is the prediction error rate in the training time period, pˆ2 is the prediction error rate in the testing time period, pˆ is the overall prediction error rate, and n1 and n2 are the number of samples in the training time period and the testing time period, respectively. (4) We then determine the significance level (i.e., p-value) from the Z-test. When the p-value is less than 0.05, we reject the null hypothesis (i.e., pˆ1 − ˆp2 = 0) and consider the alternative that there is a concept drift between the two time periods. (5) To understand the magnitude of the concept drift between two adjacent time periods, we also calculate the relative difference between the prediction error rates in the training and testing time periods ((ˆp2 − ˆp1)/ ˆp1).
Modified on	24/02/2022 09:06

Name	An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions
Number of Coding References	31
Number of Codes Coding	4
Coverage	10.54%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 20:46
Coded Text	The Challenge of Concept Drift: In machine learning and data mining, the distribution of the data and the relationship between the variables may evolve over time, which is known as concept drift [64, 84–86]. Concept drift may lead to obsolescence of models trained on previous data and negatively impact the performance. To mitigate the impact of concept drift, prior works propose approaches for detecting concept drift [26, 30, 64, 87] and handling concept drift [9, 12, 21, 28, 32, 60, 61, 77, 85]. For example, Nishida et al. [64] propose a concept drift detection method using statistical testing. It assumes that the prediction accuracy on the data from a recent time window would be equal to the overall accuracy if the target concept is stationary, and a significant decrease in the recent accuracy suggests a concept drift. When there is concept drift, aside from retraining a model from scratch, online learning updates the current model using the most recent data incrementally. Such model process input examples one-by-one and update
Modified on	10/02/2022 11:01

Name	An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions
Number of Coding References	31
Number of Codes Coding	4
Coverage	10.54%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 20:46
Coded Text	the model after receiving each example [28]. For example, CVFDT [33] is a decision tree model that incrementally updates itself when new data becomes available and can adapt to the drifting concept. Time-based ensembles combine individual base models trained on data from small time peri- ods. The intuition is that the base models trained from such small time periods can better capture the relationship between the variables, as the concept drift in a smaller period is relatively small. For example, Steet and Kim propose the Streaming Ensemble Algorithm (SEA) [77], which is a majority-voting ensemble approach that constantly replaces the weakest classifier in the ensemble with a quality measure that considers both the accuracy and diversity ofclassifiers in the ensemble. Cano and Krawczyk propose the Kappa Updated Ensemble (KUE) [12], which is a combination of online and block-based ensemble approaches. KUE uses the Kappa statistic for dynamic weighing and selection of base classifiers. Other advanced techniques in handling concept drift include an enhancement of the time-based ensemble methods by Krawczyk et al. [43] that improves the model's robustness to drift and noise by adding abstaining options to classifiers, allowing classifiers in the ensemble to refrain from making a decision if they have a confidence level below a specified threshold. Cano et al. [11] propose a rule-based classifier for drifting data streams using grammar-guided genetic programming. The model, namely, evolving rule-based classifier for drifting data streams (ERulesD2S), can provide accurate predictions and adapt to concept drift while offering the full interpretability based on classification rules.
Modified on	10/02/2022 11:01

Name	An Empirical Study on Deployment Faults of Deep Learning Based Mobile Applications
Number of Coding References	13
Number of Codes Coding	2
Coverage	6.69%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	06/01/2022 15:05
Coded Text	
Modified on	10/02/2022 10:53

Name	An Empirical Study on Deployment Faults of Deep Learning Based Mobile Applications
Number of Coding References	13
Number of Codes Coding	2
Coverage	6.69%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	06/01/2022 15:05
Coded Text	
Modified on	10/02/2022 10:54

Name	An Empirical Study on Deployment Faults of Deep Learning Based Mobile Applications
Number of Coding References	13
Number of Codes Coding	2
Coverage	6.69%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	06/01/2022 15:05
Coded Text	
Modified on	10/02/2022 10:54

Name	An Empirical Study on Deployment Faults of Deep Learning Based Mobile Applications
Number of Coding References	13
Number of Codes Coding	2
Coverage	6.69%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	06/01/2022 15:05
Coded Text	After building projects, developers can run mobile apps to make it predictable. However, in this phase, many developers encounter Framework loading failure (B.2) and Model loading failure (B.3), which refer to the failures in loading DL frameworks and models respectively and account for a total of 36.8% of faults in DL Integration. What is more, developers may configure projects to make it able to use the GPU backend on mobile devices. However, some developers complain that they encounter the GPU delegate failure (B.4) when running mobile DL apps. B.4 represents 21.1% of faults in DL Integration. Finding 3: Faults appearing in the DL integration stage account for 12.5% of the total deployment faults and cover five symptom categories. A large proportion (34.2%) of these faults are thrown with dependency resolution errors.
Modified on	10/02/2022 10:53

Name	An Empirical Study on Deployment Faults of Deep Learning Based Mobile Applications
Number of Coding References	13
Number of Codes Coding	2
Coverage	6.69%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	06/01/2022 15:05
Coded Text	Besides the faults with explicit errors thrown during the model conversion stage, sometimes developers get unexpected models even after model conversion appears to be successfully done. For example, developers may find that the number, shape, or format of input/output tensors of the model changes. We classify these cases into the category Unexpected model (A.11), accounting for 4.1% faults in Model Conversion. Finding 2: Most (i.e., 48.4%) of deployment faults occur during the model conversion stage, covering a wide spectrum of symptoms (i.e., 12 categories). Among these categories, unsupported operation is the most common, accounting for 31.3% of faults in this stage
Modified on	10/02/2022 10:53
Name	An Empirical Study on Deployment Faults of Deep Learning Based Mobile Applications
Number of Coding References	13
Number of Codes Coding	2
Coverage	6.69%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	06/01/2022 15:05
Coded Text	Deep learning (DL) is moving its step into a growing number of mobile software applications. These software applications, named as DL based mobile applications (abbreviated as mobile DL apps) integrate DL models trained using large-scale data with DL programs. A DL program encodes the structure of a desirable DL model and the process by which the model is trained using training data. Due to the increasing dependency of current mobile apps on DL, software engineering (SE) for mobile DL apps has become important. However, existing efforts in SE research community mainly focus on the development of DL models and extensively analyze faults in DL programs. In contrast, faults related to the deployment of DL models on mobile devices (named as deployment faults of mobile DL apps) have not been well studied. Since mobile DL apps have been used by billions of end users daily for various purposes including for safety-critical scenarios, characterizing their deployment faults is of enormous importance.
Modified on	10/02/2022 10:48

Name	An Empirical Study on Deployment Faults of Deep Learning Based Mobile Applications
Number of Coding References	13
Number of Codes Coding	2
Coverage	6.69%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	06/01/2022 15:05
Coded Text	Finding 7: The fix strategies for faults in inference are diverse. They cover many stages of the deployment process, including fixing data processing, fixing the model conversion stage (e.g., fixing/using quantization), fixing the DL integration stage (e.g., fixing API usage during DL integration), etc.
Modified on	10/02/2022 10:55

Name	An Empirical Study on Deployment Faults of Deep Learning Based Mobile Applications
Number of Coding References	13
Number of Codes Coding	2
Coverage	6.69%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	06/01/2022 15:05
Coded Text	In addition to the faults that affect the output results, there are also 25.5% of faults that have impact on the memory usage and inference speed of mobile DL apps. We use Memory issue (D.4) and Speed issue (D.5) to refer to the two types of faults. Specifically, Memory issue (D.4) includes symptoms such as out of memory, memory leak, failures in memory allocation, and segment faults; Speed issue (D.5) is mainly manifested as long latency time of making inference. Finding 4: 36.2% of faults occur when mobile DL apps make inference based on input data, covering six symptom categories. In particular, 35.5% of the faults in this stage are captured since developers observe unexpected results.
Modified on	10/02/2022 10:53

Name	An Empirical Study on Deployment Faults of Deep Learning Based Mobile Applications
Number of Coding References	13
Number of Codes Coding	2
Coverage	6.69%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	06/01/2022 15:05
Coded Text	IV. RQ1: SYMPTOMS Fig. 3 presents the hierarchical taxonomy of deployment fault symptoms of mobile DL apps. The taxonomy is organized into three-level categories, including a root category (i.e., Deployment Faults), five inner categories linked to stages in deploying DL models (e.g., Model Conversion), and 23 specific leaf categories (e.g., Model parse failure). Finding 1: We construct a taxonomy of 23 fault symptom categories related to deploying DL models on mobile devices, indicating the diversity of deployment faults. For each category, the number in the top right corner refers to the number of faults in it. Due to space limit, we address only frequent and non-trivial symptoms (i.e., #faults ≥ 3). For Data Preparation and Model Update
Modified on	10/02/2022 10:53

Name	An Empirical Study on Deployment Faults of Deep Learning Based Mobile Applications
Number of Coding References	13
Number of Codes Coding	2
Coverage	6.69%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	06/01/2022 15:05
Coded Text	To fill in the knowledge gap, this paper presents the first comprehensive study on analyzing symptoms and fix strategies of deployment faults of mobile DL apps. Given the surging popularity of mobile DL apps, this study is of enormous importance. It can help in understanding what are the common deployment faults of mobile DL apps and how these faults are resolved in practice, so as to provide a high-level categorization that can serve as a guide for developers to resolve common faults and for researchers to develop tools for detecting and fixing deployment faults of the increasing mobile DL apps.
Modified on	10/02/2022 10:51

Name	An Empirical Study on Deployment Faults of Deep Learning Based Mobile Applications
Number of Coding References	13
Number of Codes Coding	2
Coverage	6.69%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	06/01/2022 15:05
Coded Text	To fill in the knowledge gap, this paper presents the first comprehensive study to date on the deployment faults of mobile DL apps. We identify 304 real deployment faults from Stack Overflow and GitHub, two commonly used data sources for studying software faults. Based on the identified faults, we construct a fine-granularity taxonomy consisting of 23 categories regarding to fault symptoms and distill common fix strategies for different fault symptoms. Furthermore, we suggest actionable implications and research avenues that can potentially facilitate the deployment of DL models on mobile devices.
Modified on	10/02/2022 10:48

Name	An Empirical Study on Deployment Faults of Deep Learning Based Mobile Applications
Number of Coding References	13
Number of Codes Coding	2
Coverage	6.69%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	06/01/2022 15:15
Coded Text	Deep learning (DL) is moving its step into a growing number of mobile software applications. These software applications, named as DL based mobile applications (abbreviated as mobile DL apps) integrate DL models trained using large-scale data with DL programs. A DL program encodes the structure of a desirable DL model and the process by which the model is trained using training data. Due to the increasing dependency of current mobile apps on DL, software engineering (SE) for mobile DL apps has become important. However, existing efforts in SE research community mainly focus on the development of DL models and extensively analyze faults in DL programs. In contrast, faults related to the deployment of DL models on mobile devices (named as deployment faults of mobile DL apps) have not been well studied. Since mobile DL apps have been used by billions of end users daily for various purposes including for safety-critical scenarios, characterizing their deployment faults is of enormous importance.
Modified on	10/02/2022 10:48

Name	An Empirical Study on Deployment Faults of Deep Learning Based Mobile Applications
Number of Coding References	13
Number of Codes Coding	2
Coverage	6.69%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	06/01/2022 15:15
Coded Text	Recently, the rapid growth of mobile DL apps [22] has posed urgent challenges to the deployment of DL models, i.e., deploying DL models on mobile devices. For example, computation-intensive DL models can be executed efficiently on PC/server platforms, but they cannot be directly deployed and executed on mobile devices with limited computing power [23]. Although major vendors have rolled out specific DL frameworks such as TF Lite [24] and Core ML [25] to facilitate this deployment process, various specific faults are still emerging in this process and frequently asked on Stack Overflow (SO), one of the most popular Q&A forums for developers [13]. Moreover, previous work [13] has demonstrated that relevant questions are increasing rapidly on SO and more difficult to resolve than those related to other aspects of DL based applications. In addition, mobile DL apps are not only used by billions of end users for their daily activities (e.g., speech-to-text and photo beauty) [22], [26], but also reported to be increasingly adopted in various safety-critical scenarios (e.g., driver assistance [27] and autonomous vehicles [28]). Therefore, the emerging faults related to the deployment of DL models on mobile devices (named as deployment faults of mobile DL apps) should be carefully addressed. Unfortunately, the characteristics of these faults have not been well understood.
Modified on	10/02/2022 10:51

Name	An empirical study on ML DevOps adoption trends, efforts, and benefits analysis
Number of Coding References	3
Number of Codes Coding	1
Coverage	2.13%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 10:15
Coded Text	In this section, we discuss the implications of our empirical analysis. The following is a list of actionable items we identified: • Our analysis on DevOps adoption rates and trends, detailed in Section 4.1, identified that ML Applied projects were slow in adopting DevOps. They also had a lower adoption across different DevOps tool categories such as Build, CI and Code Analyzer. While analyzing the exact reasons behind the barriers to adoption of DevOps tools is by ML projects is not within this work's scope, our results shed a light on the necessity for researchers to study the barriers to adopting DevOps in ML projects and identify possible improvement scopes. These may include ML DevOps task automation, DevOps tools for ML models evaluation and monitoring, etc. On the other hand, tool developers can employ program analysis [71] techniques to automatically generate ML DevOps configuration files which can lower the barriers of entry for data scientists who might be unfamiliar with DevOps concepts and practices. • Our DevOps tool maintenance effort analysis, detailed within Sections 4.2.1 and 4.2.2, reveals that even though ML Applied projects much less adoption of DevOps than the other two categories (ML Tools and Non-ML projects), their developers are changing DevOps configuration files more frequently. This highlights the necessity of working on support for automatic synchronization of DevOps configuration files. This may be provided via change recommendation tools [72], safe refactoring tools [73], and others. These tools can help reduce maintenance overhead, and can provide technical support to developers and data scientists who may not be very familiar with DevOps tools. • Our analysis on events that trigger DevOps file changes, within Section 4.2.3, identified that bug-fixing commits within Tool project that alter DevOps configuration files were much more prevalent in comparison to ML Applied and Non-ML projects. This indicates that the software maintenance research community should invest more heavily in co-evolution analysis [74] of functional code and DevOps configuration files to facilitate early bug-detection. In turn, this will save both time and resources and allow teams to invest them in improving their software product's quality and reputation, rather than resolving problems within it. • Our analysis on DevOps adoption advantages, within Section 4.3, identified that for all project types, adopting DevOps has positive consequences on the code sharing and code integration speed and frequency and helped decrease the duration necessary for issue resolution and improve its quality. Even though using DevOps tools for all types of projects, including ML projects, introduced adoption and maintenance overhead, it appears that the benefits of DevOps outweigh the associated costs. Thus, data scientists and ML developers should adopt DevOps tools within their projects. Furthermore, we believe that adopting DevOps tools present these benefits for all ML projects, even for those with smaller teams. This is especially prevalent in the case of ML Applied projects, which had smaller team sizes overall but generally saw larger improvements resultant of DevOps adoption than ML Tool projects.
Modified on	16/02/2023 09:55

Name	An empirical study on ML DevOps adoption trends, efforts, and benefits analysis
Number of Coding References	3
Number of Codes Coding	1
Coverage	2.13%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 10:15
Coded Text	In this study, we conducted an empirical study on 4031 ML projects and a comparative set of 4076 Non-ML projects hosted in GitHub for ML DevOps adoption, maintenance effort and benefit analysis. Through our analysis, we found evidence of a lower adoption of DevOps tools within ML Applied projects, as well as different development practices and efforts in relation to these files that tended to be less efficient than those of ML Tool and Non-ML projects. In contrast, this type of projects has the most to gain from adopting these tools, and with similar advantages for both ML Tool and Non-ML projects. To the best of our knowledge, this is the first large scale empirical study on ML DevOps adoption, maintenance effort and benefit analysis. This exploratory work lays the foundation for future works, where we plan to investigate the roadblocks developers encounter when adopting different DevOps tools and the features they need to adopt to ease their adoption by ML developers. Our data and code are available at
Modified on	16/02/2023 09:55

Name	An empirical study on ML DevOps adoption trends, efforts, and benefits analysis
Number of Coding References	3
Number of Codes Coding	1
Coverage	2.13%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 10:15
Coded Text	Recently, there have been many works focused on ML DevOps support. MLFlow [9] and Amazon SageMaker [10] were designed to improve the workflow of ML project development, which involves the data collection, data preparation, model definition and training, and results-testing [11]. Package managers such as Spack [12] and EasyBuild [13] were conceived to allow the automatic rebuilding of ML models. Container-based technology such as Docker [14] and Kubernetes [15] has proven apt for shareable models. Aguilar et al. [16] proposed Ease.ml/CI for continuous integration (CI) and data management within ML projects. Fursin et al. [17] proposed CodeReef to perform benchmarking for ML projects and enable their reusable automation. However, the majority of these tools are still premature, require an important development effort, and can only be used in conjunction with specific ML technologies or frameworks [17–19].
Modified on	16/02/2023 09:52

Name	An empirical study on ML DevOps adoption trends, efforts, and benefits analysis Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	An Empirical Study on Real Bugs for Machine Learning Programs
Number of Coding References	21
Number of Codes Coding	2
Coverage	14.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	
Modified on	09/02/2022 13:57

Name	An Empirical Study on Real Bugs for Machine Learning Programs
Number of Coding References	21
Number of Codes Coding	2
Coverage	14.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	
Modified on	09/02/2022 13:58

Name	An Empirical Study on Real Bugs for Machine Learning Programs
Number of Coding References	21
Number of Codes Coding	2
Coverage	14.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	
Modified on	09/02/2022 13:58

Name	An Empirical Study on Real Bugs for Machine Learning Programs
Number of Coding References	21
Number of Codes Coding	2
Coverage	14.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	1) Modify assertion module (MAM): ASSERT () is
Modified on	09/02/2022 13:56

Name	An Empirical Study on Real Bugs for Machine Learning Programs
Number of Coding References	21
Number of Codes Coding	2
Coverage	14.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	2) Add loop module (ALM): This fix pattern is widely used in ML projects. Adding loop statements, in many cases, matches the if statement to check
Modified on	09/02/2022 13:56

Name	An Empirical Study on Real Bugs for Machine Learning Programs
Number of Coding References	21
Number of Codes Coding	2
Coverage	14.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	3) Add test module (ATM): Module testing is a small piece of code to verify that a small, well-defined function of the code being tested has high quality. In general, a unit test is used to judge the behavior of a particular function under a given condition.
Modified on	09/02/2022 13:56

Name	An Empirical Study on Real Bugs for Machine Learning Programs
Number of Coding References	21
Number of Codes Coding	2
Coverage	14.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	4) Add some prompt information (ASPI): Such fix pattern is to give prompt message to users to help them
Modified on	09/02/2022 13:57

Name	An Empirical Study on Real Bugs for Machine Learning Programs
Number of Coding References	21
Number of Codes Coding	2
Coverage	14.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	A. Bug Categories in ML Projects In our study, we further divided compatibility bugs into five types of bugs: In COS, there are some incompatible problems due to platform problems, so we suggest that developers need to do some tests on the mainstream platform to ensure the stability of the new tool when developers publish a new toolkit. In CAV, many users often open some similar bugs, which are often due to the fact that the developers do not update the document in a timely manner, so we suggest that developers should release the latest fix version in a timely manner and send some update news to notify the users. For technical documentation bug, we found that the cause of these bugs is mainly because there are some
Modified on	09/02/2022 13:58

Name	An Empirical Study on Real Bugs for Machine Learning Programs
Number of Coding References	21
Number of Codes Coding	2
Coverage	14.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	B. Fix Pattern for ML Bugs In total, we find that there are twelve commonly used fix patterns for bugs in ML projects. In our study, we find that AIM, CIC and MCV are commonly used to resolve ML bugs. This means that developers can give priority to using these fix patterns when encountering bugs in ML projects. In addition, these fix patterns may be also useful in some
Modified on	09/02/2022 13:58

Name	An Empirical Study on Real Bugs for Machine Learning Programs
Number of Coding References	21
Number of Codes Coding	2
Coverage	14.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	C. Software Maintenance Finally, we find that corrective maintenance is the most occurred activity during ML program evolution. This means that during ML project implementation, developers need to do more testing. In addition, the academic and industry need to develop some testing techniques and tools fit for ML projects.
Modified on	09/02/2022 13:58

Name	An Empirical Study on Real Bugs for Machine Learning Programs
Number of Coding References	21
Number of Codes Coding	2
Coverage	14.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	Our study provides a series of empirical results that can answer the four research questions. 329 studied bugs can be categorized into seven categories, and twelve fix patterns were identified. For these bugs, 68.4% were fixed timely (within 1 month) and more than 37% of them were fixed with micro-fix mode. For ML projects, corrective maintenance activity accounts for 47.77%. Based on these results, our study suggests possible improvements on ML projects for bug fixing and prevention, for example: Automatic bug fixing techniques for certain types of ML bugs can be used to quickly fix ML bugs. Developers can invite other ML project developers to work together to solve some difficult open bugs. When developers release a new feature, they should test it in the mainstream platform, which can avoid too much adaptive maintenance activiey. Developers can specifically arrange some people to update the technical documentation in a timely manner, and promptly remind the users to update the tool version.
Modified on	09/02/2022 13:55

Name	An Empirical Study on Real Bugs for Machine Learning Programs
Number of Coding References	21
Number of Codes Coding	2
Coverage	14.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	RQ2: FIX PATTERN Software fix pattern can effectively help developers fix bugs, and is conducive to the implementation of automated bug fixing technology [14], [15]. We manually analyzed the change history of these bug commits in Github, and found twelve fix patterns, as shown in Figure 1. For the if fix pattern, this is a common pattern (about 29.03%) to fix the bugs in ML projects. According to our study, we find that there are two kinds of if fix patterns as follows:
Modified on	09/02/2022 13:56

Name	An Empirical Study on Real Bugs for Machine Learning Programs
Number of Coding References	21
Number of Codes Coding	2
Coverage	14.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	RQ4: MAINTENANCE TYPES OF ML ISSUES
Modified on	09/02/2022 13:57

Name	An Empirical Study on Real Bugs for Machine Learning Programs
Number of Coding References	21
Number of Codes Coding	2
Coverage	14.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	To summarize, we analyze the fix patterns for bugs of ML programs during their fixing process, and find that there are commonly twelve fix patterns to fix bugs in ML programs.
Modified on	09/02/2022 13:57

Name	An Empirical Study on Real Bugs for Machine Learning Programs
Number of Coding References	21
Number of Codes Coding	2
Coverage	14.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	To summarize, we find that most of the bugs (68.39%) can be solved by developers within a month, especially 40.73% of bugs can be solved in a short time on ML projects. What’s more, 37.87% bug fix belong to microscale-fix and 18.30% bug fix belong to large-scale-fix for the bugs in our study.
Modified on	09/02/2022 13:57

Name	An Empirical Study on Real Bugs for Machine Learning Programs
Number of Coding References	21
Number of Codes Coding	2
Coverage	14.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	03/02/2022 14:17
Coded Text	we conduct an empirical study on real machine learning bugs to examine their patterns and how they evolve over time. We collect three popular machine learning projects on Github, and manually analyzed 329 closed bugs from the perspectives of their bug category, fix pattern, fix scale, fix duration, and type of software maintenance. The results show that (1) there are seven categories of bugs in machine learning programs; (2) twelve different fix patterns are commonly used to fix the bugs; (3) 63.83% of the patches belong to micro-scale-fix and small-scale-fix, and 68.39% of the bugs are fixed within one month; (4) 47.77% of the bug fixes belong to corrective activity from the view of software maintenance.
Modified on	09/02/2022 13:53

Name	An Empirical Study on Real Bugs for Machine Learning Programs
Number of Coding References	21
Number of Codes Coding	2
Coverage	14.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	03/02/2022 14:16
Coded Text	
Modified on	09/02/2022 13:55

Name	An Empirical Study on Real Bugs for Machine Learning Programs
Number of Coding References	21
Number of Codes Coding	2
Coverage	14.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	03/02/2022 14:16
Coded Text	Due to the availability of various open source Ma- chine Learning (ML) tools and libraries, developers nowadays can easily implement their purposes by just invoking machine learning APIs without knowing the details of the algorithm. However, the owners of ML tools and libraries usually pay more attention to the correctness and functionality of their algorithm, while spending much less effort on maintaining their code and keeping their code at a high quality level. Considering the popularity of machine learning in today's world, low quality ML tools and libraries can have a huge impact on the software products that use ML algorithms.
Modified on	09/02/2022 13:53

Name	An Empirical Study on Real Bugs for Machine Learning Programs
Number of Coding References	21
Number of Codes Coding	2
Coverage	14.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	03/02/2022 14:16
Coded Text	For compatibility bug, we can find that there are many different reasons leading to these bugs. Based on our analysis, we identify four bug reasons for compatibility bug, including: (1) Conflicts with the OS (COS): The COS problem for compatibility bugs accounts for 44.59%, which is the highest. For example, the bug in Scikit-learn, i.e., BUG#1225, which is a bug about installation failed in Solaris x86 64bit. (2) Conflicts with algorithm model version (CAV): We notice that 13.51% of the compatibility bug is about CAV problem. We find this kind of bug in both Scikit-learn and Paddle. For example, BUG#1140 in Scikit-learn, the user cannot build docs by using docutils 0.9.1; BUG#18 in Paddle, the bug is about the compatible issue with CUDA 8.0. (3) Conflicts with hardware (CH): In our study, we also find that about 22.98% of compatibility bugs are caused by hardware conflicts. For example, Paddle BUG#1444, which is about Py swig prediction tools with GPU configuration error. (4) Conflicts with platform software (CPS):In caffe, we find that 40.91% compatibility bugs belong to this kind. This is because that users develop their software in different
Modified on	09/02/2022 13:55

Name	An Empirical Study on Real Bugs for Machine Learning Programs
Number of Coding References	21
Number of Codes Coding	2
Coverage	14.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	03/02/2022 14:16
Coded Text	For variable bug (VB), there are about 30% of ML bugs belonging to variable bug. We find that most of the open source projects have this kind of bug. According to our study, we divide this kind of bugs into five sub categories, including: (1) Variable Transfer (VT): Variable transfer is a typical kind of variable bugs. For example, BUG#870 in Scikit-learn is about forward pass of HMM, which creates NAN when transition matrix contains zeros; BUG#1059 is about referencing best params before variable assignment. (2) Variable Evaluation (VA): This kind of bug usually behaves in its own way. For example, BUG#364 in Paddle is about using the forecasting tool to predict the sparse binary vector's instance. (3) Variable Constraint (VC): This kind of bug is also a common problem in the variable bug. For example, BUG#1509 in Scikit-learn is about the threshold parameter of the squared Hinge loss, which is ignored. (4) Data Format (DF): This kind of bug is mainly caused by parameters. For example, BUG#1995 in Scikit-learn. (5) Data Initialization (DI): This kind of bug is about the
Modified on	09/02/2022 13:55

Name	An Empirical Study on Real Bugs for Machine Learning Programs
Number of Coding References	21
Number of Codes Coding	2
Coverage	14.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	03/02/2022 14:16
Coded Text	In the field of software engineering (SE), many ML methods have been successfully applied to bug prediction [9], [10], [11], bug localization [12], [13], automated and semiautomated bug fixing [14], [15], [16], code recommendation [17],[18], code searching [19], and many others [?], [20]. As a result, the quality of many software engineering tasks is highly dependent on the quality of ML algorithms, and it is common that those SE tasks are taking advantage of the existing ML projects or libraries, which are the implementation of ML algorithms, without implementing ML algorithms themselves. Therefore, the quality of ML projects and libraries has a huge impact on software engineering research and practice. In reality, however, many people tend to trust the correctness of ML projects and libraries without questioning the potential defects contained in them. Like many other software projects, bugs are also common in ML projects and libraries. Because of the popularity and large-scale usage of ML projects, even a little bit bugs may lead to severe consequences and impact a huge number of users.
Modified on	09/02/2022 13:54

Name	An Empirical Study Towards Characterizing Deep Learning Development and Deployment Across Different Frameworks and Platforms
Number of Coding References	10
Number of Codes Coding	3
Coverage	6.94%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	Answer to RQ1: Given DL models with the same runtime configuration, PYTORCH generally provides more stable training and validation process than TENSORFLOW, CNTK, and MXNET in our study. Although it is understandable that the computing differences exist across frameworks, such differences can sometimes be very obvious under certain scenarios (e.g., model conversion), leading to a misclassification on DL models. The existing model conversion between frameworks is currently not reliable due to the computing differences, which requires special attention and inspection before applying directly. Note that, 100% participates in the questionnaire are interested in the quantitative differences across frameworks and the corresponding results can be used to provide development insights. Challenge: How to identify real framework bugs according to the computing differences? How to amplify the computing differences to help find more similar issues in SE testing field?
Modified on	28/02/2023 15:09

Name	An Empirical Study Towards Characterizing Deep Learning Development and Deployment Across Different Frameworks and Platforms
Number of Coding References	10
Number of Codes Coding	3
Coverage	6.94%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	Answer to RQ2: Given the same architecture design and runtime configuration, DL models from different frameworks exhibit diverse robustness against adversarial attacks. Generally, CNTK achieves the most robust result in our evaluated settings among all the frameworks when training DL models, and models trained from PYTORCH and MXNET tend to be more vulnerable to adversarial attacks. Challenge: How to improve the robustness of DL models in training stage from the perspective of engineering DL frameworks? How to develop advanced testing techniques to generate specific tests for improving robustness?
Modified on	28/02/2023 15:09

Name	An Empirical Study Towards Characterizing Deep Learning Development and Deployment Across Different Frameworks and Platforms
Number of Coding References	10
Number of Codes Coding	3
Coverage	6.94%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	Answer to RQ3: Different platform devices hold different time and memory performance in capability of supporting DL software. For mobile devices, Android devices take much less time than iOS devices for simple DNN models. However, as the complexity of the model increases, iOS devices achieve better time performance. Moreover, the capability of supporting DL software on mobile platform is likely related to the types of specific DNN models. For web platforms, Chrome generally outperforms others in both prediction time cost and system memory consumption in our study. The overall performance for web DL software is unsatisfactory, especially running complex DL models. Challenge: How to reduce the time cost memory consumption after model migration and quantization? How to further test the performance of different platforms when deploying and running DL software systematically?
Modified on	28/02/2023 15:09
Name	An Empirical Study Towards Characterizing Deep Learning Development and Deployment Across Different Frameworks and Platforms
Number of Coding References	10
Number of Codes Coding	3
Coverage	6.94%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	Answer to RQ4-1: The prediction accuracy on original data has not been affected much by the migration process. However, compatibility issues persist in model migration from PC to browsers (e.g., 77.08 vs. 82.66 on ResNet20). Even worse, there still exists a obvious difference on computation mechanism between PC and web browsers, leading to a computing distinction of each layer within the model, which has been acknowledged and confirmed by the team of TENSORFLOW.JS. This result explains why the industry has failed to meet expectations after model migration based on our online questionnaire, which provides a reasonable explanation for the industrial developers.
Modified on	28/02/2023 15:09

Name	An Empirical Study Towards Characterizing Deep Learning Development and Deployment Across Different Frameworks and Platforms
Number of Coding References	10
Number of Codes Coding	3
Coverage	6.94%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	Answer to RQ4-2: Quantization does not affect the prediction accuracy obviously. Prediction on Android devices after quantization is faster than the original model, and the improvement is more significant for complex models. Strikingly, quantization on iOS devices slows down the prediction speed, which deserves further optimization for CORE ML.
Modified on	28/02/2023 15:09

Name	An Empirical Study Towards Characterizing Deep Learning Development and Deployment Across Different Frameworks and Platforms
Number of Coding References	10
Number of Codes Coding	3
Coverage	6.94%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	In this paper, we initiate the first step to investigate how existing deep learning frameworks and platforms influence the development and deployment of deep learning software. Our study provides many practical guidelines for developers and researchers under different scenarios for different research communities. Given the same model weights/biases, an obvious accuracy decline occurs when the model is converted from one framework to another. The compatibility and reliability issues and accuracy loss would arise when migrating and quantizing a deep learning model from the PC platform to other platforms, and the accuracy loss is due to several deep learning software bugs we found. In addition, the universal deep learning solutions across platforms are desperately on demand, especially for mobile and web platforms. This study makes the first step along this direction towards building universal deep learning software across various platforms based on our practical guidelines. We hope our work draws the attention of deep learning software community, altogether to address the urgent demands towards the new challenges in deep learning software development and deployment processes.
Modified on	28/02/2023 15:09

Name	An Empirical Study Towards Characterizing Deep Learning Development and Deployment Across Different Frameworks and Platforms
Number of Coding References	10
Number of Codes Coding	3
Coverage	6.94%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	migration and quantization on mobile and web platforms. Although the diverse DL frameworks and platforms promote the evolution of DL software, understanding the characteristics of them becomes a time-consuming task for DL software developers and researchers. Moreover, the differences compared with the traditional software brings new challenges for DL software development and deployment processes. These challenges include that (1) for the development process, there lacks a deep understanding of various frameworks under a) the training and prediction accuracy given the same runtime configuration; b) the prediction accuracy given the same model weights/biases; and c) the robustness of trained models. (2) For the deployment process, when deploying the trained models from PC/Server to different platforms, there lacks a benchmarking understanding of the migration and quantization processes, such as the impacts on prediction accuracy, performance (i.e., time cost and memory consumption). To address the aforementioned challenges, with an over ten man-month effort, we design and perform an empirical study on the state-of-the-art DL frameworks and platforms from two aspects to investigate the following research questions. (1) As for the development process: • RQ1: Accuracy on different frameworks. Given the same runtime configuration or same model weights/biases, what are the differences of training and prediction accuracy when implemented with different DL frameworks? • RQ2: Adversarial robustness of trained models. Do DL models trained from different DL frameworks exhibit the same adversarial robustness against adversarial examples? (2) As for the deployment process: • RQ3: Performance after migration and quantization. What are the differences of performance (i.e., time cost and memory consumption) in the capabilities of supporting DL software when migrating or quantizing the trained models to the real mobile devices and web browsers? • RQ4: Prediction accuracy after migration and quantization. Given the same trained DL model, what is the prediction accuracy of the migrated model for mobile and web platforms? How do quantization methods influence the prediction accuracy of quantized model on mobile devices? Through answering these research questions, we aim to characterize the impacts of current DL frameworks and platforms on DL software development and deployment processes, and provide practical guidelines to developers and researchers from different research communities such as SE and AI fields and under different practical scenarios. In summary, we make the following main contributions: • To the best of our knowledge, this is the first empirical study on how the current DL frameworks and platforms influence the development and deployment processes, especially for the study on the migration and quantization processes on different DL platforms.
Modified on	28/02/2023 15:08

Name	An Empirical Study Towards Characterizing Deep Learning Development and Deployment Across Different Frameworks and Platforms
Number of Coding References	10
Number of Codes Coding	3
Coverage	6.94%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	Remarks for inspection of generated data: (1) The accuracy of migrated models does not change in our evaluation on Android devices, while has a relatively obvious decline on iOS devices. As for the web platforms, the results (i.e., compatibility bugs) are consistent to that on original data. (2) The accuracy of all quantized models has a significant decline on our generated testing data, which indicates the quantization process still suffers from severe reliability issues tested by generated data. Meanwhile, the decline is correlated with the value nbits when reducing the floating point on iOS devices. (3) Furthermore, we conduct statistical analysis [13] on the accuracy-dropping cases in Column Generated after quantization of Table III. The results give a p< 0.05, indicating there exists a statistically significant difference in accuracy on generated data, which reconfirms the reliability issues. Challenge: How to detect and fix the compatibility issues/bugs when migrating the trained models to web platforms and iOS devices, and the reliability issues when quantizing the trained models to mobile platforms?
Modified on	28/02/2023 15:09

Name	An Empirical Study Towards Characterizing Deep Learning Development and Deployment Across Different Frameworks and Platforms
Number of Coding References	10
Number of Codes Coding	3
Coverage	6.94%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13
Coded Text	Deep Learning (DL) has recently achieved tremen- dous success. A variety of DL frameworks and platforms play a key role to catalyze such progress. However, the differences in architecture designs and implementations of existing frameworks and platforms bring new challenges for DL software development and deployment. Till now, there is no study on how various mainstream frameworks and platforms influence both DL software development and deployment in practice. To fill this gap, we take the first step towards understanding how the most widely-used DL frameworks and platforms support the DL software development and deployment. We conduct a systematic study on these frameworks and platforms by using two types of DNN architectures and three popular datasets. (1) For development process, we investigate the prediction accuracy under the same runtime training configuration or same model weights/biases. We also study the adversarial robustness of trained models by leveraging the existing adversarial attack techniques. The experimental results show that the computing differences across frameworks could result in an obvious prediction accuracy decline, which should draw the attention of DL developers. (2) For deployment process, we investigate the prediction accuracy and performance (refers to time cost and memory consumption) when the trained models are migrated/quantized from PC to real mobile devices and web browsers. The DL platform study unveils that the migration and quantization still suffer from compatibility and reliability issues. Meanwhile, we find several DL software bugs by using the results as a benchmark. We further validate the results through bug confirmation from stakeholders and industrial positive feedback to highlight the implications of our study. Through our study, we summarize practical guidelines, identify challenges and pinpoint new research directions, such as understanding the characteristics of DL frameworks and platforms, avoiding compatibility and reliability issues, detecting DL software bugs, and reducing time cost and memory consumption towards developing and deploying high quality DL systems effectively.
Modified on	28/02/2023 15:08

Name	An Empirical Study Towards Characterizing Deep Learning Development and Deployment Across Different Frameworks and Platforms
Number of Coding References	10
Number of Codes Coding	3
Coverage	6.94%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	Deep Learning (DL) has recently achieved tremen- dous success. A variety of DL frameworks and platforms play a key role to catalyze such progress. However, the differences in architecture designs and implementations of existing frameworks and platforms bring new challenges for DL software development and deployment. Till now, there is no study on how various mainstream frameworks and platforms influence both DL software development and deployment in practice. To fill this gap, we take the first step towards understanding how the most widely-used DL frameworks and platforms support the DL software development and deployment. We conduct a systematic study on these frameworks and platforms by using two types of DNN architectures and three popular datasets. (1) For development process, we investigate the prediction accuracy under the same runtime training configuration or same model weights/biases. We also study the adversarial robustness of trained models by leveraging the existing adversarial attack techniques. The experimental results show that the computing differences across frameworks could result in an obvious prediction accuracy decline, which should draw the attention of DL developers. (2) For deployment process, we investigate the prediction accuracy and performance (refers to time cost and memory consumption) when the trained models are migrated/quantized from PC to real mobile devices and web browsers. The DL platform study unveils that the migration and quantization still suffer from compatibility and reliability issues. Meanwhile, we find several DL software bugs by using the results as a benchmark. We further validate the results through bug confirmation from stakeholders and industrial positive feedback to highlight the implications of our study. Through our study, we summarize practical guidelines, identify challenges and pinpoint new research directions, such as understanding the characteristics of DL frameworks and platforms, avoiding compatibility and reliability issues, detecting DL software bugs, and reducing time cost and memory consumption towards developing and deploying high quality DL systems effectively.
Modified on	28/02/2023 15:07

Name	An Empirical Study Towards Characterizing Deep Learning Development and Deployment Across Different Frameworks and Platforms Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	An End-to-End Framework for Productive Use of Machine Learning in Software Analytics and Business Intelligence Solutions
Number of Coding References	21
Number of Codes Coding	3
Coverage	14.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	
Modified on	09/02/2022 13:35

Name	An End-to-End Framework for Productive Use of Machine Learning in Software Analytics and Business Intelligence Solutions
Number of Coding References	21
Number of Codes Coding	3
Coverage	14.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	The framework is structured in three iterative cycles representing different stages in a model’s lifecycle: prototyping, deployment, update. As a result, the framework specifically supports the transitions between these stages while also covering all important activities from data collection to retraining deployed ML models. To validate the applicability of the framework in practice, we compare it to and apply it in a real-world ML-based SA/BI solution.
Modified on	09/02/2022 13:23

Name	An End-to-End Framework for Productive Use of Machine Learning in Software Analytics and Business Intelligence Solutions
Number of Coding References	21
Number of Codes Coding	3
Coverage	14.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	While the literature review examines the topics data management and pro- cessing, model building,and model deployment and serving individually, in reality a separation of the three is not that trivial. In fact, for building end-to-end solutions the fields are very much interrelated as the activities depend on each other and sometimes even overlap. Oftentimes, ML projects start out as a prototypical analysis due to a lim- ited amount of time and resources [15,30]. In order to use and actually benefit from the ML model, it needs to be deployed to a production environment which can be time and cost-intensive but nonetheless crucial [21,30]. To avoid the deployed models from being outdated, it is important to provide a functionality for dynamically deploying new models or iteratively retraining and updating existing models [9,21]. As a result, we identify three iterative cycles which are passed through during an end-to-end development of ML solutions and, therefore, serve as the main dimensions in our framework: 1) Prototyping cycle (blue), 2) deployment cycle (green), and 3) update cycle (orange).
Modified on	09/02/2022 13:39

Name	An End-to-End Framework for Productive Use of Machine Learning in Software Analytics and Business Intelligence Solutions
Number of Coding References	21
Number of Codes Coding	3
Coverage	14.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13
Coded Text	
Modified on	09/02/2022 13:31

Name	An End-to-End Framework for Productive Use of Machine Learning in Software Analytics and Business Intelligence Solutions
Number of Coding References	21
Number of Codes Coding	3
Coverage	14.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13
Coded Text	
Modified on	09/02/2022 13:32

Name	An End-to-End Framework for Productive Use of Machine Learning in Software Analytics and Business Intelligence Solutions
Number of Coding References	21
Number of Codes Coding	3
Coverage	14.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13
Coded Text	
Modified on	09/02/2022 13:32

Name	An End-to-End Framework for Productive Use of Machine Learning in Software Analytics and Business Intelligence Solutions
Number of Coding References	21
Number of Codes Coding	3
Coverage	14.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13
Coded Text	
Modified on	09/02/2022 13:33

Name	An End-to-End Framework for Productive Use of Machine Learning in Software Analytics and Business Intelligence Solutions
Number of Coding References	21
Number of Codes Coding	3
Coverage	14.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13
Coded Text	
Modified on	09/02/2022 13:34

Name	An End-to-End Framework for Productive Use of Machine Learning in Software Analytics and Business Intelligence Solutions
Number of Coding References	21
Number of Codes Coding	3
Coverage	14.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13
Coded Text	
Modified on	09/02/2022 13:34

Name	An End-to-End Framework for Productive Use of Machine Learning in Software Analytics and Business Intelligence Solutions
Number of Coding References	21
Number of Codes Coding	3
Coverage	14.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13
Coded Text	
Modified on	09/02/2022 13:34

Name	An End-to-End Framework for Productive Use of Machine Learning in Software Analytics and Business Intelligence Solutions
Number of Coding References	21
Number of Codes Coding	3
Coverage	14.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13
Coded Text	Data Management and Processing The most important prerequisite for training accurate ML models is providing high-quality training data [26,29]. At the same time, assembling high-quality data sets, and engineering and selecting appropriate features based on it, is very time-consuming and requires a vast amount of effort and resources [14]. As a result, we investigate the common activities (see Table 1) in data man- agement and data processing required for a successful application in machine learning systems as well as the challenges (see Table 2) that come with these activities. The identified activities can be grouped into six overarching categories: 1) Data preparation; 2) data cleaning; 3) data validation; 4) data evaluation; 5) data serving; and 6) extract, transform, and load (ETL) tasks. During the data preparation, raw input data is examined for suitable features before being transformed (e.g. aggregations of one or more raw input data fields) into training data [4,5,14,21,26,27]. Next, the data is cleaned by filtering out uncorrelated data [10,26], specifying quality rules, detecting errors, inconsistencies and anomalies [4,8,19], and fixing these errors [8,19,26,36]. To guarantee a successful preparation and cleaning of the data, each batch of data needs to be validated based on its properties [4,5,26,27,29,36] and potential
Modified on	09/02/2022 13:30

Name	An End-to-End Framework for Productive Use of Machine Learning in Software Analytics and Business Intelligence Solutions
Number of Coding References	21
Number of Codes Coding	3
Coverage	14.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13
Coded Text	dependencies [26], deviations [5,26], or impact of features on model accuracy or performance [14,26] need to be identified. Once a model is trained, the goal of data evaluation is to evaluate the choice and encoding of the data based on the results produced by a model trained on the data, for instance by performing sanity checks [14,26]. After a suitable solution was found, the newly emerging input data needs to be transformed to so-called serving data which is processible by the model [4,26]. This usually involves the same transformation steps as required for the training data. After the serving data was successfully processed by the model, it is channeled back as training data for future iterations [26].
Modified on	09/02/2022 13:30

Name	An End-to-End Framework for Productive Use of Machine Learning in Software Analytics and Business Intelligence Solutions
Number of Coding References	21
Number of Codes Coding	3
Coverage	14.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13
Coded Text	Nowadays, machine learning (ML) is an integral component in a wide range of areas, including software analytics (SA) and business intelligence (BI). As a result, the interest in custom ML-based software analytics and business intelligence solutions is rising. In practice, however, such solutions often get stuck in a prototypical stage because setting up an infrastructure for deployment and maintenance is considered complex and time-consuming. For this reason, we aim at structuring the entire process and making it more transparent by deriving an end-to-end framework from existing literature for building and deploying ML-based software analytics and business intelligence solutions.
Modified on	09/02/2022 13:22

Name	An End-to-End Framework for Productive Use of Machine Learning in Software Analytics and Business Intelligence Solutions
Number of Coding References	21
Number of Codes Coding	3
Coverage	14.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13
Coded Text	there is often a need for customized software analytics or business intelligence (SA/BI) solutions that leverage the full potential of modern machine learning (ML) techniques. However, as such solutions are used as internal systems for monitoring or decision-making, these are often not perceived as something of direct customer value by managers. This results in a lack of priority, time and, resources assigned to setup and maintain ML-based SA/BI solutions [15]. In addition to that, the effort of going beyond a prototypical analysis and deploying it to and maintaining it in production is perceived as extremely high [15,30]. Paired with a lack of expertise in this domain, which is often the case if the actual product is not related to ML [6], custom ML-based SA/BI solutions are rarely deployed in production [15]. Nevertheless, this is considered crucial in order to continuously gain valuable insights and use it for actual decision making [21].
Modified on	09/02/2022 13:28

Name	An End-to-End Framework for Productive Use of Machine Learning in Software Analytics and Business Intelligence Solutions
Number of Coding References	21
Number of Codes Coding	3
Coverage	14.02%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56
Coded Text	
Modified on	09/02/2022 13:32

Name	An End-to-End Framework for Productive Use of Machine Learning in Software Analytics and Business Intelligence Solutions
Number of Coding References	21
Number of Codes Coding	3
Coverage	14.02%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56
Coded Text	
Modified on	09/02/2022 13:33

Name	An End-to-End Framework for Productive Use of Machine Learning in Software Analytics and Business Intelligence Solutions
Number of Coding References	21
Number of Codes Coding	3
Coverage	14.02%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56
Coded Text	
Modified on	09/02/2022 13:34

Name	An End-to-End Framework for Productive Use of Machine Learning in Software Analytics and Business Intelligence Solutions
Number of Coding References	21
Number of Codes Coding	3
Coverage	14.02%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56
Coded Text	
Modified on	09/02/2022 13:34

Name	An End-to-End Framework for Productive Use of Machine Learning in Software Analytics and Business Intelligence Solutions
Number of Coding References	21
Number of Codes Coding	3
Coverage	14.02%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56
Coded Text	
Modified on	09/02/2022 13:34

Name	An End-to-End Framework for Productive Use of Machine Learning in Software Analytics and Business Intelligence Solutions
Number of Coding References	21
Number of Codes Coding	3
Coverage	14.02%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56

Coded Text	Activity Data serving Transformation of serving input data to serving data processible by model Channeling serving data back as training data Extract, transform, load (ETL) Extraction of data from sources Transportation of data to processing pipeline (e.g. for data cleaning or filtering) Transformation of source data to target values Loading of cleaned & transformed data Publications [4,26] [26] [13,34,35] [13,34,35] [13,34,35] [13,34,35]
Modified on	09/02/2022 13:32

Name	An End-to-End Framework for Productive Use of Machine Learning in Software Analytics and Business Intelligence Solutions
Number of Coding References	21
Number of Codes Coding	3
Coverage	14.02%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56
Coded Text	Table 1. Common activities in data management and processing for machine learning Activity Publications Data preparation Identification of features and their properties based on raw data Transformation of input data to training data Data cleaning Investigating and understanding effect of cleaning data on model accuracy & filtering out uncorrelated data Definition of data fixes & execution of error repairs Data validation Triggering validation pipeline for each batch of data Generation of descriptive statistics of data, checking data properties based on specified schema/patterns & identification of errors or anomalies in training data Identification of features with significant impact on model accuracy Identification of dependencies to other data sources or infrastructure [14,21,26] [4,5,14,26,27] [10,26] Ensure data quality, specification of (quality) rules & actions for rules [4,8,19] Detection of data errors [8,19] [8,19,26,36] [5,29,36] [4,5,26,27,29,36] [14,26] [26] Comparison of training and serving data to identify potential deviations [5,26] Data evaluation Performing sanity checks on data [26] Evaluation of choice and encoding of data based on model results [14,26] (continued)
Modified on	09/02/2022 13:32

Name	An Overview of Processing-in-Memory Circuits for Artificial Intelligence and Machine Learning
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	An Overview of Processing-in-Memory Circuits for Artificial Intelligence and Machine Learning Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Apache Mahout~ Machine Learning on Distributed Data ow Systems
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Apache Mahout~ Machine Learning on Distributed Dataflow Systems
Number of Coding References	8
Number of Codes Coding	3
Coverage	14.88%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	Over time, it became apparent that the MapReduce paradigm is suboptimal for the distributed execution of ML algorithms, both for reasons of usability and performance. At the same time, the underlying Hadoop platform had been rewritten to expose resource management and job scheduling capabilities4 to allow systems with parallel processing paradigms different from MapReduce to operate on data stored in the distributed filesystem. Examples of such systems are Apache Spark (Zaharia et al. (2012)) and Apache Flink (Alexandrov et al. (2014)). Unfortunately, these systems are still difficult to program, as their programming model is heavily influenced by the underlying data-parallel execution scheme. Furthermore, the available programming abstractions typically rely on partitioned, unordered bags; this is a mismatch for ML applications that mostly operate on tensors, matrices and vectors. Therefore, implementing ML algorithms on dataflow systems is still a tedious and difficult task. While ML systems such as Tensorflow (Abadi et al. (2016)) excel at efficiently executing programs built from linear algebra operations, they are not designed to execute general dataflow programs and have to rely on complicated integrations with other systems for such operations, e.g., Apache Beam5 in the case of the Tensorflow Extended Platform (Baylor et al. (2017)).
Modified on	09/02/2022 13:17

Name	Apache Mahout~ Machine Learning on Distributed Dataflow Systems
Number of Coding References	8
Number of Codes Coding	3
Coverage	14.88%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	Relationship to the Python ML ecosystem: The majority of recent ML research relies on implementations in Python (e.g., numpy, Scikit-learn (Pedregosa et al. (2011)) or Jupyter), and often operates on single, static and relatively well understood datasets. In contrast to that, production systems typically include complex data integration and preprocessing pipelines, which continuously ingest new data. Such systems are often built on top of the JVM and deployed in the cloud, for reasons of reliability, scalability and ease of operations. Python-based solutions are typically very difficult to integrate into such setups (Schelter et al. (2018)), and therefore JVM-based solutions that only require a single system and code base for the whole pipeline (such as Apache Spark) are often preferred, even though the model training performance might be sub-par in many cases (Boden et al. (2017)). Samsara thereby targets the same set of use cases as the SparkML library (Meng et al. (2016)), which however only exposes a collection of pre-made algorithms and lacks the flexibility offered by a linear algebra DSL such as Samsara, (e.g., to allow users to easily implement their own algorithms or to adjust existing ones).
Modified on	09/02/2022 13:19

Name	Apache Mahout~ Machine Learning on Distributed Dataflow Systems
Number of Coding References	8
Number of Codes Coding	3
Coverage	14.88%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	Apache Mahout is a library for scalable machine learning (ML) on distributed dataflow systems, offering various implementations of classification, clustering, dimensionality reduction and recommendation algorithms. Mahout was a pioneer in large-scale machine learning in 2008, when it started and targeted MapReduce, which was the predominant abstraction for scalable computing in industry at that time. Mahout has been widely used by leading web companies and is part of several commercial cloud offerings. In recent years, Mahout migrated to a general framework enabling a mix of dataflow programming and linear algebraic computations on backends such as Apache Spark and Apache Flink. This design allows users to execute data preprocessing and model training in a single, unified dataflow system, instead of requiring a complex integration of several specialized systems.
Modified on	23/07/2022 11:56

Name	Apache Mahout~ Machine Learning on Distributed Dataflow Systems
Number of Coding References	8
Number of Codes Coding	3
Coverage	14.88%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	Over time, it became apparent that the MapReduce paradigm is suboptimal for the distributed execution of ML algorithms, both for reasons of usability and performance. At the same time, the underlying Hadoop platform had been rewritten to expose resource management and job scheduling capabilities4 to allow systems with parallel processing paradigms different from MapReduce to operate on data stored in the distributed filesystem. Examples of such systems are Apache Spark (Zaharia et al. (2012)) and Apache Flink (Alexandrov et al. (2014)). Unfortunately, these systems are still difficult to program, as their programming model is heavily influenced by the underlying data-parallel execution scheme. Furthermore, the available programming abstractions typically rely on partitioned, unordered bags; this is a mismatch for ML applications that mostly operate on tensors, matrices and vectors. Therefore, implementing ML algorithms on dataflow systems is still a tedious and difficult task. While ML systems such as Tensorflow (Abadi et al. (2016)) excel at efficiently executing programs built from linear algebra operations, they are not designed to execute general dataflow programs and have to rely on complicated integrations with other systems for such operations, e.g., Apache Beam5 in the case of the Tensorflow Extended Platform (Baylor et al. (2017)). As a consequence, Mahout has been rebuilt on top of Samsara (Lyubimov and Palumbo (2016)), a domain-specific language for declarative machine learning in cluster environments. Samsara allows its users to specify programs using a set of common matrix abstractions and linear algebraic operations, which at the same time integrate with existing dataflow operators. Samsara then optimizes and executes these programs on distributed dataflow systems (Schelter et al. (2016)). The aim of Samsara is to allow mathematicians and data scientists to easily integrate their algorithms into ML workloads running on distributed dataflow systems via common declarative abstractions. Figure 1a illustrates the architecture of Samsara. Applications are written using the Scala DSL, and developers have to choose between an in-memory and a distributed representation of matrices used in the program (Figure 1b). Operations on in-memory matrices are executed eagerly, while operations on distributed matrices (which are partitioned among the machines in the cluster) are deferred. The system records the actions to perform on these distributed matrices as a directed acyclic graph (DAG) of logical operations, where vertices refer to matrices and edges correspond to transformations between them.
Modified on	23/07/2022 11:57

Name	Apache Mahout~ Machine Learning on Distributed Dataflow Systems
Number of Coding References	8
Number of Codes Coding	3
Coverage	14.88%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	Relationship to the Python ML ecosystem: The majority of recent ML research relies on implementations in Python (e.g., numpy, Scikit-learn (Pedregosa et al. (2011)) or Jupyter), and often operates on single, static and relatively well understood datasets. In contrast to that, production systems typically include complex data integration and preprocessing pipelines, which continuously ingest new data. Such systems are often built on top of the JVM and deployed in the cloud, for reasons of reliability, scalability and ease of operations. Python-based solutions are typically very difficult to integrate into such setups (Schelter et al. (2018)), and therefore JVM-based solutions that only require a single system and code base for the whole pipeline (such as Apache Spark) are often preferred, even though the model training performance might be sub-par in many cases (Boden et al. (2017)). Samsara thereby targets the same set of use cases as the SparkML library (Meng et al. (2016)), which however only exposes a collection of pre-made algorithms and lacks the flexibility offered by a linear algebra DSL such as Samsara, (e.g., to allow users to easily implement their own algorithms or to adjust existing ones).
Modified on	23/07/2022 11:58

Name	Apache Mahout~ Machine Learning on Distributed Dataflow Systems
Number of Coding References	8
Number of Codes Coding	3
Coverage	14.88%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	
Modified on	09/02/2022 13:18

Name	Apache Mahout~ Machine Learning on Distributed Dataflow Systems
Number of Coding References	8
Number of Codes Coding	3
Coverage	14.88%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Apache Mahout is a library for scalable machine learning (ML) on distributed dataflow systems, offering various implementations of classification, clustering, dimensionality reduction and recommendation algorithms. Mahout was a pioneer in large-scale machine learning in 2008, when it started and targeted MapReduce, which was the predominant abstraction for scalable computing in industry at that time. Mahout has been widely used by leading web companies and is part of several commercial cloud offerings. In recent years, Mahout migrated to a general framework enabling a mix of dataflow programming and linear algebraic computations on backends such as Apache Spark and Apache Flink. This design allows users to execute data preprocessing and model training in a single, unified dataflow system, instead of requiring a complex integration of several specialized systems.
Modified on	09/02/2022 13:15

Name	Apache Mahout~ Machine Learning on Distributed Dataflow Systems
Number of Coding References	8
Number of Codes Coding	3
Coverage	14.88%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	As a consequence, Mahout has been rebuilt on top of Samsara (Lyubimov and Palumbo (2016)), a domain-specific language for declarative machine learning in cluster environments. Samsara allows its users to specify programs using a set of common matrix abstractions and linear algebraic operations, which at the same time integrate with existing dataflow operators. Samsara then optimizes and executes these programs on distributed dataflow systems (Schelter et al. (2016)). The aim of Samsara is to allow mathematicians and data scientists to easily integrate their algorithms into ML workloads running on distributed dataflow systems via common declarative abstractions. Figure 1a illustrates the architecture of Samsara. Applications are written using the Scala DSL, and developers have to choose between an in-memory and a distributed representation of matrices used in the program (Figure 1b). Operations on in-memory matrices are executed eagerly, while operations on distributed matrices (which are partitioned among the machines in the cluster) are deferred. The system records the actions to perform on these distributed matrices as a directed acyclic graph (DAG) of logical operations, where vertices refer to matrices and edges correspond to transformations between them.
Modified on	09/02/2022 13:18

Name	Applied Machine Learning at Facebook~ A Datacenter Infrastructure Perspective
Number of Coding References	24
Number of Codes Coding	3
Coverage	16.69%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	As a general solution, we want to decouple the data workload from the training workload. These two workloads have very different characteristics. The data workload is very complex, ad-hoc, business dependent, and changing fast. The training workload on the other hand is usually regular (e.g. GEMM), stable (there are relatively few core operations), highly optimized, and much prefers a “clean” environment (e.g., exclusive cache usage and minimal thread contention). To optimize for both, we physically isolate the different workloads to different machines. The data processing machines, aka “readers”, read the data from storage, process and condense them, and then send to the training machines aka “trainers”. The trainers, on the other hand, solely focus on executing the training options rapidly and efficiently. Both readers and trainers can be distributed to provide great flexibility and scalability. We also optimize the machine configurations for different workloads. Another important optimization metric is network usage. The data traffic generated by training can be significant and sometimes bursty. If not handled intelligently, this can easily saturate network devices and even disrupt other services. To address these concerns, we employ optimization in compression, scheduling algorithms, data/compute placement, etc.
Modified on	09/02/2022 13:08

Name	Applied Machine Learning at Facebook~ A Datacenter Infrastructure Perspective
Number of Coding References	24
Number of Codes Coding	3
Coverage	16.69%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	B. Leveraging Scale As a company serving users across the world, Facebook must maintain a large fleet of servers designed to handle the peak load at any given time. As seen in Figure 4, due to variations in user activity due to diurnal load and peaks during spe-
Modified on	09/02/2022 13:08

Name	Applied Machine Learning at Facebook~ A Datacenter Infrastructure Perspective
Number of Coding References	24
Number of Codes Coding	3
Coverage	16.69%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	cial events (e.g. regional holidays), a large pool of servers are often idle at certain periods in time. This effectively provides an enormous pool of compute resources available during the off -peak hours. A major ongoing effort explores opportunities to take advantage of these heterogeneous resources that can be allocated to various tasks in an elastic manner. For machine learning applications, this provides a prime opportunity to take advantage of distributed training mechanisms that can scale to a large number of heterogeneous resources (e.g. different CPU and GPU platforms with differing RAM allocations). The sheer scale of the number of compute resources available during these low utilization periods leads to fundamentally different distributed training approaches, imposing a few challenges. The scheduler must first balance the load properly across heterogeneous hardware, so that hosts do not have to wait for each other for synchronization.
Modified on	09/02/2022 13:09

Name	Applied Machine Learning at Facebook~ A Datacenter Infrastructure Perspective
Number of Coding References	24
Number of Codes Coding	3
Coverage	16.69%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54

Coded Text	Disaster Recovery The ability to seamlessly handle the loss of a portion of Facebook’s global compute, storage, and network footprint has been a long-standing goal of Facebook Infrastructure [14]. Internally, our disaster recovery team regularly performs drills to identify and remedy the weakest links in our global infrastructure and software stacks. Disruptive actions include taking an entire data center offline with little to no notice in order to confirm that the loss of any of our global data centers results in minimal disruption to the business.
Modified on	09/02/2022 13:09

Name	Applied Machine Learning at Facebook~ A Datacenter Infrastructure Perspective
Number of Coding References	24
Number of Codes Coding	3
Coverage	16.69%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	Getting Data to the Models For many machine learning models at Facebook, success is predicated on the availability of extensive, high-quality data. The ability to rapidly process and feed these data to the training machines is important for ensuring that we have fast and efficient offline training. For sophisticated ML applications such as Ads and Feed Ranking, the amount of data to ingest for each training task is more than hundreds of terabytes. Moreover, complex preprocessing logic is applied to ensure that data is cleaned and normalized to allow efficient transfer and easy learning. These impose very high resource requirement especially on storage, network, and CPU.
Modified on	09/02/2022 13:07

Name	Applied Machine Learning at Facebook~ A Datacenter Infrastructure Perspective
Number of Coding References	24
Number of Codes Coding	3
Coverage	16.69%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	ing our work on distributed training, we have found Ethernetbased networking to be sufficient, providing near-linear scaling capability. The ability to scale close to linearly is closely related to both model size and network bandwidth. If networking bandwidth is too low such that performing parameter synchronization takes more time than performing gradient computations, the benefits of data parallelism across machines diminishes. With its 50G Ethernet NIC, our Big Basin server has allowed us to scale out training of vision models without inter-machine synchronization being a bottleneck. In all cases, the updates need to be shared with the other replicas using techniques that provide trade-offs on synchronization (every replica sees the same state), consistency (every replica generates correct updates), and performance (which scales sub-linearly), which may impact training quality. For example, the Translation service cannot currently train on large mini-batches without degrading model quality. As a counterexample, using certain hyperparameter settings, we can train our image classification models to very large mini-batches, scaling to 256+ GPUs [13]. For one of our larger workloads, data parallelism has been demonstrated to provide 4x the throughput using 5x the machine count (e.g., for a family of models that trains over 4 days, a pool of machines training 100 different models could now train 20 models per day, so training throughput drops by 20%, but the wait time for potential engineering advancement improves from four days to one day). If models become exceptionally large, model parallelism training can be employed, where the model layers are grouped and distributed to optimize for throughput with activations pipelined between machines. The optimizations might be associated with network bandwidth or latency, or balancing internal machine limitations. This increases end-to-end latency of the model, so the raw performance gain in step time is often associated with a degradation in step quality. This may further degrade model accuracy per step. The combined degradation of step accuracy may lead to an optimal amount of parallel processing. In many cases, during inference, the DNN models them- selves are designed to be run on a single machine, as partitioning the model graph among the machines can result in large amount of communication. But major services are consistently weighing the cost/benefits of scaling their models. These considerations may dictate changes in network capacity needs.
Modified on	09/02/2022 13:05

Name	Applied Machine Learning at Facebook~ A Datacenter Infrastructure Perspective
Number of Coding References	24
Number of Codes Coding	3
Coverage	16.69%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	Scaling Considerations and Distributed Training. Train- ing a neural network involves optimization of parameter weights through Stochastic Gradient Descent (SGD). This technique, used for fitting neural nets, involves iterative weight updates through assessments of small subsets (i.e., a “batch” or “mini-batch”) of labeled examples. Data parallelism involves spawning model replicas (parallel instances) to process multiple batches in parallel. Traditionally, models were trained on a single machine. Larger or deeper models can be more expressive and provide higher accuracy, although training these models may require processing more examples. Within a single machine, training performance can be maximized by increasing model replicas and employing data parallelism across GPUs. Given that the data needed for training is increasing over time, hardware limitations can result in an unacceptable increase in overall training latency and time to convergence. Distributed training is one solution for overcoming these hardware limitations and reducing latency. This is an active research area not only at Facebook, but also in the general AI research community. A common assumption is that for data parallelism across machines, a specialized interconnect is required. However, dur-
Modified on	09/02/2022 13:04

Name	Applied Machine Learning at Facebook~ A Datacenter Infrastructure Perspective
Number of Coding References	24
Number of Codes Coding	3
Coverage	16.69%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	What Happens If We Don’t Train Our Models? We analyzed three key services that leverage ML training, to ascertain the impact of being unable to perform frequent updates to the models through training, including Ads, News Feed, and Community Integrity. Our goal was to understand the implications of losing the ability to train their models for one week, one month, and six months.
Modified on	09/02/2022 13:10

Name	Applied Machine Learning at Facebook~ A Datacenter Infrastructure Perspective
Number of Coding References	24
Number of Codes Coding	3
Coverage	16.69%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	
Modified on	23/07/2022 11:38

Name	Applied Machine Learning at Facebook~ A Datacenter Infrastructure Perspective
Number of Coding References	24
Number of Codes Coding	3
Coverage	16.69%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44

Coded Text	Getting Data to the Models For many machine learning models at Facebook, success is predicated on the availability of extensive, high-quality data. The ability to rapidly process and feed these data to the training machines is important for ensuring that we have fast and efficient offline training. For sophisticated ML applications such as Ads and Feed Ranking, the amount of data to ingest for each training task is more than hundreds of terabytes. Moreover, complex preprocessing logic is applied to ensure that data is cleaned and normalized to allow efficient transfer and easy learning. These impose very high resource requirement especially on storage, network, and CPU. As a general solution, we want to decouple the data workload from the training workload. These two workloads have very different characteristics. The data workload is very complex, ad-hoc, business dependent, and changing fast. The training workload on the other hand is usually regular (e.g. GEMM), stable (there are relatively few core operations), highly optimized, and much prefers a “clean” environment (e.g., exclusive cache usage and minimal thread contention). To optimize for both, we physically isolate the different workloads to different machines. The data processing machines, aka “readers”, read the data from storage, process and condense them, and then send to the training machines aka “trainers”. The trainers, on the other hand, solely focus on executing the training options rapidly and efficiently. Both readers and trainers can be distributed to provide great flexibility and scalability. We also optimize the machine configurations for different workloads. Another important optimization metric is network usage. The data traffic generated by training can be significant and sometimes bursty. If not handled intelligently, this can easily saturate network devices and even disrupt other services. To address these concerns, we employ optimization in compression, scheduling algorithms, data/compute placement, etc. B. Leveraging Scale As a company serving users across the world, Facebook must maintain a large fleet of servers designed to handle the peak load at any given time. As seen in Figure 4, due to variations in user activity due to diurnal load and peaks during spe-
Modified on	23/07/2022 11:42

Name	Applied Machine Learning at Facebook~ A Datacenter Infrastructure Perspective
Number of Coding References	24
Number of Codes Coding	3
Coverage	16.69%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	II. MACHINE LEARNING AT FACEBOOK Machine Learning, or ML, refers to any instance where a product leverages a series of inputs to build a tuned model, and leverages that model to create a representation, a prediction, or other forms of useful signals. Figure 1 illustrates this process which consists of the following steps, executed in turn: 1) A training phase to build the model. This phase is generally performed offline. 2) An inference phase to run the trained model in production and make a (set of) real-time predictions. This phase is performed online. Training the models is done much less frequently than inference – the time scale varies, but it is generally on the order of days. Training also takes a relatively long time to complete – typically hours or days. Meanwhile, depending on the product, the online inference phase may be run tens-oftrillions of times per day, and generally needs to be performed in real time. In some cases, particularly for recommendation systems, additional training is also performed online in a continuous manner [5]. One salient feature of machine learning at Facebook is the impact of the massive amounts of data that is potentially available to train the models. The scale of this data has many implications that span the entire infrastructure stack.
Modified on	23/07/2022 11:38

Name	Applied Machine Learning at Facebook~ A Datacenter Infrastructure Perspective
Number of Coding References	24
Number of Codes Coding	3
Coverage	16.69%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	Machine Learning Models All machine learning based services use “features” (or inputs) to produce quantified outputs. Machine learning algorithms used at Facebook include Logistic Regression (LR), Support Vector Machines (SVM), Gradient Boosted Decision Trees (GBDT), and Deep Neural Networks (DNN). LR and SVM are efficient to train and use for prediction. GBDT can improve accuracy at the expense of additional computing resources [7]. DNNs are the most expressive, potentially providing the most accuracy, but utilizing the most resources (with at least an order of magnitude compute over linear models like LR and SVM). All three types correspond to models with increasing numbers of free parameters, which must be trained by optimizing predictive accuracy against labeled input examples. Among deep neural networks, there are three general classes in use: Multi-Layer Perceptrons (MLP), Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNN/LSTM). MLP networks usually operate on structured input features (often ranking), CNNs work as spatial processors (often image processing), and RNN/LSTM networks are sequence processors (often language processing). Table I shows the mapping between these ML model types and products/services. 622 Authorized licensed use limited to: UNIVERSITY OF OSLO. Downloaded on January 12,2022 at 16:28:35 UTC from IEEE Xplore. Restrictions apply. C. ML-as-a-Service Inside Facebook Several internal platforms and toolkits exist that aim to simplify the task of leveraging machine learning within Facebook products. The primary examples include FBLearner, Caffe2, and PyTorch. FBLearner is a suite of three tools, each of which focuses on different parts of the machine learning pipeline. FBLearner leverages an internal job scheduler to allocate resources and schedule jobs on a shared pool of GPUs and CPUs, as shown in Figure 1. Most of the ML training at Facebook is run through the FBLearner platform. Working together, these tools and platforms are designed to make ML engineers more productive and help them focus on algorithmic innovation. FBLearner Feature Store. The starting point for any ML modeling task is to gather and generate features. The Feature Store is essentially a catalog of several feature generators that can be used both for training and real-time prediction, and it serves as a marketplace that multiple teams can use to share and discover features. Having this list of features is a good starting point for teams starting to use ML and also to help augment existing models with new features. FBLearner Flow is Facebook’s machine learning platform for model training [8]. Flow is a pipeline management system that executes a workflow describing the steps to train and/or evaluate a model and the resources required to do so. Workflows are built out of discrete units, or operators, each of which have inputs and outputs. The connections between the operators are automatically inferred by tracing the flow of data from one operator to the next and Flow handles the scheduling and resource management to execute the workflow. Flow also has tooling for experiment management and a simple user interface which keeps track of all of the artifacts and metrics generated by each workflow execution or experiment. The user interface makes it simple to compare and manage these experiments. FBLearner Predictor is Facebook’s internal inference engine that uses the models trained in Flow to provide predictions in real time. The Predictor can be used as a multi tenancy service or as a library that can be integrated in productspecific backend services. The Predictor is used by multiple product teams at Facebook, many of which require low latency solutions. The direct integration between Flow and Predictor also helps with running online experiments and managing multiple versions of models in productions.
Modified on	23/07/2022 11:38

Name	Applied Machine Learning at Facebook~ A Datacenter Infrastructure Perspective
Number of Coding References	24
Number of Codes Coding	3
Coverage	16.69%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	Machine learning sits at the core of many essential products and services at Facebook. This paper describes the hardware and software infrastructure that supports machine learning at global scale. Facebook's machine learning workloads are extremely diverse: services require many different types of models in practice. This diversity has implications at all layers in the system stack. In addition, a sizable fraction of all data stored at Facebook flows through machine learning pipelines, presenting significant challenges in delivering data to high-performance distributed training flows. Computational requirements are also intense, leveraging both GPU and CPU platforms for training and abundant CPU capacity for real-time inference. Addressing these and other emerging challenges continues to require diverse efforts that span machine learning algorithms, software, and hardware design.
Modified on	23/07/2022 11:37

Name	Applied Machine Learning at Facebook~ A Datacenter Infrastructure Perspective
Number of Coding References	24
Number of Codes Coding	3
Coverage	16.69%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	The amount of training data leveraged during offline training varies widely by service. In nearly all cases, the training data sets are trending toward continued and sometimes dramatic growth. For instance, some services leverage millions of rows of data before the ROI degrades, while others leverage billions of rows (100s of TB) and are bounded only by resources. Scaling Considerations and Distributed Training. Train- ing a neural network involves optimization of parameter weights through Stochastic Gradient Descent (SGD). This technique, used for fitting neural nets, involves iterative weight updates through assessments of small subsets (i.e., a “batch” or “mini-batch”) of labeled examples. Data parallelism involves spawning model replicas (parallel instances) to process multiple batches in parallel. Traditionally, models were trained on a single machine. Larger or deeper models can be more expressive and provide higher accuracy, although training these models may require processing more examples. Within a single machine, training performance can be maximized by increasing model replicas and employing data parallelism across GPUs. Given that the data needed for training is increasing over time, hardware limitations can result in an unacceptable increase in overall training latency and time to convergence. Distributed training is one solution for overcoming these hardware limitations and reducing latency. This is an active research area not only at Facebook, but also in the general AI research community. A common assumption is that for data parallelism across machines, a specialized interconnect is required. However, dur- 625 Authorized licensed use limited to: UNIVERSITY OF OSLO. Downloaded on January 12,2022 at 16:28:35 UTC from IEEE Xplore. Restrictions apply. ing our work on distributed training, we have found Ethernetbased networking to be sufficient, providing near-linear scaling capability. The ability to scale close to linearly is closely related to both model size and network bandwidth. If networking bandwidth is too low such that performing parameter synchronization takes more time than performing gradient computations, the benefits of data parallelism across machines diminishes. With its 50G Ethernet NIC, our Big Basin server has allowed us to scale out training of vision models without inter-machine synchronization being a bottleneck. In all cases, the updates need to be shared with the other replicas using techniques that provide trade-offs on synchronization (every replica sees the same state), consistency (every replica generates correct updates), and performance (which scales sub-linearly), which may impact training quality. For example, the Translation service cannot currently train on large mini-batches without degrading model quality. As a counterexample, using certain hyperparameter settings, we can train our image classification models to very large mini-batches, scaling to 256+ GPUs [13]. For one of our larger workloads, data parallelism has been demonstrated to provide 4x the throughput using 5x the machine count (e.g., for a family of models that trains over 4 days, a pool of machines training 100 different models could now train 20 models per day, so training throughput drops by 20%, but the wait time for potential engineering advancement improves from four days to one day). If models become exceptionally large, model parallelism training can be employed, where the model layers are grouped and distributed to optimize for throughput with activations pipelined between machines. The optimizations might be associated with network bandwidth or latency, or balancing internal machine limitations. This increases end-to-end latency of the model, so the raw performance gain in step time is often associated with a degradation in step quality. This may further degrade model accuracy per step. The combined degradation of step accuracy may lead to an optimal amount of parallel processing. In many cases, during inference, the DNN models them- selves are designed to be run on a single machine, as partitioning the model graph among the machines can result in large amount of communication. But major services are consistently weighing the cost/benefits of scaling their models. These considerations may dictate changes in network capacity needs
Modified on	23/07/2022 11:39

Name	Applied Machine Learning at Facebook~ A Datacenter Infrastructure Perspective
Number of Coding References	24
Number of Codes Coding	3
Coverage	16.69%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	The key contributions of this paper include the following major insights about machine learning at Facebook: • Machine learning is applied pervasively across nearly all services, and computer vision represents only a small fraction of the resource requirements. • Facebook relies upon an incredibly diverse set of machine learning approaches including, but not limited to, neural networks. • Tremendous amounts of data are funneled through our machine learning pipelines, and this creates engineering and efficiency challenges far beyond the compute nodes. • Facebook currently relies heavily on CPUs for inference, and both CPUs and GPUs for training, but constantly prototypes and evaluates new hardware solutions from a performance-per-watt perspective. • The worldwide scale of people on Facebook and corresponding diurnal activity patterns result in a huge number of machines that can be harnessed for machine learning tasks such as distributed training at scale.
Modified on	23/07/2022 11:37

Name	Applied Machine Learning at Facebook~ A Datacenter Infrastructure Perspective
Number of Coding References	24
Number of Codes Coding	3
Coverage	16.69%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	
Modified on	09/02/2022 12:57

Name	Applied Machine Learning at Facebook~ A Datacenter Infrastructure Perspective
Number of Coding References	24
Number of Codes Coding	3
Coverage	16.69%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	
Modified on	09/02/2022 12:59

Name	Applied Machine Learning at Facebook~ A Datacenter Infrastructure Perspective
Number of Coding References	24
Number of Codes Coding	3
Coverage	16.69%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03

Coded Text	All machine learning based services use “features” (or inputs) to produce quantified outputs. Machine learning algorithms used at Facebook include Logistic Regression (LR), Support Vector Machines (SVM), Gradient Boosted Decision Trees (GBDT), and Deep Neural Networks (DNN). LR and SVM are efficient to train and use for prediction. GBDT can improve accuracy at the expense of additional computing resources [7]. DNNs are the most expressive, potentially providing the most accuracy, but utilizing the most resources (with at least an order of magnitude compute over linear models like LR and SVM). All three types correspond to models with increasing numbers of free parameters, which must be trained by optimizing predictive accuracy against labeled input examples. Among deep neural networks, there are three general classes in use: Multi-Layer Perceptrons (MLP), Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNN/LSTM). MLP networks usually operate on structured input features (often ranking), CNNs work as spatial processors (often image processing), and RNN/LSTM networks are sequence processors (often language processing). Table I shows the mapping between these ML model types and products/services.
Modified on	09/02/2022 12:59

Name	Applied Machine Learning at Facebook~ A Datacenter Infrastructure Perspective
Number of Coding References	24
Number of Codes Coding	3
Coverage	16.69%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	As a general solution, we want to decouple the data workload from the training workload. These two workloads have very different characteristics. The data workload is very complex, ad-hoc, business dependent, and changing fast. The training workload on the other hand is usually regular (e.g. GEMM), stable (there are relatively few core operations), highly optimized, and much prefers a “clean” environment (e.g., exclusive cache usage and minimal thread contention). To optimize for both, we physically isolate the different workloads to different machines. The data processing machines, aka “readers”, read the data from storage, process and condense them, and then send to the training machines aka “trainers”. The trainers, on the other hand, solely focus on executing the training options rapidly and efficiently. Both readers and trainers can be distributed to provide great flexibility and scalability. We also optimize the machine configurations for different workloads. Another important optimization metric is network usage. The data traffic generated by training can be significant and sometimes bursty. If not handled intelligently, this can easily saturate network devices and even disrupt other services. To address these concerns, we employ optimization in compression, scheduling algorithms, data/compute placement, etc.
Modified on	09/02/2022 13:08

Name	Applied Machine Learning at Facebook~ A Datacenter Infrastructure Perspective
Number of Coding References	24
Number of Codes Coding	3
Coverage	16.69%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03

Coded Text	cial events (e.g. regional holidays), a large pool of servers are often idle at certain periods in time. This effectively provides an enormous pool of compute resources available during the off-peak hours. A major ongoing effort explores opportunities to take advantage of these heterogeneous resources that can be allocated to various tasks in an elastic manner. For machine learning applications, this provides a prime opportunity to take advantage of distributed training mechanisms that can scale to a large number of heterogeneous resources (e.g. different CPU and GPU platforms with differing RAM allocations). The sheer scale of the number of compute resources available during these low utilization periods leads to fundamentally different distributed training approaches, imposing a few challenges. The scheduler must first balance the load properly across heterogeneous hardware, so that hosts do not have to wait for each other for synchronization.
Modified on	09/02/2022 13:09

Name	Applied Machine Learning at Facebook~ A Datacenter Infrastructure Perspective
Number of Coding References	24
Number of Codes Coding	3
Coverage	16.69%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Machine learning sits at the core of many essential products and services at Facebook. This paper describes the hardware and software infrastructure that supports machine learning at global scale. Facebook's machine learning workloads are extremely diverse: services require many different types of models in practice. This diversity has implications at all layers in the system stack. In addition, a sizable fraction of all data stored at Facebook flows through machine learning pipelines, presenting significant challenges in delivering data to high-performance distributed training flows. Computational requirements are also intense, leveraging both GPU and CPU platforms for training and abundant CPU capacity for real-time inference. Addressing these and other emerging challenges continues to require diverse efforts that span machine learning algorithms, software, and hardware design.
Modified on	09/02/2022 12:56

Name	Applied Machine Learning at Facebook~ A Datacenter Infrastructure Perspective
Number of Coding References	24
Number of Codes Coding	3
Coverage	16.69%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03

Coded Text	Open Neural Network Exchange (ONNX) [9], which is a format to represent deep learning models in a standard way to enable interoperability across different frameworks and vendor-optimized libraries. ONNX is designed as an open specification, allowing framework authors and hardware vendors to contribute to the design and to own the various converters between frameworks and libraries. We are working with these partners to make ONNX more of a living collaboration between all these tools than as an official standard. Within Facebook, we're using ONNX as primary means of transferring research models from the PyTorch environment to high-performance production environment in Caffe2. ONNX provides the ability to automatically capture and translate static parts of the models. We have an additional toolchain that facilitates transfer of dynamic graph parts from Python by either mapping them to control-flow primitives in Caffe2 or reimplementing them in C++ as custom operators.
Modified on	12/09/2022 15:37

Name	Applied Machine Learning at Facebook~ A Datacenter Infrastructure Perspective
Number of Coding References	24
Number of Codes Coding	3
Coverage	16.69%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Several internal platforms and toolkits exist that aim to sim- plify the task of leveraging machine learning within Facebook products. The primary examples include FBLearner, Caffe2, and PyTorch. FBLearner is a suite of three tools, each of which focuses on different parts of the machine learning pipeline. FBLearner leverages an internal job scheduler to allocate resources and schedule jobs on a shared pool of GPUs and CPUs, as shown in Figure 1. Most of the ML training at Facebook is run through the FBLearner platform. Working together, these tools and platforms are designed to make ML engineers more productive and help them focus on algorithmic innovation. FBLearner Feature Store. The starting point for any ML modeling task is to gather and generate features. The Feature Store is essentially a catalog of several feature generators that can be used both for training and real-time prediction, and it serves as a marketplace that multiple teams can use to share and discover features. Having this list of features is a good starting point for teams starting to use ML and also to help augment existing models with new features. FBLearner Flow is Facebook's machine learning platform for model training [8]. Flow is a pipeline management system that executes a workflow describing the steps to train and/or evaluate a model and the resources required to do so. Workflows are built out of discrete units, or operators, each of which have inputs and outputs. The connections between the operators are automatically inferred by tracing the flow of data from one operator to the next and Flow handles the scheduling and resource management to execute the workflow. Flow also has tooling for experiment management and a simple user interface which keeps track of all of the artifacts and metrics generated by each workflow execution or experiment. The user interface makes it simple to compare and manage these experiments. FBLearner Predictor is Facebook's internal inference en- gine that uses the models trained in Flow to provide predictions in real time. The Predictor can be used as a multi tenancy service or as a library that can be integrated in productspecific backend services. The Predictor is used by multiple product teams at Facebook, many of which require low latency solutions. The direct integration between Flow and Predictor also helps with running online experiments and managing multiple versions of models in productions.
Modified on	09/02/2022 13:01

Name	Applied Machine Learning at Facebook~ A Datacenter Infrastructure Perspective
Number of Coding References	24
Number of Codes Coding	3
Coverage	16.69%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	The key contributions of this paper include the following major insights about machine learning at Facebook: • Machine learning is applied pervasively across nearly all services, and computer vision represents only a small fraction of the resource requirements. • Facebook relies upon an incredibly diverse set of machine learning approaches including, but not limited to, neural networks. • Tremendous amounts of data are funneled through our machine learning pipelines, and this creates engineering and efficiency challenges far beyond the compute nodes. • Facebook currently relies heavily on CPUs for inference, and both CPUs and GPUs for training, but constantly prototypes and evaluates new hardware solutions from a performance-per-watt perspective. • The worldwide scale of people on Facebook and corresponding diurnal activity patterns result in a huge number of machines that can be harnessed for machine learning tasks such as distributed training at scale.
Modified on	09/02/2022 12:57

Name	Applying AI in Practice~ Key Challenges and Lessons Learned
Number of Coding References	15
Number of Codes Coding	3
Coverage	4.97%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	(4) Interpretability by Correction Model Approach, see Sect. 3.4; (5) Software Quality by Automated Code Analysis and Documentation Generation, see Sect. 3.5; (6) The ALOHA Toolchain for Embedded Platforms, see Sect. 3.6; (7) Human AI Teaming as Key to Human Centered AI, see Sect. 3.7.
Modified on	08/02/2022 13:55

Name	Applying AI in Practice~ Key Challenges and Lessons Learned
Number of Coding References	15
Number of Codes Coding	3
Coverage	4.97%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	Approach 1: Automated and Continuous Data Quality Assurance In times of large and volatile amounts of data, which are often generated automatically by sensors (e.g., in smart home solutions of housing units or industrial settings), it is especially important to, (i), automatically, and, (ii), continuously monitor the quality of data [22,88]. A recent study [20] shows that the continuous monitoring of data quality is only supported by very few
Modified on	08/02/2022 13:56

Name	Applying AI in Practice~ Key Challenges and Lessons Learned
Number of Coding References	15
Number of Codes Coding	3
Coverage	4.97%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	Approach 2: The Domain Adaptation Approach for Tackling Deviating Data Characteristics at Training and Test Time In [106] and [108] we introduce a novel distance measure, the so-called Centralized Moment Discrepancy (CMD), for aligning probability distributions in the context of domain adaption. Domain adaptation algorithms are designed to minimize the misclassification risk of a discriminative
Modified on	08/02/2022 13:56

Name	Applying AI in Practice~ Key Challenges and Lessons Learned
Number of Coding References	15
Number of Codes Coding	3
Coverage	4.97%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	Approach 3: Hybrid Model Design for Improving Model Accuracy by Integrating Expert Hints in Biomedical Diagnostics For diagnostics based on biomedical image analysis, image segmentation serves as a prerequisite step to extract quantitative information [70]. If, however, segmentation results are not accurate, quantitative analysis can lead to results
Modified on	08/02/2022 13:56
Name	Applying AI in Practice~ Key Challenges and Lessons Learned
Number of Coding References	15
Number of Codes Coding	3
Coverage	4.97%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	Approach 4: Interpretability by Correction Model Approach Last year, at a symposium on predictive analytics in Vienna [93], we introduced an approach to the problem of formulating interpretability of AI models for classification or regression problems [37] with a given basis model, e.g., in the context of model predictive control [32]. The basic idea is to root the problem of interpretability in the basic model by considering the contribution of the AI model as correction of this basis model and is referred to as “Before and After Correction Parameter Comparison (BAPC)”. The idea
Modified on	08/02/2022 13:57

Name	Applying AI in Practice~ Key Challenges and Lessons Learned
Number of Coding References	15
Number of Codes Coding	3
Coverage	4.97%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	Approach 5: Software Quality by Code Analysis and Automated Documentation Quality assurance measures in software engineering include, e.g., automated testing [2], static code analysis [73], system redocumentation [69], or symbolic execution [4]. These measures need to be risk-based [23,83], exploiting knowledge about system and design dependencies, business requirements, or characteristics of the applied development process.
Modified on	08/02/2022 13:57

Name	Applying AI in Practice~ Key Challenges and Lessons Learned
Number of Coding References	15
Number of Codes Coding	3
Coverage	4.97%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	Approach 6: The ALOHA Toolchain for Embedded Platforms In [66] and [65] we introduce ALOHA, an integrated tool flow that tries to make the design of deep learning (DL) applications and their porting on embedded heterogeneous architectures as simple and painless as possible. ALOHA is the result of interdisciplinary research funded by the EU13.
Modified on	08/02/2022 13:57

Name	Applying AI in Practice~ Key Challenges and Lessons Learned
Number of Coding References	15
Number of Codes Coding	3
Coverage	4.97%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	Approach 7: Human AI Teaming Approach as Key to Human Centered AI In [36], we introduce an approach for human-centered AI in working environments utilizing knowledge graphs and relational machine learning ([72,79]). This approach is currently being refined in the ongoing Austrian project Humancentred AI in digitised working environments (AI@Work). The discussion starts with a critical analysis of the limitations of current AI systems whose learning/training is restricted to predefined structured data, most vector-based with a pre-defined format.
Modified on	08/02/2022 13:57

Name	Applying AI in Practice~ Key Challenges and Lessons Learned
Number of Coding References	15
Number of Codes Coding	3
Coverage	4.97%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	Approaches, In-Progress Research and Lessons Learned In this section we discuss ongoing research facing the outlined challenges in the previous section, comprising: (1) Automated and Continuous Data Quality Assurance, see Sect. 3.1; (2) Domain Adaptation Approach for Tackling Deviating Data Characteristics at Training and Test Time, see Sect. 3.2; (3) Hybrid Model Design for Improving Model Accuracy, see Sect. 3.3;
Modified on	08/02/2022 13:55

Name	Applying AI in Practice~ Key Challenges and Lessons Learned
Number of Coding References	15
Number of Codes Coding	3
Coverage	4.97%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13
Coded Text	AI Intrinsic Challenges There are peculiarities of deep learning methods that affect the correct interpretation of the system's output and the transparency of the system's configuration. Lack of Uniqueness of Internal Configuration: First of all, in contrast to traditional engineering, there is a lack of uniqueness of internal configuration causing difficulties in model comparison. Systems based on
Modified on	08/02/2022 13:54

Name	Applying AI in Practice~ Key Challenges and Lessons Learned
Number of Coding References	15
Number of Codes Coding	3
Coverage	4.97%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13
Coded Text	AI System Engineering Challenges In a data-driven AI systems there are two equally consequential components: software code and data. However, some input data are
Modified on	08/02/2022 13:54

Name	Applying AI in Practice~ Key Challenges and Lessons Learned
Number of Coding References	15
Number of Codes Coding	3
Coverage	4.97%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13
Coded Text	In particular, data-driven AI methods such as DNNs allow data to shape models and software systems that operate them. System engineering of AI-driven software therefore faces novel challenges at all stages of the system lifecycle [51]: – Key Challenge 1: AI intrinsic challenges due to peculiarities or shortcomings of today's AI methods; in particular, current data-driven AI is characterized by: • data challenge in terms of quality assurance and procurement; • challenge to integrate expert knowledge and models; • model integrity and reproducibility challenge due to unstable performance profiles triggered by small variations in the implementation or input data (adversarial noise); – Key Challenge 2: Challenges in the process of AI system engineering ranging from requirements analysis and specification to deployment including • testing, debugging and documentation challenges; • challenge to consider the constraints of target platforms at design time; • certification and regulation challenges resulting from highly regulated target domains such as in a bio-medical laboratory setting; – Key Challenge 3: Interpretability and trust challenge in the operational environment, in particular • trust challenge in terms of lack of interpretability and transparency by opaque models; • challenge posed by ethical guideline; • acceptance challenge in terms of societal barriers to AI adoption in society, healthcare or working environments;
Modified on	08/02/2022 13:54

Name	Applying AI in Practice~ Key Challenges and Lessons Learned
Number of Coding References	15
Number of Codes Coding	3
Coverage	4.97%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13
Coded Text	Interpretability and Trust Challenge In contrast to traditional computing, AI can now perform tasks that previously only humans were able to do. As such it contains the possibility to revolutionize every aspect of our society.
Modified on	08/02/2022 13:54

Name	Applying AI in Practice~ Key Challenges and Lessons Learned
Number of Coding References	15
Number of Codes Coding	3
Coverage	4.97%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56
Coded Text	(4) Interpretability by Correction Model Approach, see Sect. 3.4; (5) Software Quality by Automated Code Analysis and Documentation Generation, see Sect. 3.5; (6) The ALOHA Toolchain for Embedded Platforms, see Sect. 3.6; (7) Human AI Teaming as Key to Human Centered AI, see Sect. 3.7.
Modified on	08/02/2022 13:56

Name	Applying AI in Practice~ Key Challenges and Lessons Learned
Number of Coding References	15
Number of Codes Coding	3
Coverage	4.97%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56
Coded Text	Approaches, In-Progress Research and Lessons Learned In this section we discuss ongoing research facing the outlined challenges in the previous section, comprising: (1) Automated and Continuous Data Quality Assurance, see Sect. 3.1; (2) Domain Adaptation Approach for Tackling Deviating Data Characteristics at Training and Test Time, see Sect. 3.2; (3) Hybrid Model Design for Improving Model Accuracy, see Sect. 3.3;
Modified on	08/02/2022 13:56

Name	ASML~ Algorithm-Agnostic Architecture for Scalable Machine Learning
Number of Coding References	6
Number of Codes Coding	2
Coverage	4.57%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	The development of easy-to-use machine-learning application frameworks has enabled the development of advanced artificial intelligence (AI) applications with only a few lines of self-explanatory code. As a result, ML-based AI is becoming approachable by mainstream developers and small businesses. However, the deployment of ML algorithms for remote high throughput ML task execution, involving complex data-processing pipelines can still be challenging, especially with respect to production ML use cases.
Modified on	08/02/2022 13:50

Name	ASML~ Algorithm-Agnostic Architecture for Scalable Machine Learning
Number of Coding References	6
Number of Codes Coding	2
Coverage	4.57%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	
Modified on	08/02/2022 13:51

Name	ASML~ Algorithm-Agnostic Architecture for Scalable Machine Learning
Number of Coding References	6
Number of Codes Coding	2
Coverage	4.57%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	
Modified on	08/02/2022 13:51

Name	ASML~ Algorithm-Agnostic Architecture for Scalable Machine Learning
Number of Coding References	6
Number of Codes Coding	2
Coverage	4.57%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	
Modified on	08/02/2022 13:51

Name	ASML~ Algorithm-Agnostic Architecture for Scalable Machine Learning
Number of Coding References	6
Number of Codes Coding	2
Coverage	4.57%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	This paper addresses the problem of remote high through- put ML task execution involving complex data-processing pipelines. It aims to cope with well-recognized challenges [23], [24] that include the deployment of ML applications in a generic and standard way through a framework that provides the necessary level of abstraction. This framework is independent from the application domain and implementation details, such as the ML algorithms and the different programming languages used for the implementation of different components within these pipelines. To implement this framework, we propose a novel Algorithm-agnostic Scalable architecture for ML applications (ASML) that combines: • Algorithm-agnostic architecture design, that enable arbitrary ML applications to be modeled • Modular design and extensible components that allow extensibility both in terms of the supported tasks and the input and output of the architecture • Highly scalable architecture multi-client and parallel execution task support, enabling SaaS deployment scenarios • Synchronous and asynchronous task execution. To the best of our knowledge, no such ML-oriented system architecture has ever been proposed, despite the emerging needs for remote artificial intelligence (AI) services in different application domains. The main contributions of this paper include: • It provides an answer to the open research question of how to design and implement an abstraction framework, suitable for the deployment of end-to-end ML pipelines in a generic and standard way. • It provides technical details and application scenarios that can be used as examples for implementation of other ML application pipelines, • It provides a performance evaluation indicating its efficiency.
Modified on	08/02/2022 13:50

Name	ASML~ Algorithm-Agnostic Architecture for Scalable Machine Learning
Number of Coding References	6
Number of Codes Coding	2
Coverage	4.57%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	To cope with this issue, in this paper we propose a novel system architecture that enables Algorithm-agnostic, Scalable ML (ASML) task execution for high throughput applications. It aims to provide an answer to the research question of how to design and implement an abstraction framework, suitable for the deployment of end-to-end ML pipelines in a generic and standard way. The proposed ASML architecture manages horizontal scaling, task scheduling, reporting, monitoring and execution of multi-client ML tasks using modular, extensible components that abstract the execution details of the underlying algorithms. Experiments in the context of obstacle detection and recognition, as well as in the context of abnormality detection in medical image streams, demonstrate its capacity for parallel, mission critical, task execution.
Modified on	08/02/2022 13:50

Name	Asset Management in Machine Learning~ State-of-research and State-of-practice
Number of Coding References	12
Number of Codes Coding	2
Coverage	8.72%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	04/02/2022 13:03
Coded Text	
Modified on	16/02/2023 08:09

Name	Asset Management in Machine Learning~ State-of-research and State-of-practice
Number of Coding References	12
Number of Codes Coding	2
Coverage	8.72%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	04/02/2022 13:03
Coded Text	
Modified on	16/02/2023 08:14

Name	Asset Management in Machine Learning~ State-of-research and State-of-practice
Number of Coding References	12
Number of Codes Coding	2
Coverage	8.72%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	04/02/2022 13:03
Coded Text	and ingesting data for learning inferences. We classify other assets under Resource as Generic. The tools often track such generic resources as arbitrary resources. For example, some subject tools such as MLFlow and Neptune provide a log_artifact method to track any arbitrary file used during an experiment. Some subject tools track resources through their metadata and do not necessarily support the actual resource type (e.g., pointer information to location and the hash of data stored locally or on remote cloud storage systems). For example, DeepDiva [2] manages datasets by providing paths to the file directory containing the datasets. Other subject tools provide at least one dedicated method to track, provide resource-type-specific support, and sometimes internally store the resources. We only consider the latter to support resources fully. As an example, the subject PolyAxon allows users to use log_dataframe to track and store a dataset asset type. Dataset: The feature Dataset is available for subjects that identify datasets as an asset type. Data is the core asset type in machine learning since its quality plays a major role in the performance of a machine learning component. Hence, the machine learning workflow stages of data collection, data transformation, feature extraction, model training, and evaluation are data-dependent. The presence of the feature Dataset implies that the subject tool supports the tracking of datasets along with experiment-associated assets to provide dataset provenance. We found that most of our subjects, such as ModelHub, Runway, Deep-water, Neptune, and Sa- cred, require users to explicitly track operations, such as pre-processing or feature engineering carried out on the datasets. The supported dataset usually ranges from database-based data to filebased data such as spreadsheets, CSVs, or streaming datasets. For example, Deep-Water [27] allows users to provide CSV-based datasets as the training or test datasets, while tools such as Vamsa [61] support Panda1 Dataframes and automatically collect information about the Dataframes from Python scripts. Model: Machine learning models are created by learning from datasets using learning algo- rithms. The latter are typically provided by machine learning development frameworks, such as TensorFlow2 and SciKit-Learn.3 The feature Model is available for the subjects that identify models as an asset type. Like datasets,most subjects support direct or indirect tracking or storing ofmodels and their metadata to support asset management operations such as provenance analysis. Support for the feature Model is required for certain operations, such as the model comparison from different experimental runs or management ofmodel evolution through different stages. Some subjects, such as ModelHub [54, 55], offer model storage and its efficient retrieval as their primary functionality, while other subjects, such as Runway [79] and ModelKB[28, 29], primarily track models and their metadata for post-experiment result analysis. Generic: Thepresenceofthe feature Generic indicates support for tracking and managing any resource files used during machine learning workflows, typically for resources required along with datasets and models. Examples include credentials for authentication in external services that host other resources. In addition, several subjects lack dedicated support for tracking Dataset or Model types; instead, they provide a “one-size-fits-all” tracking of resources. Consequently, subjects with feature Generic can track all binary files of any type that are required or generated during an experiment, without differentiating them or providing dedicated operation beyond storage. 4.1.2 Software. Traditional VCSs, such as Git, are essential source code management tools. The engineering ofmachine-learning-based systems partly involves managing the source code used to
Modified on	16/02/2023 08:10

Name	Asset Management in Machine Learning~ State-of-research and State-of-practice
Number of Coding References	12
Number of Codes Coding	2
Coverage	8.72%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	04/02/2022 13:03
Coded Text	Asset management extends to activities in practice areas, including dataset management, model management, hyper-parameter management, process execution management, and report management. Several classes ofsupporting tools are currently available for use during machine learning model development. Silva et al. [21] classify the group of supporting tools used by machine learning users into five non-exclusive categories based on their main functionality: (a) data management systems, (b) model development systems, (c) systems for the management of machine learning model lifecycle, (d) systems for themanagement ofmachine learning models, and (e) model serving systems.We briefly explain these categories and discuss their asset management capabilities in the following paragraphs. Data management. The quality ofdatasets used in a machine learning model development plays a crucial role in the model's performance. Therefore, data understanding, preparation, and validation are crucial aspects of the machine learning engineering. In this management area, tools (e.g., OrhpeusDB) focus on the machine learning lifecycle's data-oriented works and provide operations such as tracking, versioning, and provenance on dataset assets. Model development. Management tools in this area focus on model-oriented works of the machine learning lifecycle. They provide supervised and unsupervised learning methods, such as classification, regression, and clustering algorithms, to generate and evaluatemachine learning models. Themachine learning community has mainly focused on model-oriented work, as witnessed by an extensive collection of available systems, frameworks, and libraries for model development (e.g., PyTorch, Scikit-Learn, or TensorFlow). Lifecycle management. Management tools in this category focus on all the machine learning lifecycle stages and provide management support for all asset types produced during those stages. These include experiment management tools (e.g., MLFlow, Neptune) and pipeline management tools (e.g., KubeFlow). Model management. These tools provide more specific support for managing already produced machine learning models. Such support includes the efficient storage and retrieval of models, model selection, and model comparison. Model serving. Tools under this area focus on model operation. They provide efficient storage and retrieval ofmodels to support the deployment, monitoring, and serving process. They provide information on the lineage of related assets and various evaluation performances of models (e.g., ModelDB). There are overlapping asset management functionalities across the different categories described above. According to Silva et al.'s classification, experiment management tools—the focus of this study—fall under the category ofmachine-learning lifecycle management tools. Our description ofthe machine learning lifecycle, machine learning experiments, and assets management also apply to deep learning, a family ofmachine learning based on artificial neural networks [8]. In fact, some experiment management tools specialize in support for deep learning. Machine learning experiments are a core aspect ofmachine-learning development. Consequently, the rest ofthis article focuses on the experiment management tools as our subject tools because they support users with asset management support during experimentation.
Modified on	16/02/2023 08:09

Name	Asset Management in Machine Learning~ State-of-research and State-of-practice
Number of Coding References	12
Number of Codes Coding	2
Coverage	8.72%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	04/02/2022 13:03
Coded Text	implement the machine-learning operations. Users often try to balance managing assets with traditional VCSs versus using alternative approaches tailored toward machine learning workflow. The exploratory working style of data scientists and machine learning practitioners challenges the effective use of traditional VCSs to manage assets when engineering machine learning systems [6]. The pain points ofdata scientists and machine learning practitioners when versioning their source code also motivate the need for a different approach to machine-learning software asset management. The feature Software refers to implementation assets ofthe machine learning process, such as SourceCode, Parameter,and Notebook, which are typically involved in the implementation of stages of the machine-learning workflow. This feature heavily relies on the machine learning model development tools (e.g., SciKit-Learn,4 PyTorch,5 TensorFlow,6 and Keras7) that provide a collection of general machine learning techniques to users. Source Code: This feature represents support for text-based files with implementation to carry out specific machine learning operations. Managing source code (or scripts) is generally less challenging than notebook formats for functional and large-scale engineering of machine-learningbased systems because ofavailable IDEs to support code assistance, dependency management, and debugging [19]. Source code consists of text-based files, therefore, it is easily version-controlled using traditional VCSs. Consequently, about 70% of our subjects track source code via metadata, which we represent by CodeMetadata. Other tools provide an integrated source code management approach: for example, DVC builds on Git and provides new commands tailored toward managing source code along with other machine learning assets. Parameter: Hyper-parameters are parameters utilized to control the learning process of a ma- chine learning algorithm during the model training (e.g., learning rate, regularization, and tree depth). Some subjects track hyper-parameters to facilitate the analysis ofexperiment results. Some subjects (e.g., Comet, Polyaxon, and Valoh.ai) provide hyper-parameter tuning and search features to facilitate the model-oriented stages of a machine learning workflow. In addition to hyperparameters, the asset type Parameter also represents other configurable parameters that users may require to influence their machine learning workflow. Notebook: Similar to source code, notebooks contain the implementation to carry out specific machine learning tasks. Notebooks, written in multiple execution cells, are usually used for smallscale, exploratory, and experimental machine learning tasks, where it is difficult to achieve acceptable software engineering practices, such as modular design or code reuse. Notebooks (e.g., Jupyter [50]) are crucial for reproducible machine learning workflows that require literate and interactive programming. The feature Notebook indicates the support to track notebooks as an asset type. The feature Notebook is available in six state-of-practice tools, and none of the stateof-research tools provides explicit management options for notebook formats. 4.1.3 Metadata. Conventionally, metadata allows the semantic description of entities. In our context, as a sub-feature of the Asset Type, Metadata represents the descriptive and structural static information about core assets, including machine learning experiments and their associated resource assets. These include information such as name, version, and URI. We identified the subfeatures of Metadata as Experiment, DatasetMetadata, CodeMetadata,and ModelMetadata.
Modified on	16/02/2023 08:10

Name	Asset Management in Machine Learning~ State-of-research and State-of-practice
Number of Coding References	12
Number of Codes Coding	2
Coverage	8.72%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	04/02/2022 13:03
Coded Text	Machine-learning experiment management tools, an emerging class of asset management tools, aims at addressing the challenges ofmanaging machine-learning-specific assets. Examples of such tools used in practice include MLFlow, NeptuneML,and WandB, while examples from the literature include Deep-water [27], Runway [79], and ModelKB [28, 29]. These tools target practical experiment concerns, including reproducibility [7, 46, 78] and traceability [59], by providing functionalities to store, track, and version assets from a different experiment run.
Modified on	16/02/2023 08:12

Name	Asset Management in Machine Learning~ State-of-research and State-of-practice
Number of Coding References	12
Number of Codes Coding	2
Coverage	8.72%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	04/02/2022 13:03
Coded Text	Resources. Resources are commonly referred to as artifacts by many of the subject tools. We describe Resources as the asset types required as input or produced as output from a stage of the machine learning workflow (see Figure 1). We identified features Dataset, Model,and Generic as the sub-feature of Asset Type. We identified Dataset and Model as the most critical resource types. The data-oriented stages of the machine learning workflow usually require a dataset as input and output a transformed version of it. The model-oriented stages deal with models as output and input data for the training and testing stages. The DevOps stages also involve serving models
Modified on	16/02/2023 08:09

Name	Asset Management in Machine Learning~ State-of-research and State-of-practice
Number of Coding References	12
Number of Codes Coding	2
Coverage	8.72%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	04/02/2022 13:05
Coded Text	Challenges of Asset Management in Machine Learning Experiments. The challenges en- countered during machine learning experiments are often related to the lack of explicit tooling support to address experiment management concerns, including reproducibility [7, 46, 78], replicability [14, 24], traceability [59], explainability [16, 67], interpretability [17, 31], collaboration [89], and auditability [68]. In what follows, we conceptually describe different challenges of asset management for machine learning experiments. Standardized management methods. There is a lack of standardized and explicit management methods to store, version, or operate the machine learning assets. Users rely on ad hoc approaches that may limit their efficiency during model development. For example, it may be difficult to reuse or compare assets, operations, and techniques across multiple projects [3]. The varying asset type formats (e.g., different language codebases, models, data, and metrics formats) contribute to this challenge and limit knowledge transfer regarding asset management across various projects. Researchers recently started developing approaches for fostering interoperability and reuse of machine learning assets across multiple projects. For example, Mitchell et al. propose a generic framework intended to increase the transparency of models across different application contexts and stakeholders [58]. Tracking assets and operations. Tracking machine learning assets and operations, their versions, and the decisions engaged from one specific run to another are vital for effective machine learning asset management. Tracking the assets and their corresponding information serves as a foundation to support different machine learning concerns. For example, snapshots of assets, operations, and decisions should be captured per experiment run to enable support for traceability and audibility on factual questions such as: what version ofa particular asset was used for a specific experiment run? or what operations or thought processes were considered at a specific point ofan experiment? Domain-specific operations. Asset management operations should be offered at the right abstrac- tion level. Similar to how traditional software IDEs support representation and querying of code artifacts (e.g., functions, variables, and interfaces), machine learning asset management needs to offer domain-specific asset operations on abstraction levels specific to machine learning. For example, it should allow querying for data features or hyperparameters used in a specific run with model evaluation values as conditions. Such operations can be supported through dedicated artifact meta-models [37]. Managing experiment concerns. Acquiring effective methods to address experiment concerns such as reproducibility, replicability, and traceability has the potential to improve model development processes. However, there is a lack of adequate tools that explicitly support such concerns. For reproducibility, apart from dataset and code, essential experiment dependencies such as random seeds (for non-deterministic experiments) have to be systematically captured during the experiment to offer developers the possibility to reproduce experiments. Traceability is an
Modified on	16/02/2023 08:05

Name	Asset Management in Machine Learning~ State-of-research and State-of-practice
Number of Coding References	12
Number of Codes Coding	2
Coverage	8.72%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	04/02/2022 13:05
Coded Text	important concern requiring systematic tracking and querying ofmachine-learning assets to trace operational models' behavior to concrete experiments. Users and collaboration. Collaboration between multiple developers is a "soft challenge" when working on machine learning experiments. This is partly due to the lack ofsuitable tools and workflows. Currently, collaborations between machine learning developers primarily involve sharing experiment resources or resulting models and performance metrics. We believe that better tools suited for specific machine learning assets have the potential to improve collaboration opportunities for developers. For example, versioning tools that support merging and branching for different asset types, including models, can foster teamwork and asynchronous or concurrent collaboration among developers. Comparing, analyzing, and interpreting results. A great deal of machine learning result analysis requires the use of visualization for easy explanation and interpretation. Hence, the ability to effectively infer decisions and conclusions from the obtained result is comparable to explicit tools' support in analyzing and comparing such results. Users often require comparison across multiple runs to select appropriate models. Consequently, interpretability [17, 31] ofexperiment results and explainability [67] ofmodels are essential concerns that can benefit from addressing challenges in comparing, analyzing, and interpreting outcomes ofmachine learning experiments.
Modified on	16/02/2023 08:05
Name	Asset Management in Machine Learning~ State-of-research and State-of-practice
Number of Coding References	12
Number of Codes Coding	2
Coverage	8.72%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	04/02/2022 13:05
Coded Text	Machine-learning experiment management tools, an emerging class of asset management tools, aims at addressing the challenges ofmanaging machine-learning-specific assets. Examples of such tools used in practice include MLFlow, NeptuneML,and WandB, while examples from the literature include Deep-water [27], Runway [79], and ModelKB [28, 29]. These tools target practical experiment concerns, including reproducibility [7, 46, 78] and traceability [59], by providing functionalities to store, track, and version assets from a different experiment run. While these tools have become available recently, they are not fully matured yet, especially compared to their traditional counterparts. Factors affecting their maturity include the lack of interoperability across different tools, tight coupling with specific libraries, friction, and overhead incurred during usage due to required code instrumentation for tracking assets [60, 64]. It is essential to assess the support found in the current tool landscape to facilitate research and practice towards improving existing tools, developing new ones, and improving the engineering processes of machine-learning-based systems. Important questions include: What support do these tools provide to users? What are the asset types they track? What are their commonalities and variabilities?
Modified on	16/02/2023 08:06

Name	Asset Management in Machine Learning~ State-of-research and State-of-practice
Number of Coding References	12
Number of Codes Coding	2
Coverage	8.72%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	04/02/2022 13:05
Coded Text	The exploratory and experimentation-oriented nature ofmachine learning projects significantly differs from traditional software engineering. The workflow diagram in Figure 1 contains a linear progression from requirements analysis to DevOps stages; however, machine learning workflows are typically non-linear and include multiple feedback loops (indicated by the upward arrows) [3]. These feedback loops reflect the multiple experiment runs (a.k.a experiment iterations) often performed during machine learning model development. A run refers to a one-time cycle through the relevant workflow stages, often resulting in a trained model. Each run employs specific assets' versions (e.g., datasets, hyperparameters, and source code) within the solution space ofa particular task. The machine learning workflow relies on the multiple runs of trial-and-error steps due to the unpredictable nature ofmachine-learning model performance [3, 15, 88]. Consequently, experiment runs are repeatedly performed while modifying or using new assets until the process results in a model that meets a specific target objective [6]. Such modification includes adding, removing, or engineering features, changing learning algorithms, testing different hyperparameters, and using various performance evaluation metrics. The decision to perform new runs is usually based on the result analysis of a current run and its model. Maintaining the provenance of the assets and processes used during these runs is essential to address important management concerns of machine learning model development. Also, during the DevOps works (deployment, monitoring, and control of models), there is often a need to modify and make new experiment runs based on newly available data or drift corrections to ensure models stay within the target objective's course.
Modified on	16/02/2023 08:08

Name	Asset Management in Machine Learning~ State-of-research and State-of-practice
Number of Coding References	12
Number of Codes Coding	2
Coverage	8.72%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	04/02/2022 13:05
Coded Text	This need is pronounced due to many development-related challenges of machine learning components such as asset, experiment, and dependency management. Recently, many asset management tools addressing these challenges have become available. It is essential to understand the support such tools offer to facilitate research and practice on building new management tools with native supports for machine learning and software engineering assets.
Modified on	16/02/2023 08:00

Name	Asset Management in Machine Learning~ State-of-research and State-of-practice Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	AugMix~ A Simple Data Processing Method to Improve Robustness and Uncertainty
Number of Coding References	5
Number of Codes Coding	2
Coverage	10.20%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	AUGMIX AUGMIX is a data augmentation technique which improves model robustness and uncertainty estimates, and slots in easily to existing training pipelines. At a high level, AugMix is characterized by its utilization of simple augmentation operations in concert with a consistency loss. These augmentation operations are sampled stochastically and layered to produce a high diversity of augmented images. We then enforce a consistent embedding by the classifier across diverse augmentations of the same input image through the use of Jensen-Shannon divergence as a consistency loss. Mixing augmentations allows us to generate diverse transformations, which are important for inducing robustness, as a common failure mode of deep models in the arena of corruption robustness is the memorization of fixed augmentations (Vasiljevic et al., 2016; Geirhos et al., 2018). Previous methods have attempted to increase diversity by directly composing augmentation primitives in a chain, but this can cause the image to quickly degrade and drift off the data manifold, as depicted in Figure 3. Such image degradation can be mitigated and the augmentation diversity can be maintained by mixing together the results of several augmentation chains in convex combinations. A concrete account of the algorithm is given in the pseudocode below.
Modified on	28/02/2023 14:47

Name	AugMix~ A Simple Data Processing Method to Improve Robustness and Uncertainty
Number of Coding References	5
Number of Codes Coding	2
Coverage	10.20%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	Modern deep neural networks can achieve high accuracy when the training distribution and test distribution are identically distributed, but this assumption is frequently violated in practice. When the train and test distributions are mismatched, accuracy can plummet. Currently there are few techniques that improve robustness to unforeseen data shifts encountered during deployment. In this work, we propose a technique to improve the robustness and uncertainty estimates of image classifiers. We propose AUGMIX, a data processing technique that is simple to implement, adds limited computational overhead, and helps models withstand unforeseen corruptions. AUGMIX significantly improves robustness and uncertainty measures on challenging image classification benchmarks, closing the gap between previous methods and the best possible performance in some cases by more than half.
Modified on	28/02/2023 14:47

Name	AugMix~ A Simple Data Processing Method to Improve Robustness and Uncertainty
Number of Coding References	5
Number of Codes Coding	2
Coverage	10.20%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	new image without veering too far from the original. Augmentations. Our method consists of mixing the results from augmentation chains or compositions of augmentation operations. We use operations from AutoAugment. Each operation is visualized in Appendix C. Crucially, we exclude operations which overlap with ImageNet-C corruptions. In particular, we remove the contrast, color, brightness, sharpness, and Cutout operations so that our set of operations and the ImageNet-C corruptions are disjoint. In turn, we do not use any image noising nor image blurring operations so that ImageNet-C corruptions are encountered only at test time. Operations such as rotate can be realized with varying severities, like 2◦ or −15◦. For operations with varying severities, we uniformly sample the severity upon each application. Next, we randomly sample k augmentation chains, where k = 3 by default. Each augmentation chain is constructed by composing from one to three randomly selected augmentation operations. Mixing. The resulting images from these augmentation chains are combined by mixing. While we considered mixing by alpha compositing, we chose to use elementwise convex combinations for simplicity. The k-dimensional vector of convex coefficients is randomly sampled from a Dirichlet(α, . . . , α) distribution. Once these images are mixed, we use a “skip connection” to combine the result of the augmentation chain and the original image through a second random convex combination sampled from a Beta(α, α) distribution. The final image incorporates several sources of randomness from the choice of operations, the severity of these operations, the lengths of the augmentation chains, and the mixing weights. Jensen-Shannon Divergence Consistency Loss. We couple with this augmentation scheme a loss that enforces smoother neural network responses. Since the semantic content of an image is approximately preserved with AUGMIX, we should like the model to embed xorig, xaugmix1, xaugmix2 similarly. Toward this end, we minimize the Jensen-Shannon divergence among the posterior distributions of the original sample xorig and its augmented variants. That is, for porig = ˆp(y | xorig), paugmix1 = ˆp(y | xaugmix1), paugmix2 = ˆp(y|xaugmix2), we replace the original loss L with the loss L(porig, y) + λ JS(porig; paugmix1; paugmix2). (1) To interpret this loss, imagine a sample from one of the three distributions porig, paugmix1, paugmix2. The Jensen-Shannon divergence can be understood to measure the average information that the sample reveals about the identity of the distribution from which it was sampled. This loss can be computed by first obtaining M = (porig +paugmix1 +paugmix2)/3 and then computing JS(porig; paugmix1; paugmix2) = 1 3 ⌃ KL[porig‖M] + KL[paugmix1‖M] + KL[paugmix2‖M] ⌄ . (2) Unlike an arbitrary KL Divergence between porig and paugmix, the Jensen-Shannon divergence is upper bounded, in this case by the logarithm of the number of classes. Note that we could instead compute JS(porig; paugmix1), though this does not perform as well. The gain of training with JS(porig; paugmix1; paugmix2; paugmix3) is marginal. The Jensen-Shannon Consistency Loss impels to model to be stable, consistent, and insensitive across a diverse range of inputs (Bachman et al., 2014; Zheng et al., 2016; Kannan et al., 2018). Ablations are in Section 4.3 and Appendix A.
Modified on	28/02/2023 14:48

Name	AugMix~ A Simple Data Processing Method to Improve Robustness and Uncertainty
Number of Coding References	5
Number of Codes Coding	2
Coverage	10.20%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	Robustness under Data Shift. Geirhos et al. (2018) show that training against distortions can often fail to generalize to unseen distortions, as networks have a tendency to memorize properties of the specific training distortion. Vasiljevic et al. (2016) show training with various blur augmentations can fail to generalize to unseen blurs or blurs with different parameter settings. Hendrycks & Dietterich (2019) propose measuring generalization to unseen corruptions and provide benchmarks for doing so. Kang et al. (2019) construct an adversarial version of the aforementioned benchmark. Gilmer et al. (2018); Gilmer & Hendrycks (2019) argue that robustness to data shift is a pressing problem which greatly affects the reliability of real-world machine learning systems. Calibration under Data Shift. Guo et al. (2017); Nguyen & O'Connor (2015) propose metrics for determining the calibration of machine learning models. Lakshminarayanan et al. (2017) find that simply ensembling classifier predictions improves prediction calibration. Hendrycks et al. (2019a) show that pre-training can also improve calibration. Ovadia et al. (2019) demonstrate that model calibration substantially deteriorates under data shift. Data Augmentation. Data augmentation can greatly improve generalization performance. For image data, random left-right flipping and cropping are commonly used He et al. (2015). Random occlusion techniques such as Cutout can also improve accuracy on clean data (Devries & Taylor, 2017; Zhong et al., 2017). Rather than occluding a portion of an image, CutMix replaces a portion of an image with a portion of a different image (Yun et al., 2019; Takahashi et al., 2019). Mixup also uses information from two images. Rather than implanting one portion of an image inside another, Mixup produces an elementwise convex combination of two images (Zhang 2 Figure 2: Example ImageNet-C corruptions. These corruptions are encountered only at test time and not during training. ImageNet-C Corruptions
Modified on	28/02/2023 14:47

Name	AugMix~ A Simple Data Processing Method to Improve Robustness and Uncertainty
Number of Coding References	5
Number of Codes Coding	2
Coverage	10.20%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	31/01/2022 14:44
Coded Text	Small corruptions to the data distribution are enough to subvert existing classifiers, and techniques to improve corruption robustness remain few in number. Hendrycks & Dietterich (2019) show that classification error of modern models rises from 22% on the usual ImageNet test set to 64% on ImageNet-C, a test set consisting of various corruptions applied to ImageNet test images. Even methods which aim to explicitly quantify uncertainty, such as probabilistic and Bayesian neural networks, struggle under data shift, as recently demonstrated by Ovadia et al. (2019). Improving performance in this setting has been difficult. One reason is that training against corruptions only encourages networks to memorize the specific corruptions seen during training and leaves models unable to generalize to new corruptions (Vasiljevic et al., 2016; Geirhos et al., 2018). Further, networks trained on translation augmentations remain highly sensitive to images shifted by a single pixel (Gu et al., 2019; Hendrycks & Dietterich, 2019). Others have proposed aggressive data augmentation schemes (Cubuk et al., 2018), though at the cost of a computational increase. Chun et al. (2019) demonstrates that many techniques may improve clean accuracy at the cost of robustness while many techniques which improve robustness harm uncertainty, and contrariwise. In all, existing techniques have considerable trade-offs.
Modified on	28/02/2023 14:47

Name	AugMix~ A Simple Data Processing Method to Improve Robustness and Uncertainty Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Auptimizer -- an Extensible, Open-Source Framework for Hyperparameter Tuning
Number of Coding References	3
Number of Codes Coding	2
Coverage	3.11%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	03/02/2022 10:04
Coded Text	Auptimizer design goals are focused on a user-friendly in- terface. Auptimizer benefits both practitioners and researchers and its design simplifies the integration and development of HPO algorithms. Specifically, the framework design helps both users to easily use Auptimizer in their workflows and researchers to quickly implement novel HPO algorithms. To reach these goals, the Auptimizer design has fulfilled the following requirements: • Flexibility. All implemented HPO algorithms share the same interface. This enables users to switch between different algorithms without changes in the code. A pool of HPO algorithms is integrated into the Auptimizer for users to explore and for researchers to benchmark against. • Usability. Changes to existing user's code are limited to a minimal level. It reduces the friction for users to switch to the Auptimizer framework. • Scalability. Auptimizer can deploy to a pool of computing resources to automatically scale out the experiment, and users only need to specify the resource. • Extensibility. New HPO algorithms can be easily integrated into the Auptimizer framework if they followed the specified interface (see Section III-A). Auptimizer addresses a critical missing piece in the appli- cation aspect of HPO research. It provides a universal platform to develop new algorithms efficiently. More importantly, Auptimizer lowers the barriers for data scientists in adopting HPO into their practice. Its scalability helps users to train their models efficiently with all computing resources available. Switching between different HPO algorithms is simple and only needs changing the proposer name
Modified on	08/02/2022 13:48

Name	Auptimizer -- an Extensible, Open-Source Framework for Hyperparameter Tuning
Number of Coding References	3
Number of Codes Coding	2
Coverage	3.11%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	03/02/2022 10:18
Coded Text	There is no universal HPO algorithm having the best performance over all problems. Thus, trying different ones is necessary to reveal the best results and business value. However, a high adoption cost commonly prevents user from trying different algorithms. We summarize the common factors that limit the current HPO toolboxes as flexibility, usability, scalability, and extensibility: • Flexibility. It is challenging to switch between HPO algorithms, as the interfaces are dramatically different. • Usability. It is time-consuming to integrate an existing ML project into an HPO package. Often, users need to rewrite their code for a specific HPO toolbox, and resulting script cannot be used anywhere else. • Scalability. The integration with large-scale computational resources is missing and it is typically hard to scale the toolbox to a multi-node environment. • Extensibility. It is challenging to introduce a new algorithm into the existing libraries as these libraries are tightly coupled with the implemented algorithms. We summarize the comparison of representative HPO so- lutions based on the above criteria in Table I. Based on our experience in developing an in-house solution, we release an HPO framework, Auptimizer, to mitigate the above-mentioned challenges.
Modified on	08/02/2022 13:48

Name	Auptimizer -- an Extensible, Open-Source Framework for Hyperparameter Tuning
Number of Coding References	3
Number of Codes Coding	2
Coverage	3.11%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	03/02/2022 10:18
Coded Text	Tuning machine learning models at scale, especially finding the right hyperparameter values, can be difficult and time-consuming. In addition to the computational effort required, this process also requires some ancillary efforts including engineering tasks (e.g., job scheduling) as well as more mundane tasks (e.g., keeping track of the various parameters and associated results).
Modified on	08/02/2022 13:46

Name	Auto-Keras~ An Efficient Neural Architecture Search System
Number of Coding References	8
Number of Codes Coding	2
Coverage	5.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	
Modified on	08/02/2022 13:45

Name	Auto-Keras~ An Efficient Neural Architecture Search System
Number of Coding References	8
Number of Codes Coding	2
Coverage	5.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38

Coded Text	Automated Machine Learning (AutoML) has become a very important research topic with wide applications of machine learning techniques. The goal ofAutoML is to enable people with limited machine learning background knowledge to use machine learning models easily. Work has been done on automated model selection, automated hyperparameter tunning, and etc. In the context of deep
Modified on	08/02/2022 13:42

Name	Auto-Keras~ An Efficient Neural Architecture Search System
Number of Coding References	8
Number of Codes Coding	2
Coverage	5.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	Based on the proposed neural architecture search method, we developed an open-source AutoML system, namely Auto-Keras. It is named after Keras [11], which is known for its simplicity in creating neural networks. Similar to SMAC [21], TPOT [35], AutoWEKA [44], and Auto-Sklearn [15], the goal is to enable domain experts who are not familiar with machine learning technologies to use machine learning techniques easily. However, Auto-Keras is focusing on the deep learning tasks, which is different from the systems focusing on the shallow models mentioned above.
Modified on	08/02/2022 13:44

Name	Auto-Keras~ An Efficient Neural Architecture Search System
Number of Coding References	8
Number of Codes Coding	2
Coverage	5.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38

Coded Text	In addition, we have developed a widely adopted open-source AutoML system based on our proposed method, namely Auto-Keras. It is an open-source AutoML system, which can be download and installed locally. The system is carefully designed with a concise interface for people not specialized in computer programming and data science to use. To speed up the search, the workload on CPU and GPU can run in parallel. To address the issue of different GPU memory, which limits the size ofthe neural architectures, a memory adaption strategy is designed for deployment. The main contributions of the paper are as follows: • Propose an algorithm for efficient neural architecture search based on network morphism guided by Bayesian optimization. • Conduct intensive experiments on benchmark datasets to demonstrate the superior performance of the proposed method over the baseline methods. • Develop an open-source system, namely Auto-Keras, which is one of the most widely used AutoML systems.
Modified on	08/02/2022 13:44

Name	Auto-Keras~ An Efficient Neural Architecture Search System
Number of Coding References	8
Number of Codes Coding	2
Coverage	5.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	learning, neural architecture search (NAS), which aims to search for the best neural network architecture for the given learning task and dataset, has become an effective computational tool in AutoML. Unfortunately, existing NAS algorithms are usually computationally expensive. The time complexity ofNAS is O(n¯t), where n is the number of neural architectures evaluated during the search, and ¯t is the average time consumption for evaluating each of the n neural networks. Many NAS approaches, such as deep reinforcement learning [2, 37, 47, 50, 51], gradient-based methods [8, 31, 33] and evolutionary algorithms [12, 17, 30, 38, 39, 41], require a large n to reach a good performance. Moreover, many of them train each of the n neural networks from scratch, which is very slow.
Modified on	08/02/2022 13:43

Name	Auto-Keras~ An Efficient Neural Architecture Search System
Number of Coding References	8
Number of Codes Coding	2
Coverage	5.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	Neural architecture search (NAS) has been proposed to automat- ically tune deep neural networks, but existing search algorithms, e.g., NASNet [51], PNAS [29], usually suffer from expensive computational cost. Network morphism, which keeps the functionality of a neural network while changing its neural architecture, could be helpful for NAS by enabling more efficient training during the search. In this paper, we propose a novel framework enabling Bayesian optimization to guide the network morphism for efficient neural architecture search. The framework develops a neural network kernel and a tree-structured acquisition function optimization algorithm to efficiently explores the search space. Extensive experiments on real-world benchmark datasets have been done to demonstrate the superior performance of the developed framework over the state-of-the-art methods.
Modified on	08/02/2022 13:42

Name	Auto-Keras~ An Efficient Neural Architecture Search System
Number of Coding References	8
Number of Codes Coding	2
Coverage	5.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	07/02/2022 14:10
Coded Text	As we know, Bayesian optimization [40] has been widely adopted to efficiently explore black-box functions for global optimization, whose observations are expensive to obtain. For example, it has been used in hyperparameter tuning for machine learning models [3, 15, 21, 24, 40, 44], in which Bayesian optimization searches among different combinations of hyperparameters. During the search, each evaluation of a combination of hyperparameters involves an expensive process of training and testing the machine learning model, which is very similar to the NAS problem. The unique properties of Bayesian optimization motivate us to explore its capability in guiding the network morphism to reduce the number of trained neural networks n to make the search more efficient. It is non-trivial to design a Bayesian optimization method for network morphism-based NAS due to the following challenges. First, the underlying Gaussian process (GP) is traditionally used for learning probability distribution of functions in Euclidean space. To update the Bayesian optimization model with observations, the underlying GP is to be trained with the searched architectures and their performances. However, the neural network architectures are not in Euclidean space and hard to parameterize into a fixed-length vector.
Modified on	08/02/2022 13:43

Name	Auto-Keras~ An Efficient Neural Architecture Search System
Number of Coding References	8
Number of Codes Coding	2
Coverage	5.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	07/02/2022 14:10
Coded Text	Initial efforts have been devoted to making use of network mor- phism in neural architecture search [7, 13]. It is a technique to morph the architecture of a neural network but keep its functionality [10, 45]. Therefore, we are able to modify a trained neural network into a new architecture using the network morphism operations, e.g., inserting a layer or adding a skip-connection. Only a few more epochs are required to further train the new architecture towards better performance. Using network morphism would reduce the average training time ¯t in neural architecture search. The most important problem to solve for network morphism-based NAS methods is the selection of operations, which is to select an operation from the network morphism operation set to morph an existing architecture to a new one. The network morphism-based NAS methods are not efficient enough. They either require a large number of training examples [7], or inefficient in exploring the large search space [13]. How to perform efficient neural architecture search with network morphism remains a challenging problem.
Modified on	08/02/2022 13:43

Name	Automatic Unit Test Generation for Machine Learning Libraries~ How Far Are We~
Number of Coding References	21
Number of Codes Coding	3
Coverage	8.09%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	
Modified on	08/02/2022 13:34

Name	Automatic Unit Test Generation for Machine Learning Libraries~ How Far Are We~
Number of Coding References	21
Number of Codes Coding	3
Coverage	8.09%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	are close to real faults. Along this line, mutation testing has been used to evaluate the quality of existing test suites [40, 41]. In this work, we use the widely-used mutation test frame-work PITest [42] to generate mutants and measure the mutation score of a unit test suite. Note that PITest contains many different mutation operators for generating different mutants, in this work we use the default 11 operators list in its website2
Modified on	15/06/2022 10:45

Name	Automatic Unit Test Generation for Machine Learning Libraries~ How Far Are We~
Number of Coding References	21
Number of Codes Coding	3
Coverage	8.09%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Both EVOSUITE and Randoop can significantly help cover AUX and MEB, while the performance on other three categories, i.e., VB, IVB, and EX, is limited.
Modified on	08/02/2022 13:34

Name	Automatic Unit Test Generation for Machine Learning Libraries~ How Far Are We~
Number of Coding References	21
Number of Codes Coding	3
Coverage	8.09%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Combining unit test generation with data generation: Most of existing unit test generation tools (e.g., EVOSUITE and Randoop) only focus on generating test cases by selecting method call sequences and finding arguments from previouslyconstructed inputs. However, ML libraries are data-driven, to cover most IVB and some EX, test cases with special training or input data are needed. Thus, another future direction to improve unit test generation for ML libraries is combining existing test generation with test data generation together to generate method call sequences with newly generated input data.
Modified on	15/06/2022 10:56

Name	Automatic Unit Test Generation for Machine Learning Libraries~ How Far Are We~
Number of Coding References	21
Number of Codes Coding	3
Coverage	8.09%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Combining unit test generation with parameter analysis: Our manual analysis in Section IV-D shows that most of the uncovered code from VB category is caused by missing testing valid parameters. Different from general software projects, 1556 most ML libraries contain a large number of parameters.
Modified on	15/06/2022 10:55

Name	Automatic Unit Test Generation for Machine Learning Libraries~ How Far Are We~
Number of Coding References	21
Number of Codes Coding	3
Coverage	8.09%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24

Coded Text	Current unit test suite in ML libraries has lower quality regarding code coverage (on average, 34.1%) and mutation score (on average, 21.3%). In addition, the testing effort of academic-led ML libraries is unbalanced distributed and their unit test quality is significantly worse than that of community-led ML libraries.
Modified on	08/02/2022 13:34

Name	Automatic Unit Test Generation for Machine Learning Libraries~ How Far Are We~
Number of Coding References	21
Number of Codes Coding	3
Coverage	8.09%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	EVOSUITE and Randoop lead to clear improvements in code coverage and mutation score compared to the original unit test suites of ML libraries. However, on average, 45.4% code is still uncovered with the generated test cases.
Modified on	08/02/2022 13:34

Name	Automatic Unit Test Generation for Machine Learning Libraries~ How Far Are We~
Number of Coding References	21
Number of Codes Coding	3
Coverage	8.09%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Feedback-directed based approaches build test inputs incrementally, and then the newly created test inputs extend previous ones. These test inputs are executed as soon as they are created. The results collected from these executions will then be used to guide the generation of new test inputs. We use the representative feedback-directed unit test generation tool, i.e., Randoop,
Modified on	15/06/2022 10:18

Name	Automatic Unit Test Generation for Machine Learning Libraries~ How Far Are We~
Number of Coding References	21
Number of Codes Coding	3
Coverage	8.09%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Genetic algorithm is widely used as the search techniques for test generation [21]. In the genetic algorithm, randomly selected candidate solutions are evolved by applying evolutionary operators, such as mutation and crossover, resulting in new offspring individuals, with better fitness values. The widely used objective function for unit test generation is the code coverage of the generated tests [22]. In this work, we examine the typical search-based unit test generation tool, i.e., EVOSUITE.
Modified on	15/06/2022 10:18

Name	Automatic Unit Test Generation for Machine Learning Libraries~ How Far Are We~
Number of Coding References	21
Number of Codes Coding	3
Coverage	8.09%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	In this paper, we set out to investigate the effectiveness of the widely-used automatic unit test generation techniques on ML libraries. Specifically, we select five widely-used ML libraries, i.e., Weka [13], Stanford CoreNLP [14], Mallet [15], OpenNLP [16], and Mahout [17]. Additionally, to better understand ML libraries, inspired by existing studies [10, 12], we decompose a ML library into three different types of components, i.e., data process, core model, and util (Details are in Section II-B). We use two typical automatic unit test generation tools, i.e., EVOSUITE and Randoop, as the experiment objectives following prior studies [5, 6]. For our study, we first perform an empirical study on the five ML libraries to unveil the effectiveness of their current unit test suites regarding commonly applied quality metrics such as code coverage and mutation score [18]. We then apply EVOSUITE and Randoop on these ML libraries to generate unit tests and check whether EVOSUITE and Randoop could improve test effectiveness on these libraries, regarding code coverage and mutation score, by comparing the automatically generated tests against the existing manually created ones.
Modified on	08/02/2022 13:32

Name	Automatic Unit Test Generation for Machine Learning Libraries~ How Far Are We~
Number of Coding References	21
Number of Codes Coding	3
Coverage	8.09%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	In this paper, we set out to investigate the effectiveness of exist- ing unit test generation techniques on machine learning libraries. To investigate this issue, we conducted an empirical study on five widely-used machine learning libraries with two popular unit test case generation tools, i.e., EVOSUITE and Randoop. We find that (1) most of the machine learning libraries do not maintain a high-quality unit test suite regarding commonly applied quality metrics such as code coverage (on average is 34.1%) and mutation score (on average is 21.3%), (2) unit test case generation tools, i.e., EVOSUITE and Randoop, lead to clear improvements in code coverage and mutation score, however, the improvement is limited, and (3) there exist common patterns in the uncovered code across the five machine learning libraries that can be used to improve unit test case generation tasks.
Modified on	08/02/2022 13:30

Name	Automatic Unit Test Generation for Machine Learning Libraries~ How Far Are We~
Number of Coding References	21
Number of Codes Coding	3
Coverage	8.09%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Mutation testing generates program variants (i.e., mutants) for the original program under test using mechanical transformation rules (i.e., mutation operators). Each mutant is the same with the original program except for the mutated statement. A mutant is killed by a test suite if any test from the suite failed with the mutant. Mutation score is defined as the percentage of killed mutants with the total number of mutants. A higher mutation score means a better quality of a test suite. Existing studies [37, 38, 39] have shown that mutation faults
Modified on	15/06/2022 10:45

Name	Automatic Unit Test Generation for Machine Learning Libraries~ How Far Are We~
Number of Coding References	21
Number of Codes Coding	3
Coverage	8.09%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Overall, the unit test suites in ML libraries mainly focus on a subset of valid functionalities. In addition, there exists common patterns among the uncovered code of the studied ML libraries.
Modified on	08/02/2022 13:34

Name	Automatic Unit Test Generation for Machine Learning Libraries~ How Far Are We~
Number of Coding References	21
Number of Codes Coding	3
Coverage	8.09%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24

Coded Text	This paper makes the following contributions: • We conduct a comprehensive investigation of current unit test practices on five widely-used machine learning libraries. • We examine the effectiveness and usefulness of two widely-used automatic unit test generation tools on five machine learning libraries. • We identify gaps between existing automatic unit test generation techniques and unit testing practices on machine learning libraries. • We discuss general lessons learned and future directions from the application of the automatic unit test generation to machine learning libraries. The rest of this paper is organized
Modified on	08/02/2022 13:32

Name	Automatic Unit Test Generation for Machine Learning Libraries~ How Far Are We~
Number of Coding References	21
Number of Codes Coding	3
Coverage	8.09%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	To measure the quality of a unit test suite given a project, following existing work [34], we use both code coverage [35] and mutation score [36] as the metrics. Our coverage analysis was performed using JaCoCo1, which can measure instruction and branch coverage. The instruction coverage refers to Java bytecode instructions and thus is similar to statement coverage on source code. As JaCoCo's definition of branch coverage counts only branches of conditional statements, not edges in the control flow graph, we only use instruction coverage for measuring code coverage
Modified on	15/06/2022 10:44

Name	Automatic Unit Test Generation for Machine Learning Libraries~ How Far Are We~
Number of Coding References	21
Number of Codes Coding	3
Coverage	8.09%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Transferring unit test across ML libraries: Different from general software projects, most ML libraries share the same knowledge domain and often provide the same data processing steps, machine learning algorithms, and utility support services, which makes it possible to transfer unit test cases of a specific machine learning component between ML libraries, e.g., the studied Weka and MALLET share around 50 classification algorithms with the same or similar parameters.
Modified on	15/06/2022 10:56

Name	Automatic Unit Test Generation for Machine Learning Libraries~ How Far Are We~
Number of Coding References	21
Number of Codes Coding	3
Coverage	8.09%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	Automatic unit test generation that explores the in- put space and produces effective test cases for given programs have been studied for decades. Many unit test generation tools that can help generate unit test cases with high structural coverage over a program have been examined. However, the fact that existing test generation tools are mainly evaluated on general software programs calls into question about its practical effectiveness and usefulness for machine learning libraries, which are statistically-orientated and have fundamentally different nature and construction from general software projects.
Modified on	08/02/2022 13:30

Name	Automatic Unit Test Generation for Machine Learning Libraries~ How Far Are We~
Number of Coding References	21
Number of Codes Coding	3
Coverage	8.09%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	Current unit test suite in ML libraries has lower quality regarding code coverage (on average, 34.1%) and mutation score (on average, 21.3%). In addition, the testing effort of academic-led ML libraries is unbalanced distributed and their unit test quality is significantly worse than that of community-led ML libraries.
Modified on	08/02/2022 13:34

Name	Automatic Unit Test Generation for Machine Learning Libraries~ How Far Are We~
Number of Coding References	21
Number of Codes Coding	3
Coverage	8.09%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	Lack of testing valid behaviors (VB). A class often has multiple valid behaviors, e.g., a parameter can be assigned with different values, while only a set of valid behaviors have been tested.
Modified on	15/06/2022 10:52

Name	Automatic Unit Test Generation for Machine Learning Libraries~ How Far Are We~
Number of Coding References	21
Number of Codes Coding	3
Coverage	8.09%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	We are witnessing a wide adoption of Machine Learning (ML) models in many software systems lately. Software applications powered by ML are being used in critical sectors of our daily lives; from finance and energy, to health and transportation [9, 10, 11]. Thus, building reliable and secure ML systems has become an increasingly critical challenge for software developers. However, ML libraries are often statistically-orientated, and have fundamentally different nature and construction compared to general software projects [10, 12], which makes the usefulness of existing automatic test generation tools on them unknown.
Modified on	08/02/2022 13:31

Name	Automatic Unit Test Generation for Machine Learning Libraries~ How Far Are We~
Number of Coding References	21
Number of Codes Coding	3
Coverage	8.09%
Folder Location	Codes\\Snowballing
Parent Name	Codes\\Snowballing\\Snowballing
Created on	15/06/2022 10:58
Coded Text	To test machine learning algorithms, metamorphic testing based approaches have been proposed to infer possible “test oracle” of an algorithm to indicate what the correct output should be for different inputs [59, 60, 61, 62, 63, 64, 65]. Murphy et al. [63] discussed the properties of machine learning algorithms that may be adopted as metamorphic relations to detect implementation bugs. Along this line, Xie et al. [59] conducted the first study about leveraging metamorphic testing to test the implementations of two machine learning classification algorithms, i.e., KNN, and Naive Bayes. Recently, many studies have been conducted to survey new techniques to test machine learning [66, 10, 12, 67, 68, 69, 70, 68]. Hang et al. [66] analyzed the main challenges of testing two machine learning algorithms (i.e., Naive Bayesian classifier and DNN classifier) that perform a classification task. Masuda et al. [67] examined the challenges of software quality assurance for ML-as-a-services (MLaas), which are machine learning services available through APIs on the cloud. Ishikawa et al. [69] discussed the foundational concepts 1557 that may be used in any and all machine learning testing approaches. Ma et al. [71] and Huang et al. [72] surveyed the security of deep learning models. Guo et al. [70] characterized deep learning development and deployment across different frameworks and platforms. Braiek et al. [10] and Zhang et al. [12] reviewed current existing testing practices for machine learning and deep learning frameworks. Results of the above surveys showed that most recent techniques to test machine learning mainly focused on testing deep learning models, e.g., Pei et al. [9] presented the first white-box testing of deep learning models, Tian et al. [73] proposed to utilize fuzz testing to generate more test data for autonomous driving cars, Ma et al. [74] proposed new mutation operators to evaluate the effectiveness of test cases on deep learning models, and Nejadgholi et al. [68] studied the test oracle practice in deep learning libraries.
Modified on	15/06/2022 10:58

Name	Automatically Authoring Regression Tests for Machine-Learning Based Systems
Number of Coding References	10
Number of Codes Coding	2
Coverage	6.66%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	
Modified on	08/02/2022 13:26

Name	Automatically Authoring Regression Tests for Machine-Learning Based Systems
Number of Coding References	10
Number of Codes Coding	2
Coverage	6.66%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	In this paper, we develop a new methodology aimed at functional regression testing for ML software, and apply it to a context-aware ML-based spelling checker/corrector (Speller). Our results show that the methodology can scale up the test suite to cover a large typo space, and, at the same time, reveal failure cases that can often be masked by other common misspelled inputs. Furthermore, the methodology can cover a multidimensional space with test cases automatically built upon constantly-changing production data. We show that these adaptive test suites can isolate the performance of the ML component from the end-to-end speller system, and keep up with both model and data changes.
Modified on	08/02/2022 13:25

Name	Automatically Authoring Regression Tests for Machine-Learning Based Systems
Number of Coding References	10
Number of Codes Coding	2
Coverage	6.66%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Instead of reporting individual test failures, we rely on featurizing misspell patterns and spectral clustering to automatically report subcategories of tests that contain higher proportions of defects. In particular, we make the following contributions: • We keep up with the ML software's evolving input space by automatically revising our test suite using production data to obtain new test cases and delete obsolete ones. • We learn a coverage-driven perturbation model to generalize existing cases in production data to enrich edge cases that are underrepresented in real training and test data. • We resolve the obsolete oracle problem by using the relationship between the original data from production and its perturbed counterparts; we determine the expected output of a number of test cases where the consequent feedback is positive and indicates that the users received correct outputs. • We automatically identify important failure classes by mining patterns of test cases using unsupervised learning and cluster the test cases to identify subgroups with high number of failure cases.
Modified on	08/02/2022 13:26

Name	Automatically Authoring Regression Tests for Machine-Learning Based Systems
Number of Coding References	10
Number of Codes Coding	2
Coverage	6.66%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	METHODOLOGY TO TEST ML-BASED SYSTEMS We have developed a new methodology to address the chal- lenges brought about by ML systems. The key intuition behind our methodology is to leverage the scale of production data to automatically author large numbers of coverage-adequate test cases whose Pass/Fail outcomes reveal systematic patterns that may be indicative of failures in the ML system. More specifically, as shown in Figure 1, we start by mining (A) large volumes of production data, which we then perturb using (B) a coverage-driven model-based test input curator that yields a large number of coverage-adequate test cases, with test inputs and expected outputs. These tests are then executed on the ML SUT, the actual output is obtained, and (C) an automated test oracle determines if the SUT passed or failed the test. Together with features of the inputs, test outcomes, and expected outputs, we use (D) clustering to determine which failures are related, in that all failures in a given cluster stem from a single bug. The results from clustering can also be used to improve the curator to generate more refined test suites along with oracles. We now break down our methodology workflow tailored to the Speller.
Modified on	08/02/2022 13:26

Name	Automatically Authoring Regression Tests for Machine-Learning Based Systems
Number of Coding References	10
Number of Codes Coding	2
Coverage	6.66%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	We identify four specific challenges and address them by developing a new general methodology to automatically author and maintain tests. In particular, we use the volume of production data to periodically refresh our large corpus of test inputs and expected outputs; we use perturbation of the data to obtain coverage-adequate tests; and we use clustering to help identify patterns of failures that are indicative of software bugs. We demonstrate our methodology on an ML-based context-aware Speller. Our coverage-adequate, approx. 1 million regression test cases, automatically authored and maintained for Speller (1) are virtually maintenance free, (2) detect a higher number of Speller failures than previous manually-curated tests, (3) have better coverage of previously unknown functional boundaries of the ML component, and (4) lend themselves to automatic failure triaging by clustering and prioritizing subcategories of tests with over-represented failures. We identify several systematic failure patterns which were due to previously undetected bugs in the Speller, e.g., (1) when the user misses the first letter in a short word, and (2) when the user mistakenly inserts a character in the last token of an address; these have since been fixed
Modified on	08/02/2022 13:22

Name	Automatically Authoring Regression Tests for Machine-Learning Based Systems
Number of Coding References	10
Number of Codes Coding	2
Coverage	6.66%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	End-to-end regression testing of Machine Learning (ML) software has disrupted the way we think about functional testing [1], [2]. Traditional functional tests are of the form (input, expected output, assertion()), where input is supplied to the software under test (SUT), and the test oracle (expected output and assertion()) verifies whether the SUT functioned as expected [3]. Testers strive to develop a test suite that provides adequate coverage of software features [4]. Regression testing of ML systems casts aside the 3 tra- ditional tenets of functional testing: input, expected output, and coverage in multiple ways. First, the input spaces of ML-based systems are extremely large [5] (think about all the situations to which an autonomous vehicle must react), which is why these systems are, by design, optimized for their most common inputs. Indeed, they may not always return correct outputs for all uncommon inputs. Developers may not even know all the uncommon/corner cases [6] at design time, neither would the testers during in-house test development [7]. The software's eventual functional behavior is not pre-defined; rather it emerges as it learns and evolves. Second, imperfect understanding of the input space upsets
Modified on	08/02/2022 13:23

Name	Automatically Authoring Regression Tests for Machine-Learning Based Systems
Number of Coding References	10
Number of Codes Coding	2
Coverage	6.66%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	Finally, another distinction in regression testing of ML vs. conventional systems is that individual test failures for ML systems may not be indicative of a bug. Recall that ML systems are optimized for certain classes of common inputs – they may not work for uncommon inputs; hence failures on such inputs may be perfectly acceptable. Instead, of interest to the ML-system developer are systematic test failures as well as patterns of failures that assist in software/model debugging. This shift creates new challenges for test authors, who must now create a large number of tests to reveal such patterns. Moreover, test failure triage is not always useful when looking at individual isolated failures; rather, groups of failing tests need to be examined to provide a more holistic picture of what went wrong with the ML software.
Modified on	08/02/2022 13:24

Name	Automatically Authoring Regression Tests for Machine-Learning Based Systems
Number of Coding References	10
Number of Codes Coding	2
Coverage	6.66%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	the traditional role of functional testing, which is to use functional boundaries/partitions and corner cases to ensure that the system behaves as intended within—and at the boundaries of—each partition. Consequently, test authors are unable to determine whether they have an adequate test suite that covers all functional boundaries. All their hard-coded inputs in a test suite may be distributed over an initial guesstimated set of partitions but may, over time, end up in quite another set, causing the inputs to become less important or even irrelavent. Moreover, because much of the ML decision logic is typically encoded mathematically, e.g., in a deep neural network or a logistic regression model, there is no control-flow-graph, and hence traditional coverage also does not directly apply [8]
Modified on	08/02/2022 13:24

Name	Automatically Authoring Regression Tests for Machine-Learning Based Systems
Number of Coding References	10
Number of Codes Coding	2
Coverage	6.66%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	Third, ML systems, by their very nature, are designed to better serve the most prominent inputs and constantly improve their outputs over time by learning from new training data [7]. A traditional test oracle will quickly become obsolete as its hard-coded expected output/assertion() turns stale with respect to the software's new improved output
Modified on	08/02/2022 13:24

Name	Automatically Authoring Regression Tests for Machine-Learning Based Systems
Number of Coding References	10
Number of Codes Coding	2
Coverage	6.66%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	Two key design characteristics of machine learning (ML) systems—their ever-improving nature, and learning-based emergent functional behavior—create a moving target, posing new challenges for authoring/maintaining functional regression tests.
Modified on	08/02/2022 13:21
Name	Autonomic machine learning platform
Number of Coding References	10
Number of Codes Coding	2
Coverage	11.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	
Modified on	11/02/2022 14:51

Name	Autonomic machine learning platform
Number of Coding References	10
Number of Codes Coding	2
Coverage	11.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	
Modified on	11/02/2022 14:51

Name	Autonomic machine learning platform
Number of Coding References	10
Number of Codes Coding	2
Coverage	11.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	
Modified on	11/02/2022 14:51

Name	Autonomic machine learning platform
Number of Coding References	10
Number of Codes Coding	2
Coverage	11.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	
Modified on	11/02/2022 14:52

Name	Autonomic machine learning platform
Number of Coding References	10
Number of Codes Coding	2
Coverage	11.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	
Modified on	11/02/2022 14:52

Name	Autonomic machine learning platform
Number of Coding References	10
Number of Codes Coding	2
Coverage	11.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	Acquiring information properly through machine learning requires familiarity with the available algorithms and understanding how they work and how to address the given problem in the best possible way. However, even for machine-learning experts in specific industrial fields, in order to predict and acquire information properly in different industrial fields, it is necessary to attempt several instances of trial and error to succeed with the application of machine learning. For non-experts, it is much more difficult to make accurate predictions through machine learning. In this paper, we propose an autonomic machine learning platform which provides the decision factors to be made during the developing of machine learning applications. In the proposed autonomic machine learning platform, machine learning processes are automated based on the specification of autonomic levels. This autonomic machine learning platform can be used to derive a high-quality learning result by minimizing experts' interventions and reducing the number of design selections that require expert knowledge and intuition. We also demonstrate that the proposed autonomic machine learning platform is suitable for smart cities which typically require considerable amounts of security sensitive information.
Modified on	11/02/2022 14:50

Name	Autonomic machine learning platform
Number of Coding References	10
Number of Codes Coding	2
Coverage	11.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	This study aims to present the need of the autonomic machine learning platform for the universal use of machine learning techniques in a variety of applications including Smart City, Smart Factory, and Smart Grid. This study has several unique contributions and implications, given as follow: • This study presents twelve design factors to be required by expert knowledge and intuition during the machine learning development process. • This study defines five levels of autonomic machine learning referring to as the degree of expert interventions based on the steps of the machine learning development process. • The levels of autonomic machine learning can minimize expert intervention at various autonomic levels by reducing the number of design selections that require expert knowledge and intuition is proposed. This autonomic machine learning platform can be used to derive a high-quality learning result. • This study focuses on the design issues in terms of the practical autonomic machine learning by applying the autonomic machine learning related to smart cities from an information systems perspective. Therefore, this study is useful for system developers involved in smart city development initiatives using machine learning. • In a truly smart city of the future, automation will be paramount to improve the service level of the end users (Rana et al., 2018). This capability can be derived from advanced information technologies such as the proposed autonomic machine learning platform. Like any publication, this study has certain limitations, given as follow: • This study only focuses the design of the autonomous machine learning platform, but the actual implementation or application may be very different from the proposed design structure and it does not cover issues related to implementing the autonomic machine learning techniques and systems. • As a research study, no primary data was collected or used to support the development of the proposed autonomic machine learning platform. • This study could not provide a valuable synthesis of the relevant literature by analyzing and discussing the key findings from the existing machine learning platforms on issues related to smart cities from an Information Systems Perspective.
Modified on	11/02/2022 14:52

Name	Autonomic machine learning platform
Number of Coding References	10
Number of Codes Coding	2
Coverage	11.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	07/02/2022 14:10
Coded Text	
Modified on	11/02/2022 14:51

Name	Autonomic machine learning platform
Number of Coding References	10
Number of Codes Coding	2
Coverage	11.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	07/02/2022 14:10

Coded Text	Autonomic levels In order to develop machine learning applications which work by learning target data, machine learning experts generally undertake the processes of selecting the attributes of the input data, tuning the hyperparameters, and selecting the machine learning technique and learning task. If we categorize these processes into the development steps of machine learning and if the process of each step can be performed without expert intervention, each step can then be defined as a level of autonomic machine learning. Therefore, we define five levels of autonomic machine learning referring to as the degree of expert intervention based on the development steps of machine learning. These are shown in Table 1.
Modified on	11/02/2022 14:50

Name	Autonomic machine learning platform
Number of Coding References	10
Number of Codes Coding	2
Coverage	11.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	07/02/2022 14:10
Coded Text	Autonomic machine learning level In this section, first we define autonomic machine learning based on minimizing expert intervention. We also define the autonomic levels based on the development factors of the machine learning process. 3.1. Autonomic machine learning Similar to the concept of autonomic computing, which refers to a computing framework to manage, configure, and optimize the assets of all systems while minimizing expert intervention (Autonomic Computing Strategy Perspectives, 2018), autonomic machine learning refers to autonomic machine learning with minimal expert intervention. Therefore, autonomic machine learning can be defined as the selection of an appropriate machine learning model and algorithm to achieve the desired result while autonomously detecting the
Modified on	11/02/2022 14:50

Name	AutoTrain~ An Efficient Auto-training System for Small-scale Image Classification
Number of Coding References	7
Number of Codes Coding	2
Coverage	8.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	
Modified on	08/02/2022 13:14

Name	AutoTrain~ An Efficient Auto-training System for Small-scale Image Classification
Number of Coding References	7
Number of Codes Coding	2
Coverage	8.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	AutoAugment is a method proposed by Google's Ekin D. Cubuk et al [16]. To automatically search for suitable data enhancement strategies. This method creates a search space for data enhancement strategies, and uses a reinforcement learning -based search algorithm to select specific data sets. Appropriate data enhancement strategies. In addition, the data enhancement strategies learned from one data set can be well migrated to other similar data sets. The workflow of AutoAugment is as follows:
Modified on	08/02/2022 13:13
Name	AutoTrain~ An Efficient Auto-training System for Small-scale Image Classification
Number of Coding References	7
Number of Codes Coding	2
Coverage	8.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	AutoTrain's Framework In response to the shortcomings of DeepAugment men- tioned in the previous section, we make corresponding improvements and design a more efficient model automated training system for small-scale classification—AutoTrain.
Modified on	08/02/2022 13:14

Name	AutoTrain~ An Efficient Auto-training System for Small-scale Image Classification
Number of Coding References	7
Number of Codes Coding	2
Coverage	8.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	DeepAugment is an automation tool focused on data augmentation created by Ozmen [15]. Compared with AutoAugment, DeepAugment reduces the error rate of the child model by optimizing its' architecture, the usage of random sampler on validation set solves the problem of overfitting. Instead of using reinforcement study, DeepAugment uses a Bayesian's algorithm to get the best data augmentation strategy, which is faster than AutoAugment's method. Through the above improvements, DeepAugment has 50 times faster training speed compared to AutoAugment. The workflow of AutoAugment is as follows:
Modified on	08/02/2022 13:14

Name	AutoTrain~ An Efficient Auto-training System for Small-scale Image Classification
Number of Coding References	7
Number of Codes Coding	2
Coverage	8.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	In this paper, we propose an efficient automatic training system, AutoTrain, to solve small-scale image classification problems. First, we design sample equalization scheme in data augmentation to improve the performance of training on uneven data. Then, a Bayesian optimization-based strategy controller is introduced to rapidly find the strategy applied in data augmentation. According to the label information, the AutoTrain generates different strategy sub-set for different types of images. It effectively accelerates the strategy optimization process by reducing the search space. Next, we present a dynamic adjustment model to fit tasks with different scales and complexity. We implement a modelselect module to automate model selection and adjustment. Additionally, the transfer learning and early stopping technology are used to accelerate the model training process.
Modified on	08/02/2022 13:13

Name	AutoTrain~ An Efficient Auto-training System for Small-scale Image Classification
Number of Coding References	7
Number of Codes Coding	2
Coverage	8.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	In this paper, we propose an efficient automatic training system, AutoTrain, to solve small-scale image classification problems. First, we design sample equalization in data augmentation to improve the performance of training on uneven data. Then, a Bayesian optimization-based strategy controller is introduced to rapidly find the strategy applied in data augmentation. Additionally, we present a dynamic adjustment model to fit tasks with different scales and complexity. Finally, experimental results show that the AutoTrain's training speed is about 3 times faster on average than the conventional methods. And the avergae accuracy of AutoTrain has 2% improved to the conventional methods.
Modified on	08/02/2022 13:12

Name	AutoTrain~ An Efficient Auto-training System for Small-scale Image Classification
Number of Coding References	7
Number of Codes Coding	2
Coverage	8.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	07/02/2022 14:10
Coded Text	Machine learning has become the most promising research field. However, the involved models usually require complex, tedious and expensive manual intervention. The automated machine learning technology plays a significant role in mitigating this issue. However, the current studies ignore the importance of automation in data preprocessing.
Modified on	08/02/2022 13:13

Name	Ballet~ A lightweight framework for open-source, collaborative feature engineering
Number of Coding References	5
Number of Codes Coding	2
Coverage	9.63%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	
Modified on	08/02/2022 13:10

Name	Ballet~ A lightweight framework for open-source, collaborative feature engineering
Number of Coding References	5
Number of Codes Coding	2
Coverage	9.63%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43

Coded Text	Collaborative development of open data science projects is unsupported by existing software development paradigms. We introduce Ballet, a software framework for developing new supervised learning projects with many synchronous collaborators. Feature engineering source code is submitted incrementally and validated using extensive software tests and a streaming logical feature selection algorithm. Notably, Ballet is built on the same lightweight community infrastructure as existing open-source software libraries. In a case study of predicting house prices, we show that feature source code extracted from public notebooks can be integrated using our framework to incrementally build a feature matrix without sacrificing modeling performance
Modified on	08/02/2022 13:09

Name	Ballet~ A lightweight framework for open-source, collaborative feature engineering
Number of Coding References	5
Number of Codes Coding	2
Coverage	9.63%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	In this paper, we propose a lightweight, collaborative framework for developing predictive models for open data science applications through a focus on feature engineering. Under this framework, an open-source software repository contains a curated, executable feature engineering pipeline for transforming a raw dataset into a feature matrix. Collaborators incrementally propose new features for inclusion in this pipeline as modular source code contributions. For each proposed feature, an automatic process rigorously validates it for software correctness and performs streaming logical feature selection (SLFS) to judge whether it can be merged. The resulting feature matrix can be used as the input to an automated machine learning model or a custom algorithm. Projects built on our “lightweight” framework do not require any computing infrastructure beyond that which is commonly used in open-source software, like free code hosting and CI services. We implement these ideas in the open-source Ballet framework1 and demonstrate its use in a house price prediction problem.
Modified on	08/02/2022 13:10

Name	Ballet~ A lightweight framework for open-source, collaborative feature engineering
Number of Coding References	5
Number of Codes Coding	2
Coverage	9.63%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	31/01/2022 14:44
Coded Text	Streaming feature selection. Feature selection is an important topic in the machine learning and data mining literature and has been studied extensively [11]. Interest in streaming feature selection has accelerated as big data settings have increased the dimensionality of the variable space. The grafting approach [17] uses stagewise gradient descent to alternately optimize free parameters and select new features for a regularized maximum likelihood. α-investing [25] is an adaptive complexity penalty model that bounds the false discovery rate of selecting poor features. [23] use separate statistical tests to first add relevant features to a candidate set and subsequently remove redundant ones. Similar approaches have been extended to group-wise feature value selection [22, 24]. Collaborative data science and machine learning. There has been much interest in facilitating increased collaboration in machine learning, though most existing work does not provide a structured way to ensure an effective division of labor. One exception is [20], who propose a collaborative feature engineering system in which contributors submit source code directly to a machine learning backend server. While Ballet shares several ideas, it uses a lightweight approach to integrate features — suitable for open data science projects — and improves on the feature selection techniques. In other approaches, unskilled crowd workers can be harnessed for feature engineering tasks, such as by labeling data to provide the basis for further manual feature engineering [4], or real-time editing interfaces, like that of [9, 10, 15], facilitate multiple users to edit a machine learning model specification at the same time. Collaboration can also be achieved implicitly in data science and machine learning competitions [2, 12, 14] and using networked science hubs [21]. While these have led to state-of-the-art modeling performance, there is no natural way for competitors to integrate source code components in a systematic way. Finally, large commercial data science platforms, such as Domino Data Lab4 and Dataiku5, while aiming to support all aspects of data science product infrastructure and deployment, do not explicitly provide for collaborative, incremental model development.
Modified on	23/02/2022 13:37

Name	Ballet~ A lightweight framework for open-source, collaborative feature engineering
Number of Coding References	5
Number of Codes Coding	2
Coverage	9.63%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	31/01/2022 14:44
Coded Text	These projects may attract hundreds of interested data scientists and researchers, each with unique skills and intuition. For prospective contributors to collaborate effectively at scale, there must exist a way to split data science tasks, like feature engineering, and then combine individual units of data science source code, like feature functions. Unfortunately, no such framework exists.
Modified on	08/02/2022 13:09

Name	Big Data Real Time Ingestion and Machine Learning
Number of Coding References	7
Number of Codes Coding	2
Coverage	11.83%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	Analyzing and predicting at a real time through a machine learning approach on a huge volume of heterogeneous data pool requires a novel distributed parallel computing approach-call it Big Data Online Learning.
Modified on	08/02/2022 13:02

Name	Big Data Real Time Ingestion and Machine Learning
Number of Coding References	7
Number of Codes Coding	2
Coverage	11.83%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	Data arrives in all shapes and sizes. Many time data are acquired sequentially – as an infinite ever growing stream. This real time stream data needs to be processed sequentially by taking the data source and splitting it up along temporal boundaries into finite chunks or windows. Take examples from stock market, sensors or Twitter feed data. Rather waiting for data to be collected as a whole at a long periodic interval, streaming analysis let us identify patterns – and make decisions based on them – as data start arriving. When data are nonstationary, and patterns change over time, streaming analyses adapt. At scales, where storing raw data becomes impractical, streaming analysis let us persist only smaller, more targeted representations.
Modified on	08/02/2022 13:02

Name	Big Data Real Time Ingestion and Machine Learning
Number of Coding References	7
Number of Codes Coding	2
Coverage	11.83%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	
Modified on	08/02/2022 13:05

Name	Big Data Real Time Ingestion and Machine Learning
Number of Coding References	7
Number of Codes Coding	2
Coverage	11.83%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	
Modified on	08/02/2022 13:05

Name	Big Data Real Time Ingestion and Machine Learning
Number of Coding References	7
Number of Codes Coding	2
Coverage	11.83%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Analyzing and predicting at a real time through a machine learning approach on a huge volume of heterogeneous data pool requires a novel distributed parallel computing approach-call it Big Data Online Learning.
Modified on	08/02/2022 13:02

Name	Big Data Real Time Ingestion and Machine Learning
Number of Coding References	7
Number of Codes Coding	2
Coverage	11.83%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Kafka being massively distributed client-serveroriented publisher-subscriber messaging system replaces the traditional message queue systems like Rabbit MQ, IBM MQ because of its higher throughput, reliability and replication capability [5]. It acts as a central hub for realtime processing when using along with Spark stream processing APIs [4]. 2) Apache Spark and Apache Storm: Spark and Storm are the two popular distributed stream processing computation framework. See [6][1] to learn more about Spark and Storm real time frameworks. 3) Apache Cassandra: Cassandra is a columnar NoSQL database for storge amounts of data across many commodity servers, providing high availability with no single point of failure. Refer Datastax documenation [7] for more about Cassandra. 4) Flume: Is an agent based framework which enables information gathering from multiple sources and integrate them to collate in an enterprise data lake. Flume is a high
Modified on	08/02/2022 13:04

Name	Big Data Real Time Ingestion and Machine Learning
Number of Coding References	7
Number of Codes Coding	2
Coverage	11.83%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	presentation Thus, the two-main focus area of this research are (a) the of a live streaming data ingestion and processing mechanism through Flume. Kafka and Spark Streaming framework. We take a case study of capturing click-stream data to illustrate the real time data retrieval. (b) Provide an insight into real time Big Data machine learning models like streaming regression and streaming K-means clustering.
Modified on	08/02/2022 13:02
Name	Bighead~ A Framework-Agnostic, End-to-End Machine Learning Platform
Number of Coding References	13
Number of Codes Coding	5
Coverage	11.18%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 10:07
Coded Text	
Modified on	08/02/2022 12:57

Name	Bighead~ A Framework-Agnostic, End-to-End Machine Learning Platform
Number of Coding References	13
Number of Codes Coding	5
Coverage	11.18%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 10:07
Coded Text	Many machine learning platforms have been developed at various companies. We briefly overview some major works in this section. TFX [3] is an end-to-end machine learning platform developed by Google, which spans from prototyping to production. It exclusively supports TensorFlow [7] as the model framework. Kubeflow [8] is also developed at Google, focusing on serving models in Kubernetes. MLflow [4] is developed and open sourced by Databricks. It is integrated with several cloud service providers, such as AWS and Azure. H2O [5] is a open source machine learning platform implemented in JVM with API libraries in several languages. Skymind Intelligence Layer [9], built on top of DeepLearning4J, offers model serving and scalability in its enterprise edition. Several in-house platforms cover many aspects of the machine learning workflow, such as Uber's Michelangelo [6], Facebook's FBLearner Flow [10], and Groupon's Flux [11]. However, these platforms are internal and not yet open sourced. Data Robot [12] is a popular proprietary system that offers features for automated machine learning. Several systems like Polyaxon [13], Comet [14], and Atalaya [15] provides model serving. Cloud service providers offer systems that enable the building, serving, and management of models, including Amazon's SageMaker [16], Microsoft Azure Machine Learning
Modified on	08/02/2022 12:56

Name	Bighead~ A Framework-Agnostic, End-to-End Machine Learning Platform
Number of Coding References	13
Number of Codes Coding	5
Coverage	11.18%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 19:37
Coded Text	
Modified on	08/02/2022 12:57

Name	Bighead~ A Framework-Agnostic, End-to-End Machine Learning Platform
Number of Coding References	13
Number of Codes Coding	5
Coverage	11.18%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 19:37
Coded Text	Many machine learning platforms have been developed at various companies. We briefly overview some major works in this section. TFX [3] is an end-to-end machine learning platform developed by Google, which spans from prototyping to production. It exclusively supports TensorFlow [7] as the model framework. Kubeflow [8] is also developed at Google, focusing on serving models in Kubernetes. MLflow [4] is developed and open sourced by Databricks. It is integrated with several cloud service providers, such as AWS and Azure. H2O [5] is a open source machine learning platform implemented in JVM with API libraries in several languages. Skymind Intelligence Layer [9], built on top of DeepLearning4J, offers model serving and scalability in its enterprise edition. Several in-house platforms cover many aspects of the machine learning workflow, such as Uber's Michelangelo [6], Facebook's FBLearner Flow [10], and Groupon's Flux [11]. However, these platforms are internal and not yet open sourced. Data Robot [12] is a popular proprietary system that offers features for automated machine learning. Several systems like Polyaxon [13], Comet [14], and Atalaya [15] provides model serving. Cloud service providers offer systems that enable the building, serving, and management of models, including Amazon's SageMaker [16], Microsoft Azure Machine Learning
Modified on	08/02/2022 12:56

Name	Bighead~ A Framework-Agnostic, End-to-End Machine Learning Platform
Number of Coding References	13
Number of Codes Coding	5
Coverage	11.18%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13
Coded Text	Despite rapid developments in the field, there still lacks a framework-agnostic, end-to-end machine learning platform, and existing solutions do not satisfy the needs of machine learning practitioners. First of all, many platforms lack advanced feature engineering capability, leaving many challenges unsolved in a stage in model development where many practitioners spend the majority of their time [1]. For example, it is crucial to have correct values for the features that correspond to the timestamp of the labels. This process, called temporal joins, prevents the situation of data leakage [2], that is, features incorrectly containing information on the labels because the former were observed after the latter. Another challenge is that, for features that are generated and consumed in real time, we need a framework that can process, aggregate, and join both offline and online data sources. This is not a trivial problem since aggregations and temporal joins need to be properly modeled in a principled way. Moreover, existing platforms typically focus on supporting only one model framework, often leading to tight coupling between the modeling layer and the infrastructure layer. This limits the options for practitioners when they build models, and can prevent cutting-edge algorithms and techniques from being explored and adopted. It also creates a lock-in with certain frameworks and makes migrations difficult when these frameworks evolve or get deprecated. Apart from the drawbacks of existing systems, we have identified the following four major overarching challenges when building a well designed machine learning platform. First, it is common for model developers to spend a non- trivial amount of effort to iterate on models and take them to production. Cleaning up the code, writing applications to serve the model, and thoroughly testing changes are frequent tasks. In some cases, developers even have to re-implement
Modified on	08/02/2022 12:54

Name	Bighead~ A Framework-Agnostic, End-to-End Machine Learning Platform
Number of Coding References	13
Number of Codes Coding	5
Coverage	11.18%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13
Coded Text	Second, the domain of machine learning is highly hetero- geneous and ever-changing. Models using certain algorithms are typically built on structured data, and state-of-the-art deep learning models can leverage unstructured data such as texts, images, and videos, each of which require unique processing. Meanwhile, algorithms, frameworks, and platforms are constantly being released and updated. New compute resources such as GPUs and TPUs are increasingly required. For such a platform to be useful, it needs to be versatile by supporting major frameworks and various compute resources, being flexible to accommodate frequent changes, and being extensible to allow future additions. To achieve these goals, the platform needs to decouple infrastructure from the model frameworks and provide proper abstractions. Third, models are moved across a diverse set of environments throughout their lifecycle. These environments can differ in numerous aspects, such as hardware, operating systems, versions of software dependencies, and sources of data. For example, the production environment is often vastly different from the prototyping environment. Data used for offline training often comes from a different source from online inference. Consequently, data produced by the model in production can be inconsistent with that produced during prototyping, leading to undesired situations such as incorrect results. It is therefore important to guarantee that the models are developed and productionized in a consistent setting and produce consistent results. Fourth, the scales of the datasets, throughput, latency requirements, etc. all vary drastically from model to model, and can fluctuate greatly over time. A modern convolution neural network can easily consume thousands of times more resources than a simple regression model. A fraud detection model may require sub-second latency, whereas a sales forecasting model may only need to run once per month. While having as much computing power as possible is one way to solve the scaling problem, cost adds constraints on how many resources can be deployed at a time. The ability to scale horizontally and elastically in response to the change of the workload is thus critical to the stability, reliability, and cost effectiveness of the platform.
Modified on	08/02/2022 12:55

Name	Bighead~ A Framework-Agnostic, End-to-End Machine Learning Platform
Number of Coding References	13
Number of Codes Coding	5
Coverage	11.18%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13
Coded Text	We found that these platforms do not meet the need by the machine learning community for a framework-agnostic, endto-end platform, for several reasons. First, many of them do not cover the end-to-end workflow. In particular, an important feature that most platforms lack is the integration with feature engineering and management, which is considered by some to be the most crucial part of machine learning [1]. As mentioned in Section I, there exist many challenging problems pertaining to this stage that a platform needs to solve. Second, existing platforms focus on the support of one machine learning framework, thus not giving first-class support for or even precluding the use of others. Moreover, most of the frameworks are not designed in a flexible way, and substantial work would be required for customized features, such as integration with a particular organization's data warehouse, or enforcement of data privacy policies. Lastly, some platforms are proprietary, and while they might have a more complete coverage for the workflow or popular frameworks, they cannot be leveraged by other organizations. For the above reasons, we decided to build Bighead on our own, while leveraging existing open source technologies as much as possible, such as Apache Spark [19], Apache Flink [20], Apache Airflow [21], and Kubernetes [22]. Rather than stitching separately developed components together, Bighead
Modified on	08/02/2022 12:56

Name	Bighead~ A Framework-Agnostic, End-to-End Machine Learning Platform
Number of Coding References	13
Number of Codes Coding	5
Coverage	11.18%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13
Coded Text	With the increasing need to build systems and products powered by machine learning inside organizations, it is critical to have a platform that provides machine learning practitioners with a unified environment to easily prototype, deploy, and maintain their models at scale. However, due to the diversity of machine learning libraries, the inconsistency between environments, and various scalability requirement, there is no existing work to date that addresses all of these challenges.
Modified on	08/02/2022 12:52

Name	Bighead~ A Framework-Agnostic, End-to-End Machine Learning Platform
Number of Coding References	13
Number of Codes Coding	5
Coverage	11.18%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	A. Design Goals To address the challenges in the domain of machine learning outlined in the previous sections, Bighead is designed to be: Seamless: Offer a streamlined user experience from prototyping to production, across different frameworks. Enable users to easily write, iterate on, and deploy readable, robust, and reproducible machine learning models. Provide integration with existing data infrastructure and make data access and storage straightforward. Versatile: Support all major machine learning frameworks, and make it easy to add support for new ones. Adhere to varying requirements, e.g. online and offline workloads, scheduled and ad hoc tasks, large data sizes, tight service level agreements, GPU computing, etc. Consistent: Write once and use the same data sources, transformations, and environments to produce the same results in prototyping and production, training and inference, offline and online. Scalable: Scale horizontally and elastically in response to the workload to handle changing requirements while being cost-effective.
Modified on	08/02/2022 12:58
Name	Bighead~ A Framework-Agnostic, End-to-End Machine Learning Platform
Number of Coding References	13
Number of Codes Coding	5
Coverage	11.18%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	Here, we introduce Bighead, a framework-agnostic, end-to- end platform for machine learning. It offers a seamless user experience requiring only minimal efforts that span feature set management, prototyping, training, batch (offline) inference, realtime (online) inference, evaluation, and model lifecycle management. In contrast to existing platforms, it is designed to be highly versatile and extensible, and supports all major machine learning frameworks, rather than focusing on one particular framework. It ensures consistency across different environments and stages of the model lifecycle, as well as across data sources and transformations. It scales horizontally and elastically in response to the workload such as dataset size and throughput. Its components include a feature management framework, a model development toolkit, a lifecycle management service with UI, an offline training and inference engine, an online inference service, an interactive prototyping environment, and a Docker image customization tool. It is the first platform to offer a feature management component that is a general-purpose aggregation framework with lambda architecture and temporal joins. Bighead is deployed and widely adopted at Airbnb, and has enabled the data science and engineering teams to develop and deploy machine learning models in a timely and reliable manner. Bighead has shortened the time to deploy a new model from months to days, ensured the stability of the models in production, facilitated adoption of cutting-edge models, and enabled advanced machine learning based product features of the Airbnb platform. We present two use cases of productionizing models of computer vision and natural language processing.
Modified on	08/02/2022 12:53

Name	Bighead~ A Framework-Agnostic, End-to-End Machine Learning Platform
Number of Coding References	13
Number of Codes Coding	5
Coverage	11.18%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	
Modified on	08/02/2022 12:58

Name	Bighead~ A Framework-Agnostic, End-to-End Machine Learning Platform
Number of Coding References	13
Number of Codes Coding	5
Coverage	11.18%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03

Coded Text	A. Design Goals To address the challenges in the domain of machine learning outlined in the previous sections, Bighead is designed to be: Seamless: Offer a streamlined user experience from prototyping to production, across different frameworks. Enable users to easily write, iterate on, and deploy readable, robust, and reproducible machine learning models. Provide integration with existing data infrastructure and make data access and storage straightforward. Versatile: Support all major machine learning frameworks, and make it easy to add support for new ones. Adhere to varying requirements, e.g. online and offline workloads, scheduled and ad hoc tasks, large data sizes, tight service level agreements, GPU computing, etc. Consistent: Write once and use the same data sources, transformations, and environments to produce the same results in prototyping and production, training and inference, offline and online. Scalable: Scale horizontally and elastically in response to the workload to handle changing requirements while being cost-effective.
Modified on	08/02/2022 12:58

Name	Bighead~ A Framework-Agnostic, End-to-End Machine Learning Platform
Number of Coding References	13
Number of Codes Coding	5
Coverage	11.18%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Here, we introduce Bighead, a framework-agnostic, end-to- end platform for machine learning. It offers a seamless user experience requiring only minimal efforts that span feature set management, prototyping, training, batch (offline) inference, realtime (online) inference, evaluation, and model lifecycle management. In contrast to existing platforms, it is designed to be highly versatile and extensible, and supports all major machine learning frameworks, rather than focusing on one particular framework. It ensures consistency across different environments and stages of the model lifecycle, as well as across data sources and transformations. It scales horizontally and elastically in response to the workload such as dataset size and throughput. Its components include a feature management framework, a model development toolkit, a lifecycle management service with UI, an offline training and inference engine, an online inference service, an interactive prototyping environment, and a Docker image customization tool. It is the first platform to offer a feature management component that is a general-purpose aggregation framework with lambda architecture and temporal joins. Bighead is deployed and widely adopted at Airbnb, and has enabled the data science and engineering teams to develop and deploy machine learning models in a timely and reliable manner. Bighead has shortened the time to deploy a new model from months to days, ensured the stability of the models in production, facilitated adoption of cutting-edge models, and enabled advanced machine learning based product features of the Airbnb platform. We present two use cases of productionizing models of computer vision and natural language processing.
Modified on	08/02/2022 12:53

Name	Building Responsibility in AI~ Transparent AI for Highly Automated Vehicle Systems
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Callisto~ Entropy-based Test Generation and Data Quality Assessment for Machine Learning Systems
Number of Coding References	6
Number of Codes Coding	1
Coverage	4.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	CALLISTO employs techniques to quickly and efficiently identify the inputs in the datasets for test generation. This aids the user to discover erroneous behaviours without extensive testing of the entire dataset. We illustrate the intuition behind CALLISTO's test generation approach in Figure 1. Consider a metamorphic transformation M (e.g. rotating a picture by a small amount) and inputs A1, A2 and A. A rudimentary approach would be to apply M to all the data points leading to large computational overheads. CALLISTO aims to discover points like A which will allow users to selectively apply a metamorphic relation to inputs which are likely to cause errors. CALLISTO will avoid points such as A1 and A2.
Modified on	08/02/2022 12:45

Name	Callisto~ Entropy-based Test Generation and Data Quality Assessment for Machine Learning Systems
Number of Coding References	6
Number of Codes Coding	1
Coverage	4.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Our CALLISTO approach aims to identify the mislabelled and/or low quality inputs for further intervention. The size of these identified sets should be significantly smaller in comparison to the full dataset and the cost of human intervention is minimal. This approach is first seen in identifying wrong
Modified on	08/02/2022 12:45

Name	Callisto~ Entropy-based Test Generation and Data Quality Assessment for Machine Learning Systems
Number of Coding References	6
Number of Codes Coding	1
Coverage	4.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	RQ1.1: Is the test generation effective? To evaluate the efficiency of this research question, we compute the ratio of erroneous inputs obtained via metamorphic transformations. We say a transformed input is an error when the prediction of the model does not match the corresponding label. In our evaluation, we picked four transformations, namely panning, 2D rotation, affine and perspective. These transformations can be seen in Figure 4. It has been shown that Machine Learning models are not robust to even simple
Modified on	08/02/2022 12:46

Name	Callisto~ Entropy-based Test Generation and Data Quality Assessment for Machine Learning Systems
Number of Coding References	6
Number of Codes Coding	1
Coverage	4.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	RQ1.2: How do the error rates vary with Shannon index? 0 To answer this research question, we evaluated the error rates for the Fashion MNIST as seen in Figure 5. Intuitively, the error rates should increase with an increase in the Shannon Threshold. This is because the model outputs that have a high Shannon index can be understood as being less confident in their prediction and being more brittle. Thus the respective inputs are more prone to errors due to transformations.
Modified on	08/02/2022 12:46

Name	Callisto~ Entropy-based Test Generation and Data Quality Assessment for Machine Learning Systems
Number of Coding References	6
Number of Codes Coding	1
Coverage	4.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	RQ2: Is low quality data effectively identified? The usage of the Shannon diversity to identify mislabelled data was first proposed to detect mislabelled training instances [16]. Our solution to detect low quality data (i.e. Algorithm 2) is similar, but we conduct a thorough user study to validate our results.
Modified on	08/02/2022 12:46

Name	Callisto~ Entropy-based Test Generation and Data Quality Assessment for Machine Learning Systems
Number of Coding References	6
Number of Codes Coding	1
Coverage	4.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	To alleviate this issue, we have developed CALLISTO, a test generation and data quality analysis tool. CALLISTO leverages the entropy of the outputs of the ML classifiers to quantify the uncertainty in the prediction of these classifiers. To further elucidate our motivation to research this technique, we sketch two situations that are likely to occur in the future. We also show these problems that may crop up because of the widespread proliferation of ML and then we sketch a sample solution for these problems that uses our CALLISTO technique
Modified on	08/02/2022 12:44

Name	Cats are not fish~ deep learning testing calls for out-of-distribution awareness
Number of Coding References	18
Number of Codes Coding	4
Coverage	10.12%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	Although recent progress has been made in designing novel testing techniques for DL software, which can detect thousands of errors, the current state-of-the-art DL testing techniques usually do not take the distribution of generated test data into consideration. It is therefore hard to judge whether the ”identified errors” are indeed meaningful errors to the DL application (i.e., due to quality issues of the model) or outliers that cannot be handled by the current model (i.e., due to the lack of training data). Tofill this gap, we take thefi rst step and conduct a large scale empirical study, with a total of 451 experiment configurations, 42 deep neural networks (DNNs) and 1.2 million test data instances, to investigate and characterize the impact of OOD-awareness on DL testing. We further analyze the consequences when DL systems go into production by evaluating the effectiveness of adversarial retraining with distribution-aware errors. The results confirm that introducing data distribution awareness in both testing and enhancement phases outperforms distribution unaware retraining by up to 21.5%.
Modified on	08/02/2022 12:36

Name	Cats are not fish~ deep learning testing calls for out-of-distribution awareness
Number of Coding References	18
Number of Codes Coding	4
Coverage	10.12%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	• OOD Detection for DL Testing (RQ1). In DL testing, it is still challenging to distinguish ID and OOD data especially when more similarities between the two tested data types exist. Therefore,fi ne-grained thresholds seem helpful in gaining a better understanding in similar cases. Our results in Fig. 2 provide the following guidance: if the testing tool aims at generating ID test cases, a smaller N should be selected. If we want to generate OOD test cases, a larger N should be selected. Research Guidance: a possible direction is to develop OOD techniques, which can effectively detectfi ne-grained OOD data for deep learning testing. • Mutation Operators and Coverage Criteria (RQ2&3). Our results show that the existing mutation and coverage criteria have different effects on ID data or OOD data generation. To build the distribution-aware DL testing tools, we could develop distribution-based coverage criteria that canfi lter some OOD data or ID data. Research Guidance: DL testing tools should be aware of distribution. A promising direction is to develop morefi ne-grained distribution-aware criteria for the test selection. • Robustness Enhancement (RQ4.) Our initial results have shown that distribution-aware retraining is more effective in robustness enhancement than the distribution-unaware retraining. It seems that root causes for ID errors are partially model dependent while OOD errors can be effectivelyfi xed with new training data. Research Guidance: A future research direction is to further analyze the root cause of ID and OOD errors, especially in an even morefi ne grained setting which can provide guidance for repairing the model from a data and DNN architecture perspective under regard of the presented threshold of this work.
Modified on	08/02/2022 12:42

Name	Cats are not fish~ deep learning testing calls for out-of-distribution awareness
Number of Coding References	18
Number of Codes Coding	4
Coverage	10.12%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Although recent progress has been made in designing novel testing techniques for DL software, which can detect thousands of errors, the current state-of-the-art DL testing techniques usually do not take the distribution of generated test data into consideration. It is therefore hard to judge whether the ”identified errors” are indeed meaningful errors to the DL application (i.e., due to quality issues of the model) or outliers that cannot be handled by the current model (i.e., due to the lack of training data). Tofill this gap, we take thefi rst step and conduct a large scale empirical study, with a total of 451 experiment configurations, 42 deep neural networks (DNNs) and 1.2 million test data instances, to investigate and characterize the impact of OOD-awareness on DL testing. We further analyze the consequences when DL systems go into production by evaluating the effectiveness of adversarial retraining with distribution-aware errors. The results confirm that introducing data distribution awareness in both testing and enhancement phases outperforms distribution unaware retraining by up to 21.5%.
Modified on	08/02/2022 12:36
Name	Cats are not fish~ deep learning testing calls for out-of-distribution awareness
Number of Coding References	18
Number of Codes Coding	4
Coverage	10.12%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Answer to RQ2: The data distribution generated by mutation operators is dependent on the datasets. Considering the same mutation operators, more test cases tend to be more OOD for grayscale images and less for color images. Image blur and Image Scale are the mutations strategies where the highest OOD-score is observed, whereas Image Rotation, Shear, Brightness and Contrast generate fewer OOD data. The error test cases are more likely to be OOD than benign test cases.
Modified on	08/02/2022 12:42

Name	Cats are not fish~ deep learning testing calls for out-of-distribution awareness
Number of Coding References	18
Number of Codes Coding	4
Coverage	10.12%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Answer to RQ3: Our results show that, existing coverage criteria affect the data distribution of generated test cases, which is important to address when designing a test scenario. KMNC, TKNC, NC and FANN tend to decrease the number of OOD benign test cases while NC and NBC tend to increase the OOD benign test cases. For the mutation operators that tend to generate fewer OOD data such as rotation and contrast, the existing coverage criteria can increase the number of OOD data by covering more behaviors of the DNN. For the mutation that tends to generate more OOD data such as blur, the existing coverage criteria can decrease the number byfiltering some data with the coverage guidance. For grayscale images, the coverage criteria may decrease the number ofOOD data with random mutation operators. The coverage criteria may increase the OOD data for generated error test cases.
Modified on	08/02/2022 12:42
Name	Cats are not fish~ deep learning testing calls for out-of-distribution awareness
Number of Coding References	18
Number of Codes Coding	4
Coverage	10.12%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Answer to RQ4: The results demonstrate that ID-errors tend to befi xed via DNN adjustments, while OOD-errors seem to require further training data for being correctly classified. When retraining, OOD errors tend to be on average 10.4% more effective in improving the robustness of the DNN than ID errors or randomly chosen ones. Furthermore, not all OOD errors help the model to generalize, indicating that the OOD-score distance towards the trained/tested DNN distribution matters when choosing the right data for enhancing robustness.
Modified on	08/02/2022 12:42

Name	Cats are not fish~ deep learning testing calls for out-of-distribution awareness
Number of Coding References	18
Number of Codes Coding	4
Coverage	10.12%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Overall, our results show that Outlier Exposure on Densenet-121 architecture performs the best and the results are consistent on all benchmark datasets. The existing techniques can detect the ID data effectively where most of the test data are correctly classified as in-distribution. Splitting the classes of the training set imposes a challenge to the detection techniques and grants a new perspective on their performance for application-realistic settings.
Modified on	08/02/2022 12:41
Name	Cats are not fish~ deep learning testing calls for out-of-distribution awareness
Number of Coding References	18
Number of Codes Coding	4
Coverage	10.12%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	shows the overview of our study that focuses on the data distribution and its effect on test cases generated by the coverage guided testing (CGT). Specifically, we focus on the three main components ofthe CGT for DL: 1) the effect ofmutation on data distribution of the test cases, 2) the effect of coverage criteria and 3) the effectiveness of the output test cases on robustness enhancement. To perform the study, we select three widely used datasets (i.e., MNIST, CIFAR-10 and FashionMNIST [18, 21, 53]) andfi ve state-ofthe-art OOD detection techniques (i.e., Baseline [15], ODIN [24], Mahalanobis [23], Outlier Exposure [16] and Likelihood-Ratio [40]). These OOD detection techniques are mainly proposed to distinguish two totally different datasets (e.g., CIFAR-10 and MNIST). However, in this work, the generated test cases are often similar to the training data. Therefore, wefi rst design an experiment to investigate the effectiveness of existing OOD techniques in a novel and more challenging scenario where the difference between datasets for comparison is low (i.e., RQ1). Based on the results of RQ1, we select the best OOD metric to evaluate the relationship between the data distribution and the mutation operators. In this work, we select the datasets in the image classification domain. Hence, we select 8 popular image transformations, which are mainly used in the existing CGT tools (e.g., DeepTest [48], DeepHunter [55], and TensorFuzz [33]). Then, we study which mutators tend to generate ID data and which ones tend to generate OOD data (i.e., RQ2). Next, we evaluate the relationship between the data distribution and the coverage criteria guidance in CGT. We select 6 popular testing criteria [25, 33, 35] to study which coverage criteria are more likely to guide the generation of OOD or ID data (i.e., RQ3). Adversarial training is a common way to enhance the robustness of DNNs by including the detected error data during training. Therefore, wefi nally study the possible root cause for ID and OOD errors and study the effectiveness of the OOD and ID data for DNN robustness enhancement (i.e., RQ4).
Modified on	21/06/2022 16:29

Name	Cats are not fish~ deep learning testing calls for out-of-distribution awareness
Number of Coding References	18
Number of Codes Coding	4
Coverage	10.12%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	We select 5 state-of-the-art OOD-detection techniques that are commonly used among related literature [4, 16, 23, 24, 36, 37, 40]. OOD techniques use different approaches to retrieve an OOD score. Some use input perturbation, and others require a specifically trained new DNN. Therefore, this work includes techniques with various approaches as follows: • Simple Baseline [15]. The baseline identifies that in and outof-distribution samples are classified with different probability distributions. The softmax prediction probability is used to determine whether an input is ID or OOD. • ODIN [24]. In addition to calculate the softmax prediction probability proposed by the baseline, ODIN adds temperature scaling to the input as well as small input perturbations. They show that small perturbations have stronger effects on in-distribution samples rather than out-of-distribution samples, achieving higher ID/OOD classification performance. • Mahalanobis [23]. Mahalanobis detection technique integrates the information from all layers into the score calculation. It takes the closest class for each layer, adds small noise to the test sample andfi nally computes the score by measuring the Mahalanobis distance [29] between the test sample and the closest class-conditional Gaussian distribution. • Outlier Exposure [16]. Outlier Exposure stands out by classifying inputs with a separately trained DNN which is exposed to the same training data as the DNN used for the application. However, in addition, out-of-distribution data is integrated into the training procedure of the outlier exposure DNN model. Afterward, the maximum softmax probability is taken similar to the baseline for out-of-distribution detection. • Likelihood-Ratio [40]. The latest contribution of thefi eld utilizes a separately trained DNN, namely a generative DNN model with PixelCNN++ architecture [38]. They use an estimate of input complexity to derive a parameter-free OOD score, which can be seen as a likelihood-ratio [40].
Modified on	08/02/2022 12:41

Name	Cats are not fish~ deep learning testing calls for out-of-distribution awareness
Number of Coding References	18
Number of Codes Coding	4
Coverage	10.12%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	As Deep Learning (DL) is continuously adopted in many industrial applications, its quality and reliability start to raise concerns. Similar to the traditional software development process, testing the DL software to uncover its defects at an early stage is an effective way to reduce risks after deployment. According to the fundamental assumption of deep learning, the DL software does not provide statistical guarantee and has limited capability in handling data that falls outside of its learned distribution, i.e., out-of-distribution (OOD) data.
Modified on	08/02/2022 12:35

Name	Cats are not fish~ deep learning testing calls for out-of-distribution awareness
Number of Coding References	18
Number of Codes Coding	4
Coverage	10.12%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	However, different from traditional software whose decision logic is mostly programmed by the developer, deep learning adopts a data-driven programming paradigm. In particular, the major tasks of a DL developer are preparing the training data, labeling the data, programming the architecture of the deep neural network (DNN), and specifying the training configuration. All the decision logic is automatically learned during the runtime training phase and encoded in the obtained DNN (e.g., by weights, bias, and their combinations). Due to the differences of programming paradigm, the logic encoding format, and the tasks that a DNN is often developed for (e.g., image recognition), testing techniques for traditional software cannot be directly applied and new testing techniques are needed for DNNs. While some recent progress has been made in proposing novel testing criteria [17, 25, 33, 35] and test generation techniques for quality assurance of DNNs [8, 33, 35, 43, 48, 55, 58], it still lacks interpretation and understanding on the detected errors by such techniques and their impact. For example, it is not clear whether errors are indeed caused by missing training data or insufficient training, etc. The fundamental assumption of deep learning is that
Modified on	08/02/2022 12:38
Name	Cats are not fish~ deep learning testing calls for out-of-distribution awareness
Number of Coding References	18
Number of Codes Coding	4
Coverage	10.12%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	To summarize, this paper makes the following contributions: • We perform a large scale empirical study on how deep learning testing affects the data distribution of the generated test cases; and how distribution aware testing influences DNN model robustness. • Our study identifies the impact of mutation operators and coverage criteria on the distribution of the generated test cases. We find that image rotation, contrast and brightness tend to generate more ID data while image blur is more likely to generate OOD data. In terms of the coverage criteria, NBC and SNAC facilitate to generate more OOD data than others. • We demonstrate the effectiveness of distribution aware retrain- ing, outperforming the state-of-the-art by up to 21.5%. Based on our results, we provide guidelines on distribution-aware error selection for robustness enhancement, by studying the effect of root cause of ID and OOD errors.
Modified on	08/02/2022 12:40

Name	Cats are not fish~ deep learning testing calls for out-of-distribution awareness
Number of Coding References	18
Number of Codes Coding	4
Coverage	10.12%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	
Modified on	08/02/2022 12:40

Name	Cats are not fish~ deep learning testing calls for out-of-distribution awareness
Number of Coding References	18
Number of Codes Coding	4
Coverage	10.12%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14

Coded Text	• AUROC. Given an unknown input, OOD detection techniques need to identify a threshold to classify it as ID or OOD. The area under the receiver operating characteristic curve (AUROC) [14] is usually used to evaluate the performance of a classification method across multiple thresholds. The AUROC can be thought of as the probability that an anomalous example is given a higher OOD score than an in-distribution example [16]. Thus, the higher AUROC, the better the OOD detector. • TPRN, which is the true positive rate at N% true negative rate (TPRN). We regard OOD data as the positive class. First, we use N% true negative rate to select one threshold for the OOD detector. Then, with this threshold, we evaluate the true positive rate of the detector. Note that, for the parameter N in TPRN, a larger N means we select a bigger threshold such that more data is perceived under the threshold as ID (i.e., higher true negative rate). Thus, a larger N provides more confident measurement for detecting OOD data while a smallerN provides more confident measurement for detecting ID data.
Modified on	08/02/2022 12:41

Name	Cats are not fish~ deep learning testing calls for out-of-distribution awareness
Number of Coding References	18
Number of Codes Coding	4
Coverage	10.12%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	As Deep Learning (DL) is continuously adopted in many industrial applications, its quality and reliability start to raise concerns. Similar to the traditional software development process, testing the DL software to uncover its defects at an early stage is an effective way to reduce risks after deployment. According to the fundamental assumption of deep learning, the DL software does not provide statistical guarantee and has limited capability in handling data that falls outside of its learned distribution, i.e., out-of-distribution (OOD) data.
Modified on	08/02/2022 12:35

Name	Cats are not fish~ deep learning testing calls for out-of-distribution awareness
Number of Coding References	18
Number of Codes Coding	4
Coverage	10.12%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	However, different from traditional software whose decision logic is mostly programmed by the developer, deep learning adopts a data-driven programming paradigm. In particular, the major tasks of a DL developer are preparing the training data, labeling the data, programming the architecture of the deep neural network (DNN), and specifying the training configuration. All the decision logic is automatically learned during the runtime training phase and encoded in the obtained DNN (e.g., by weights, bias, and their combinations). Due to the differences of programming paradigm, the logic encoding format, and the tasks that a DNN is often developed for (e.g., image recognition), testing techniques for traditional software cannot be directly applied and new testing techniques are needed for DNNs. While some recent progress has been made in proposing novel testing criteria [17, 25, 33, 35] and test generation techniques for quality assurance of DNNs [8, 33, 35, 43, 48, 55, 58], it still lacks interpretation and understanding on the detected errors by such techniques and their impact. For example, it is not clear whether errors are indeed caused by missing training data or insufficient training, etc. The fundamental assumption of deep learning is that
Modified on	08/02/2022 12:38

Name	Cats are not fish~ deep learning testing calls for out-of-distribution awareness
Number of Coding References	18
Number of Codes Coding	4
Coverage	10.12%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	If the new unseen input data has a similar distribution as the training data, deep learning provides some statistical guarantee on its prediction correctness in terms of accuracy. However, if the new input data does not follow the training data (i.e., out-of-distribution (OOD)), deep learning does not provide statistical guarantee on its prediction. For example, if a DNN is only trained on cat and dog data for binary classification, given an input data offi sh, the DNN can still produce a prediction result. However, this input data does not follow the distribution of cat and dog data. Hence, handling the fish data goes beyond the capability of this DNN and should not be considered as valid input. Intuitively, erroneous inputs that follow the distribution of train- ing data may reveal the real weakness of the DNN since the DNN is expected to handle such data. On the other hand, input errors that are considered out-of-distribution may either inherit new information benefitting generalization as well as a domain shift or are simply irrelevant to the DL application. Thereby, the root cause of an error may be identified through analyzing its distribution behavior, which makes us rethink how to define errors and how to test the DNN by considering the effect on its trained distribution.
Modified on	08/02/2022 12:38

Name	Cats are not fish~ deep learning testing calls for out-of-distribution awareness
Number of Coding References	18
Number of Codes Coding	4
Coverage	10.12%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	To summarize, this paper makes the following contributions: • We perform a large scale empirical study on how deep learning testing affects the data distribution of the generated test cases; and how distribution aware testing influences DNN model robustness. • Our study identifies the impact of mutation operators and coverage criteria on the distribution of the generated test cases. We find that image rotation, contrast and brightness tend to generate more ID data while image blur is more likely to generate OOD data. In terms of the coverage criteria, NBC and SNAC facilitate to generate more OOD data than others. • We demonstrate the effectiveness of distribution aware retrain- ing, outperforming the state-of-the-art by up to 21.5%. Based on our results, we provide guidelines on distribution-aware error selection for robustness enhancement, by studying the effect of root cause of ID and OOD errors.
Modified on	08/02/2022 12:40

Name	Challenges in Deploying Machine Learning~ a Survey of Case Studies
Number of Coding References	13
Number of Codes Coding	9
Coverage	23.87%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 19:37
Coded Text	In this study, we undertake a survey of these reports to capture the current challenges in deploying machine learning in production1. First, we provide an overview of the machine learning deployment workflow. Second, we review use case studies to extract problems and concerns practitioners have at each particular deployment stage. Third, we discuss cross-cutting aspects that affect every stage of the deployment workflow: ethical considerations, end users' trust and security.
Modified on	07/02/2022 23:37

Name	Challenges in Deploying Machine Learning~ a Survey of Case Studies
Number of Coding References	13
Number of Codes Coding	9
Coverage	23.87%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	04/02/2022 13:05
Coded Text	[22]. Firebird is an advisory system in the Atlanta Fire Department, that helps identify priority targets for fire inspections. As a first step towards developing Firebird data was collected from 12 datasets including history of fire incidents, business licenses, households and more. These datasets were combined to give a single dataset covering all relevant aspects of each property monitored by the Fire Department. Authors particularly highlight data joining as a difficult problem. Given the fact that building location is the best way to identify that building, each dataset contained spatial data specifying building's address. Spatial information can be presented in different formats, and sometimes contain minor differences such as different spellings. All this needed to be cleaned and corrected, and turned out to be a massive effort that consumed a lot of time. 3.3 Data augmentation There are multiple reasons why data might need to be augmented, and in practice one of the most problematic ones is the absence of labels. Real-world data is often unlabeled, thus labeling turns out to be a challenge in its own right. We discuss three possible factors for lack of labeled data: limited access to experts, absence of high-variance data, and sheer volume. Labels assignment is difficult in environments that tend to generate large volumes of data, such as network traffic analysis. To illustrate a scale of this volume, a single 1-GB/s Ethernet interface can deliver up to 1.5 million packets per second. Even with a huge downsampling rate this is still a significant number, and each sampled packet needs to be traced in order to be labeled. This problem is described by Pacheco et al. [23], which surveys applications of machine learning to network traffic classification, with tasks such as protocol identification or attack detection. There are two main ways of acquiring data in this domain, and both are complicated for labeling purposes: • Uncontrolled, collecting real traffic. This approach requires complex tracking flows belonging to a specific application. Due to this complexity very few works implement reliable ground truth assignment for real traffic. • Controlled, emulating or generating traffic. Even in this case it has been shown that existing tools for label assignment introduce errors into collected ML datasets, going as high as almost 100% for certain applications. Moreover, these tools' performance degrades severely for encrypted traffic. Access to experts can be another bottleneck for collecting high-quality labels. It is particularly true for areas where expertise mandated by the labeling process is significant, such as medical image analysis [24]. Normally multiple experts are asked to label a set of images, and then these labels are aggregated to ensure quality. This is rarely feasible for big datasets due to experts' availability. A possible option here is to use noisy label oracles or weaker annotations, however these approaches are by their definition a trade off that provides imprecise labels, which ultimately leads to a severe loss in quality of the model. Such losses are unacceptable in healthcare industry, where even the smallest deviation can cause catastrophic results (this is known as The Final Percent challenge). Lack of access to high-variance data can be among the main challenges one faces when deploying machine learning solution from lab environment to real world. Dulac-Arnold et al. [25] explain that this is the case for Reinforcement Leaning (RL). It is common practice in RL research to have access to separate environments for training and evaluation of an agent. However, in practice all data comes from the real system, and the agent can no longer have a separate exploration policy - this is simply unsafe. Therefore the data available becomes low-variance. While this approach ensures safety, it means that agent is not trained to recognize an unsafe situation and make right decision in it. A practical example of this issue is the goal specification for autonomous vehicles [26]. 3.4 Data analysis Data needs to be analyzed in order to uncover potential biases or unexpected distributions in it. Availability of high quality tools is essential for conducting any kind of data analysis. One area that practitioners find particularly challenging in that regard is visualization for data profiling [27]. Data profiling refers to all activities associated with troubleshooting data quality, such as missing values, inconsistent data types and verification of assumptions. Despite obvious relevance to the fields of databases and statistics, there are still too few tools that enable efficient execution of these data mining tasks. The need for such

	tools becomes apparent considering that, according to the 5 survey conducted by Microsoft [28], data scientists think about data issues as the main reason to doubt quality of the overall work.
Modified on	22/06/2022 11:52
Name	Challenges in Deploying Machine Learning~ a Survey of Case Studies
Number of Coding References	13
Number of Codes Coding	9
Coverage	23.87%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	04/02/2022 13:05
Coded Text	Data is an integral part of any machine learning solution. Overall effectiveness of the solution depends on the training and test data as much as on the algorithm. The process of creating quality datasets is usually the very first stage in any production ML pipeline. Unsurprisingly, practitioners face a range of issues while working with data as previously reported by Polyzotis et al. [15]. Consequently, this stage consumes time and energy that is often not anticipated beforehand. In this section, we describe issues concerning four steps within data management: data collection, data preprocessing, data augmentation and data analysis. Note that we consider storage infrastructure challenges, such as setting up databases and query engines, beyond the scope of this survey. 3.1 Data collection Data collection involves activities that aim to discover and understand what data is available, as well as how to organize convenient storage for it. The task of discovering what data exists and where it is can be a challenge by itself, especially in large production environments. Finding data sources and understanding their structure is a major task, which may prevent data scientists from even getting started on the actual application development. As explained by Lin and Ryaboy [16], at Twitter this situation often happened due to the fact that the same entity (e.g. a Twitter user) is processed by multiple services. Internally Twitter consists of multiple services calling each other, and every service is responsible for a single operation. This approach, known in software engineering as “single responsibility principle” [17], results in an architecture that is very flexible in terms of scalability and modification. However, the flip side of this approach is that at a large scale it is impossible to keep track of what data related to the entity is being stored by which service, and in which form. Besides, some data may only exist in a form of logs, which by their nature are not easily parsed or queried. An even worse case is the situation when data is not stored anywhere, and in order to build a dataset one needs to generate synthetic service API calls. Such a dispersion of data creates major hurdles for data scientists, because without a clear idea of what data is available or can be obtained it is often impossible to understand what ML solutions can realistically achieve. 3.2 Data preprocessing The preprocessing step normally involves a range of data cleaning activities: imputation of missing values, reduction of data into an ordered and simplified form, and mapping from raw form into a more convenient format. Methods for carrying out data manipulations like this is an area of research that goes beyond the scope of this study. We encourage readers to refer to review papers on the topic, such as Abedjan et al. [18], Patil and Kulkarni [19], Ridzuan et al. [20]. An approach towards classifying readiness of data for ML tasks is given by Lawrence [21]. A lesser known but also important problem that can also be considered an object of the preprocessing step is data dispersion. It often turns out that there can be multiple relevant separate data sources which may have different schemas, different conventions, and their own way of storing and accessing the data. Joining this information into a single dataset suitable for machine learning can be a complicated task on its own right. An example of this is what developers of Firebird faced
Modified on	22/06/2022 11:51

Name	Challenges in Deploying Machine Learning~ a Survey of Case Studies
Number of Coding References	13
Number of Codes Coding	9
Coverage	23.87%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	03/02/2022 10:24
Coded Text	
Modified on	07/02/2022 23:40

Name	Challenges in Deploying Machine Learning~ a Survey of Case Studies
Number of Coding References	13
Number of Codes Coding	9
Coverage	23.87%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	03/02/2022 10:24
Coded Text	In recent years, machine learning has received increased interest both as an academic research field and as a solution for real-world business problems. However, the deployment of machine learning models in production systems can present a number of issues and concerns.
Modified on	07/02/2022 23:36

Name	Challenges in Deploying Machine Learning~ a Survey of Case Studies
Number of Coding References	13
Number of Codes Coding	9
Coverage	23.87%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	03/02/2022 10:24
Coded Text	This shift comes with challenges. Just as with any other field, there are significant differences between what works in academic setting and what is required by a real world system. Certain bottlenecks and invalidated assumptions should always be expected in the course of that process. As more solutions are developed and deployed, practitioners sometimes report their experience in various forms, including publications and blog posts
Modified on	07/02/2022 23:37
Name	Challenges in Deploying Machine Learning~ a Survey of Case Studies
Number of Coding References	13
Number of Codes Coding	9
Coverage	23.87%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	04/02/2022 21:53
Coded Text	Monitoring is one of the issues associated with maintaining machine learning systems as reported by Sculley et al. [52]. The community is in the early stages of understanding what are the key metrics of data and models to monitor and how to alarm on them. Monitoring of evolving input data, prediction bias and overall performance of ML models is an open problem. Another maintenance issue highlighted by this paper that is specific to data-driven decision making is feedback loops. ML models in production can influence their own behavior over time via regular retraining. While making sure the model stays up to date, it is possible to create feedback loop where the input to the model is being adjusted to influence its behavior. This can be done intentionally, as well as happen inadvertently which is a unique challenge when running live ML systems. Klaise et al. [53] point out the importance of outlier detection as a key instrument to flag model predictions that cannot be used in a production setting. The authors name two reasons for such predictions to occur: the inability of the models to generalize outside of the training dataset and also overconfident predictions on out-of-distribution instances due to poor calibration. Deployment of the outlier detector can be a challenge in its own right, because labeled outlier data is scarce, and the detector training often becomes a semi-supervised or even an unsupervised problem. Additional insight on monitoring of ML systems can be found in Ackermann et al. [54]. This paper describes an early intervention system (EIS) for two police departments in the US. On the surface their monitoring objectives seem completely standard: data integrity checks, anomaly detection and performance metrics. One would expect to be able to use out-of-the-box tooling for these tasks. However, the authors explain that they had to build all these checks from scratch in order to maintain good model performance. For instance, the data integrity check meant verifying updates of a certain input table and checksums on historical records, performance metric was defined in terms of the number of changes in top k outputs, and anomalies were tracked on rank-order correlations over time. All of these monitoring tools required considerable investigation and implementation. This experience report highlights a common problem with currently available end-to-end ML platforms: the final ML solutions are usually so sensitive to problem's specifics that out-of-the-box tooling does not fit their needs well. As a final remark we note that there is an overlap between choice of metrics for monitoring and validation. The latter topic is discussed in Section 5.1.
Modified on	24/02/2022 10:50

Name	Challenges in Deploying Machine Learning~ a Survey of Case Studies
Number of Coding References	13
Number of Codes Coding	9
Coverage	23.87%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 11:04
Coded Text	5 Model Verification The goal of the model verification stage is multifaceted, because an ML model should generalize well to unseen inputs, demonstrate reasonable handling of edge cases and overall robustness, as well as satisfy all functional requirements. In this section, we discuss issues concerning three steps within model verification: requirement encoding, formal verification and test-based verification. 7 5.1 Requirement encoding Defining requirements for a machine learning model is a crucial prerequisite of testing activities. It often turns out that an increase in model performance does not translate into a gain in business value, as Booking.com discovered after deploying 150 models into production [44]. Therefore more specific metrics need to be defined and measured, such as KPIs and other business driven measures. In the case of Booking.comsuch metrics included conversion, customer service tickets or cancellations. Cross-disciplinary effort is needed to even define such metrics, as understanding frommodeling, engineering and business angles is required. Once defined, these metrics are used for monitoring of the production environment and for quality control of model updates. Besides, simply measuring the accuracy of the ML model is not enough to understand its performance. Essentially, performance metrics should reflect audience priorities. For instance Sato et al. [45] recommend validating models for bias and fairness, while in the case described by Wagstaff et al. [31] controlling for consumption of spacecraft resources is crucial. 5.2 Formal Verification The formal verification step verifies that the model functionality follows the requirements defined within the scope of the project. Such verification could include mathematical proofs of correctness or numerical estimates of output error bounds, but as Ashmore et. al. [14] point out this rarely happens in practice. More often quality standards are being formally set via extensive regulatory frameworks. An example of where ML solutions have to adhere to regulations is the banking industry [46]. This requirement was developed in the aftermath of the global financial crisis, as the industry realized that there was a need for heightened scrutiny towards models. As a consequence an increased level of regulatory control is now being applied to the processes that define how the models are built, approved and maintained. For instance, official guidelines has been published by the UK's Prudential Regulation Authority [47] and European Central Bank [48]. These guidelines require model risk frameworks to be in place for all business decision-making solutions, and implementation of such frameworks requires developers to have extensive tests suites in order to understand behavior of their MLmodels. The formal verification step in that context means ensuring that the model meets all criteria set by the corresponding regulations. Regulatory frameworks share similarities with country-wide policies, which we discuss in greater details in Section 7.1. 5.3 Test-based Verification Test-based verification is intended for ensuring that the model generalizes well to the previously unseen data. While collecting validation dataset is usually not a problem, as it can be derived from splitting the training dataset, it may not be enough for production deployment. In an ideal scenario testing is done in a real-life setting, where business driven metrics can be observed, as we discussed in Section 5.1. Full scale testing in real-world environment can be challenging for a variety of safety, security and scale reasons, and is often substituted with testing in simulation. That is the case for models for autonomous vehicles control [26]. Simulations are cheaper, faster to run, and provide flexibility to create situations rarely encountered in real life. Thanks to these advantages, simulations are becoming prevalent in this field. However, it is important to remember that simulation-based testing hinges on assumptions made by simulation developers, and therefore cannot be considered a full replacement for real-world testing. Even small variations between simulation and real world can have drastic effects on the system behavior, and therefore the authors conclude that validation of the model and simulation environment alone is not enough for autonomous vehicles. This point is emphasized further by the experiences from the field of

	reinforcement learning [25], where use of simulations is a de-facto standard for training agents. In addition, the dataset itself also needs to be constantly validated to ensure data errors do not creep into the pipeline and do not affect the overall quality. Breck et al. [49] argue that one of the most common scenarios when issues in data can go unnoticed is the setup where data generation is decoupled from the ML pipeline. There could be multiple reasons for such issues to appear, including bugs in code, feedback loops, changes in data dependencies. Data errors can propagate 8 and manifest themselves at different stages of the pipeline, therefore it is imperative to catch them early by including data validation routines into the ML pipeline.
Modified on	24/02/2022 11:03
Name	Challenges in Deploying Machine Learning~ a Survey of Case Studies
Number of Coding References	13
Number of Codes Coding	9
Coverage	23.87%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 20:46
Coded Text	6.3 Updating Once the initial deployment of the model is completed, it is often necessary to be able to update the model later on in order to make sure it always reflects the most recent trends in data and the environment. There are multiple techniques for adapting models to a new data, including scheduled regular retraining and continual learning [55]. Nevertheless in production setting model updating is also affected by practical considerations. A particularly important problem that directly impacts the quality and frequency of model update procedure is the concept drift. Concept drift in ML is understood as changes observed in joint distribution p(X, y), where X is the model input and y is the model output. Undetected, this phenomenon can have major adverse effects on model performance, as is shown by Jameel et al. [56] for classification problems or by Celik and Vanschoren [57] in AutoML context. Concept drift can arise due to a wide variety of reasons. For example, the finance industry faced turbulent changes as the financial crisis of 2008 was unfolding, and if advanced detection techniques were employed it could have provided additional insights into the ongoing crisis, as explained by Masegosa et al. [58]. Changes in data can also be caused by inability to avoid fluctuations in the data collection procedure, as described in paper Langenkämper et al. [59] which studies the effects of slight changes in marine images capturing gear and location on deep learning models' performance. Data shifts can have noticeable consequences even when occurring at microscopic scale, as Zenisek et al. [60] show in their research on predictive maintenance for wear and tear of industrial machinery. Even though concept drift has been known for decades [61], these examples show that it remains a critical problem for applications of ML today. On top of the question of when to retrain the model to keep it up to date, there is an infrastructural question on how to deliver the model artifact to the production environment. In software engineering such tasks are commonly solved with continuous delivery (CD), which is an approach for 10 accelerating development cycle by building an automated pipeline for building, testing and deploying software changes. CD for machine learning solutions is complicated because, unlike in regular software products where changes only happen in the code, ML solutions experience change along three axis: the code, the model and the data. An attempt to formulate CD for ML as a separate discipline can be seen in Sato et al. [45]. This work describes the pieces involved and the tools that can be used at each step of building the full pipeline. A direct illustration of benefits that a full CD pipeline can bring to the real-life ML solution can be found in Wider and Deger [62].
Modified on	24/02/2022 10:54

Name	Challenges in Deploying Machine Learning~ a Survey of Case Studies
Number of Coding References	13
Number of Codes Coding	9
Coverage	23.87%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 22:49
Coded Text	6 Model Deployment Machine learning systems running in production are complex software systems that have to be maintained over time. This presents developers with another set of challenges, some of which are shared with running regular software services, and some are unique to ML. There is a separate discipline in engineering, called DevOps, that focuses on techniques and tools required to successfully maintain and support existing production systems. Consequently, there is a necessity to apply DevOps principles to ML systems. However, even though some of the DevOps principles apply directly, there is also a number of challenges unique to productionizing machine learning. This is discussed in detail by Dang et al. [50] which uses the term AIOps for DevOps tasks for ML systems. Some of the challenges mentioned include lack of high quality telemetry data as well as no standard way to collect it, difficulty in acquiring labels which makes supervised learning approaches inapplicable3 and lack of agreed best practices around handling of machine learning models. In this section, we discuss issues concerning three steps within model deployment: integration, monitoring and updating. 6.1 Integration The model integration step constitutes of two main activities: building the infrastructure to run the model and implementing the model itself in a form that can be consumed and supported. While the former is a topic that belongs almost entirely in systems engineering and therefore lies out of scope of this work, the latter is of interest for our study, as it exposes important aspects at the intersection of ML and software engineering. In fact, many concepts that are routinely used in software engineering are now being reinvented in the ML context. Code reuse is a common topic in software engineering, and ML can benefit from adopting the same mindset. Reuse of data and models can directly translate into savings in terms of time, effort or infrastructure. An illustrative case is the approach Pinterest took towards learning image embeddings [51]. There are three models used in Pinterest internally which use similar embeddings, and initially they were maintained completely separately, in order to make it possible to iterate on the models individually. However, this created engineering challenges, as every effort in working with these embeddings had to be multiplied by three. Therefore the team decided to investigate the possibility of learning universal set of embeddings. It turned out to be possible, and this reuse ended up simplifying their deployment pipelines as well as improving performance on individual tasks. A broad selection of engineering problems that machine learning practitioners now face is given in Sculley et al. [52]. Most of them are considered anti-patterns in engineering, but are currently widespread in machine learning software. Some of these issues, such as abstraction boundaries erosion and correction cascades, are caused by the fact that ML is used in cases where the software has to take explicit dependency on external data. Others, such as glue code or pipeline jungles, stem from the general tendency in the field to develop general-purpose software packages. Yet another source of problems discussed in the paper is the configuration debt, which is caused by the fact that ML systems, besides all configurations a regular software systemmay require, add a sizable number of ML-specific configuration settings that have to be set and maintained. Researchers and software engineers often find themselves working together on the same project aiming to reach a business goal with a machine learning approach. On surface there seems to be a clear separation of responsibilities: researchers produce the model while engineers build infrastructure to run it. In reality, their areas of concern often overlap when considering the development process, model inputs and outputs and performance metrics. Contributors in both roles often work on the same code. Thus it is beneficial to loop researchers into the whole development journey, making sure they own the product code base along with the engineers, use the same version control and participate in code reviews. Despite obvious onboarding and slow-start challenges, this approach was seen to bring long term benefits in terms of speed and quality of product delivery [12].
Modified on	24/02/2022 11:04

Name	Challenges in Deploying Machine Learning~ a Survey of Case Studies
Number of Coding References	13
Number of Codes Coding	9
Coverage	23.87%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	10/02/2022 11:25
Coded Text	4 Model Learning Model learning is the stage of the deployment workflow that enjoys the most attention within the academic community. All modern research in machine learning methods contributes towards better selection and variety ofmodels and approaches that can be employed at this stage. As an illustration of the scale of the field's growth, the number of submissions to NeurIPS, primary conference on ML methods, has quadrupled in six years, going from 1678 submissions in 2014 to 6743 in 2019 [29]. Nevertheless, there is still plenty of practical considerations that affect the model learning stage. In this section, we discuss issues concerning three steps within model learning: model selection, training and hyper-parameter selection. 4.1 Model selection In many practical cases the selection of a model is often decided by one key characteristic of a model: complexity. Despite areas such as deep learning and reinforcement learning gaining increasing levels of popularitywith the research community, in practice simpler models are often chosen as we explain below. Such model include shallow network architectures, simple PCA-base approaches, decision trees and random forests. Simple models can be used as a way to prove the concept of the proposed ML solution and get the end-to-end setup in place. This approach accelerates the time to get a deployed solution, allows the collection of important feedback and also helps avoid overcomplicated designs. This was the case reported by Haldar et al. [30]. In the process of applying machine learning to AirBnB search, the team started with a complex deep learning model. The team was quickly overwhelmed by its complexity and ended up consuming development cycles. After several failed deployment attempts the neural network architecture was drastically simplified: a single hidden layer NN with 32 fully connected ReLU activations. Even such a simple model had value, as it allowed the building of a whole pipeline of deploying ML models in production setting, while providing reasonably good performance2. Over time the model evolved, with a second hidden layer being added, but it still remained fairly simple, never reaching the initially intended level of complexity. Another advantage that less complex models can offer is their relatively modest hardware requirements. This becomes a key decision point in resource constrained environments, as shown by Wagstaff et al. [31]. They worked on deploying ML models to a range of scientific instruments onboard Europa Clipper spacecraft. Spacecraft design is always a trade-off between the total weight, robustness and the number of scientific tools onboard. Therefore computational resources are scarce and their usage has to be as small as possible. These requirements naturally favor the models that are light on computational demands. The team behind Europa Clipper used machine learning for three anomaly detection tasks, some models took time series data as input and some models took images, and on all three occasions simple threshold or PCA based techniques were implemented. They were specifically chosen because of their robust performance and low demand on computational power. A further example of a resource-constrained environment is wireless cellular networks, where energy, memory consumption and data transmission are very limited. Most advanced techniques, such as deep learning, are not considered yet for practical deployment, despite being able to handle highly dimensional mobile network data [32]. The ability to interpret the output of a model into understandable business domain terms often plays a critical role in model selection, and can even outweigh performance considerations. For that reason decision trees (DT), which can be considered a fairly basic ML algorithm, are widely used in practice. For example, Hansson et al. [33] describe several cases in manufacturing that adopt DT due to its high interpretability. Banking is yet another example of an industry where DT finds extensive use. As an illustrative example, it is used by Keramati et al. [34] where the primary goal of the ML application is to predict customer churn by understanding if-then rules. While it is easy to imagine more complicated 2We discuss more benefits of setting up the automated deployment pipeline in Section 6.3. 6 models learning the eventual input-output relationship for this specific problem, interpretability is key requirement here because of the need to identify the features of churners. The authors found DT to be the best model to fulfill this requirement. Nevertheless, deep learning (DL) is commonly used for practical background tasks that require analysis a large amount of previously acquired data. This

	notion is exemplified by the field of unmanned aerial vehicles (UAV) [35]. Image sensors are commonplace in UAVs due to their low cost, low weight, and low power consumption. Consequently, processing images acquired from sensors is the main way of exploiting excellent capabilities in processing and presentation of raw data that DL offers. But computational resource demands still remain the main blocker for deploying DL as an online processing instrument on board of UAVs. 4.2 Training One of the biggest concern with model training is the economic cost associated with carrying the training stage due to the computational resources required. This is certainly true in the field of natural language processing (NLP), as illustrated by Sharir et al. [36]. The authors observe that while the cost of individual floating-point operations is decreasing, the overall cost of training NLP is only growing. They took one of the state-of-the-art models in the field, BERT [37], and found out that depending on the chosen model size full training procedure can cost anywhere between $50k and $1.6m, which is unaffordable for most research institutions and even companies. The authors observe that training dataset size, number ofmodel parameters and number of operations utilized by the training procedure are all contributing towards the overall cost. Of particular importance here is the second factor: novel NLP models are already using billions of parameters, and this number is expected to increase further in the nearest future [38]. A related concern is raised by Strubell et al. [39] regarding the impact the training ofML models has on the environment. By consuming more and more computational resources, ML models training is driving up the energy consumption and greenhouse gas emissions. According to the estimates provided in the paper, one full training cycle utilizing neural architecture search emits the amount of CO2 comparable to what four average cars emit in their whole lifetime. The authors stress how important it is for researchers to be aware of such impact of model training, and argue that community should give higher priority to computationally efficient hardware and algorithms.
Modified on	24/02/2022 11:03

Name	Challenges in Deploying Machine Learning~ a Survey of Case Studies
Number of Coding References	13
Number of Codes Coding	9
Coverage	23.87%
Folder Location	Codes\\Snowballing
Parent Name	Codes\\Snowballing\\Snowballing
Created on	21/06/2022 14:00
Coded Text	Even though ML deployments require a certain amount of software development, ML projects are fundamentally different from traditional software engineering projects, and we argue they need a different approach. The main differences arise from unique activities like data discovery, dataset preparation, model training, deployment success measurement etc. Some of these activities cannot be defined precisely enough to have a reliable time estimate, some assume huge potential risks, and some make it difficult to measure the overall added value of the project. Therefore ML deployments do not lend themselves well to widespread approaches to software engineering management paradigms, and neither to common software architectural patterns. Data Oriented Architectures (DOA, [97], [98]) is an example of an idea that suggests rethinking how things are normally approached in software development, and by doing so promises to solve quite a few issues we have discussed in this survey. Specifically, the idea behind DOA is to consider replacing micro-service architecture, widespread in current enterprise systems, with streaming-based architecture, thus making data flowing between elements of business logic more explicit and accessible. Micro-service architectures have been successful in supporting high scalability and embracing the single responsibility principle. However, they also make data flows hard to trace, and it is up to owners of every individual service to make sure inputs and outputs are being stored in a consistent form. DOA provides a solution to this problem by moving data to streams flowing between stateless execution nodes, thus making data available and traceable by design, therefore making simpler the tasks of data discovery, collection and labeling. In its essence DOA proposes to acknowledge 14 that modern systems are often data-driven, and therefore need to prioritize data in their architectural principles. As noted above, ML projects normally do not fit well with commonly used management processes, such as Scrum or Waterfall. Therefore it makes sense to consider processes tailored specifically for ML. One such attempt is done by Lavin et. al. [99], who propose Technology Readiness Levels for ML (TRL4ML) framework. TRL4ML describes a process of producing robust ML systems that takes into account key differences between ML and traditional software engineering. A very widespread practice in software engineering is to define a set of guidelines and best practices to help developers make decisions at various stages of the development process. These guidelines can cover a wide range of questions, from variable names to execution environment setup. Similarly, Zinkevich [100] compiled a collection of best practices for machine learning that are utilized in Google. While this cannot be viewed as a coherent paradigm towards doing ML deployment, this document gives practical advice on a variety of important aspects that draw from the real life experiences of engineers and researchers in the company. Holistic approaches are created withML application in mind, and therefore they have the potential to offer a significant ease of deployingML. But it should be noted that all such approaches assume a big time investment, because they represent significant changes to current norms in project management and development. Therefore a careful assessment of risks versus benefits should be carried out before adopting any of them.
Modified on	22/06/2022 11:49

Name	Challenges in Deploying Machine Learning~ a Survey of Case Studies
Number of Coding References	13
Number of Codes Coding	9
Coverage	23.87%
Folder Location	Codes\\Snowballing
Parent Name	Codes\\Snowballing\\Snowballing
Created on	21/06/2022 14:00
Coded Text	The market for machine learning tools and services is experiencing rapid growth [87]. As a result, tools for individual deployment problems are continuously developed and released. For some of the problems we have highlighted, making use of the right specific tool is an entirely reasonable approach. For example, this is most likely the case for operational maintenance ofMLmodels. Many platforms on the market offer end-to-end experience for the user, taking care of such things as data storage, retraining and deployment. Examples include AWS SageMaker [88], Microsoft AzureML [89], Uber Michelangelo [90], TensorFlow TFX [91], MLflow [92] and more. A typical ML platform would include, among other features, data storage facility, model hosting with APIs for training and inference operations, a set of common metrics to monitor model health and an interface to accept custom changes from the user. By offering managed infrastructure and a range of out-of-the-box implementations for common tasks such platforms greatly reduce operational burden associated with maintaining the ML model in production. Quality assurance also looks to be an area where better tools can be of much assistance. As mentioned in Section 3, data integrity plays a big part in quality control, and is an active branch of research [18]. Models themselves can also greatly benefit from development of a test suite to verify their behavior. This is normally done by trial and error, but more formal approaches are being developed, as is the case with CheckList methodology for NLP models [93]. As discussed in Section 3.3, obtaining labels is often a problem with real world data. Weak supervision has emerged as a separate field of ML which looks for ways to address this challenge. Consequently, a number of weak supervision libraries are now actively used within the community, and show promising results in industrial applications. Some of the most popular tools include Snorkel [94], Snuba [95] and cleanlab [96]. Using specific tools for solving individual problems is a straightforward approach. However practitioners need to be aware that by using a particular tool they introduce an additional dependency into their solution. While a single additional dependency seems manageable, their number can quickly grow and become a maintenance burden. Besides, as we mentioned above, new tools for ML are being released constantly, thus presenting practitioners with the dilemma of choosing the right tool by learning its strength and shortcomings.
Modified on	22/06/2022 11:49

Name	Cirrus~ a Serverless Framework for End-to-end ML Workflows
Number of Coding References	6
Number of Codes Coding	2
Coverage	7.96%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	The widespread adoption ofML techniques in a wide-range of domains, such as image recognition, text, and speech processing, has made machine learning one of the leading revenue-generating datacenter workloads. Unfortunately, due to the growing scale of these workloads and the increasing complexity ofML workflows, developers are often left to manually configure numerous system-level parameters (e.g., number ofworkers/parameter servers, memory footprint, amount ofcompute, physical topology), in addition to the ML-specific parameters (learning rate, algorithms, neural network structure). Importantly, modern ML workflows are iterative and increas- ingly comprised of multiple heterogeneous stages, such as (a) preprocessing, (b) training, and (c) hyperparameter searching. As a result, due to the iterative nature and diversity of stages, the endto-end ML workflow is highly complex for users and demanding in terms of resource provisioning and management, detracting users from focusing on ML specific tasks—the domain of their expertise. The complexity ofML workflows leads to two problems. First, when operating with coarse-grained VM-based clusters the provisioning complexity often leads to overprovisioning. Aggregate CPU utilization levels as low as 20% are not uncommon [27, 45]. Second, the management complexity is increasingly an obstacle for ML users because it hinders the interactive and iterative use-cases, degrading user productivity and model effectiveness.
Modified on	28/02/2023 13:20

Name	Cirrus~ a Serverless Framework for End-to-end ML Workflows
Number of Coding References	6
Number of Codes Coding	2
Coverage	7.96%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	DEMOCRATIZING MACHINE LEARNING 2.1 End-to-end MLWorkflow Challenges Machine learning researchers and developers execute a number of different tasks during the process of training models. For instance, a common workflow consists of dataset preprocessing, followed by model training and finally by hyperparameter tuning (Figure 1). In the dataset preprocessing phase, developers apply transformations (e.g., feature normalization or hashing) to datasets to improve the performance of learning algorithms. Subsequently, in the model training phase, developers coarsely fit a model to the dataset, with the goal of finding a model that performs reasonably well and converges to an acceptable accuracy level. Finally, in the hyperparameter tuning phase, the model is trained multiple times with varying ML-parameters to find the parameters that yield best results. ML training tasks have been traditionally deployed using systems designed for clusters of virtual execution environments (VMs) [19, 20, 26, 49, 51]. However, such designs create two important challenges for users: (a) they can lead to over-provisioning (b) they require explicit resource management by users. Over-provisioning. The heterogeneity of the different tasks in an MLworkflowleads to a significant resource imbalance during the execution of a training workflow. For instance, the coarse-granularity and rigidity of VM-based clusters as well as the design of the ML frameworks specialized for these environments causes developers
Modified on	28/02/2023 13:20

Name	Cirrus~ a Serverless Framework for End-to-end ML Workflows
Number of Coding References	6
Number of Codes Coding	2
Coverage	7.96%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Explicit resource management. The established approach of exposing low-level VM resources, such as storage and CPUs, puts a significant burden on ML developers who are faced with the challenge of provisioning, configuring, and managing these resources for each oftheir MLworkloads. Thus, systems that leverage VMs for machine learning workloads generally require users to repeatedly perform a series of onerous tasks we summarize in Table 1. In practice, over-provisioning and explicit resource management burden are tightly coupled—ML users often resort to over-provisioning due to the difficulty and human cost of accurately managing resource allocation for the different stages of their training workflow. 2.2 Serverless Computing Serverless computing is a promising approach to address these resource-provisioning challenges [31, 35]. It simultaneously simplifies deployment with its intuitive interface and provides mechanisms to avoid over-provisioning, with its fine-grain serverless functions that can run with as few as 128MB of memory (spatial granularity) and time out in a few minutes (temporal granularity). This ensures natural elasticity and agility of deployment. However, serverless design principles are at odds with a number of design principles of existing ML frameworks today. This presents a set of challenges in adopting serverless infrastructures for ML training workflows. This section discusses the major limitations of existing serverless environments and the impact they have for machine learning systems. Small local memory and storage. Lambda functions, by design, have very limited memory and local storage. For instance, AWS lambdas can only access at most 3GB of local RAM and 512MB of local disk. It is common to operate with lambdas provisioned for as little as 128MB of RAM. This precludes the strategy often used by many machine learning systems of replicating or sharding the training data across many workers or of loading all training data into memory. These resource limitations prevent the use of any computation frameworks that are not designed with these resource Short-livedandunpredictablelaunch times. Lambda functions are short-lived and their launch times are highly variable. For instance, AWS lambdas can take up to several minutes to start after being launched. This means that during training, lambdas start at unpredictable times and can finish in the middle of training. This requires ML runtimes for lambdas to tolerate the frequent departure and arrival ofworkers. Furthermore, it makes runtimes such as MPI (used, for instance, by Horovod [47] and Multiverso [7]) a bad fit for this type of architecture. Lack offast shared storage. Because lambda functions cannot connect between themselves, shared storage needs to be used. Because ML algorithms have stringent performance requirements, this shared storage needs to be low-latency, high-throughput, and optimized for the type of communications in ML workloads. However, as of today there is no fast serverless storage for the cloud that provides all these properties. 3 CIRRUS DESIGN Cirrus is an end-to-end framework specialized for ML training in serverless cloud infrastructures (e.g., Amazon AWS Lambdas). It provides high-level primitives to support a range of tasks in ML workflows: dataset preprocessing, training, and hyperparameter optimization. This section describes its design and architecture. 3.1 Design Principles Adaptive, fine-grained resource allocation. To avoid resource waste that arises from over-provisioning, Cirrus should flexibly adapt the amount of resources reserved for each workflow phase with fine-granularity. Stateless server-sidebackend. To ensure robust and efficient management of serverless compute resources, Cirrus, by design, operates a stateless, server-side backend (Figure 3). The information about currently deployed functions and the mapping between ML workflow tasks and compute units is managed by the client-side backend. Thus, even when all cloud-side resources become unavailable, the ML training workflow does not fail and may resume its operation when resources become available again. End-to-end serverless API.. Model training is not the only important task an ML researcher has to perform. Dataset preprocessing, feature engineering, and parameter tuning are other examples of
Modified on	28/02/2023 13:20

Name	Cirrus~ a Serverless Framework for End-to-end ML Workflows
Number of Coding References	6
Number of Codes Coding	2
Coverage	7.96%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Importantly, Cirrus is designed to efficiently support the en- tire ML workflow. In particular, Cirrus supports fine-grain, dataintensive serverless ML training and hyperparameter optimization efficiently. Based on the parameter server model (see Figure 2), Cirrus provides an easy-to-use interface to perform scalable ML training leveraging the high scalability of serverless computation environments and cloud storage. Cirrus unifies the benefits of specialized serverless frameworks with the benefits of specialized ML training frameworks and provides an easy-to-use interface (§4) that enables typical ML training workflows and supervised learning algorithms (e.g., Logistic Regression, Collaborative Filtering) for end-to-end ML workflows on serverless infrastructure. Cirrus builds on three key design properties. First, Cirrus pro- vides an ultra-lightweight (~80MB vs 800MB for PyWren's runtime) worker runtime that adapts to the full range of lambda granularity, providing mechanisms for ML developers to find the configuration that best matches their time or cost budget. Second, Cirrus saves on the cost of provisioning large amounts of memory or storage—a typical requirement for ML training frameworks. This is achieved
Modified on	28/02/2023 13:20

Name	Cirrus~ a Serverless Framework for End-to-end ML Workflows
Number of Coding References	6
Number of Codes Coding	2
Coverage	7.96%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Machine learning (ML) workflows are extremely complex. The typical workflow consists of distinct stages of user interaction, such as preprocessing, training, and tuning, that are repeatedly executed by users but have heterogeneous computational requirements. This complexity makes it challenging for ML users to correctly provision and manage resources and, in practice, constitutes a significant burden that frequently causes over-provisioning and impairs user productivity. Serverless computing is a compelling model to address the resource management problem, in general, but there are numerous challenges to adopt it for existing ML frameworks due to significant restrictions on local resources. This work proposes Cirrus—an ML framework that automates the end-to-end management of datacenter resources for ML workflows by efficiently taking advantage of serverless infrastructures. Cirrus combines the simplicity of the serverless interface and the scalability of the serverless infrastructure (AWS Lambdas and S3) to minimize user effort. We show a design specialized for both serverless computation and iterative ML training is needed for robust and efficient ML training on serverless infrastructure. Our evaluation shows that Cirrus outperforms frameworks specialized along a single dimension: Cirrus is 100x faster than a general purpose serverless system [36] and 3.75x faster than specialized ML frameworks for traditional infrastructures [49].
Modified on	28/02/2023 13:19

Name	Cirrus~ a Serverless Framework for End-to-end ML Workflows
Number of Coding References	6
Number of Codes Coding	2
Coverage	7.96%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	This work proposes Cirrus, a distributed ML training framework that addresses these challenges by leveraging serverless computing. Serverless computing relies on the cloud infrastructure, not the users, to automatically address the challenges ofresource provisioning and management. This approach relies on a more restricted unit of computation, the stateless lambda function, which is submitted by developers and scheduled to execute by the cloud infrastructure. Thus, obviating the need for users to manually configure, deploy, and manage long-term compute units (e.g., VMs). The advantages of the serverless paradigm have promoted its fast adoption by datacenters and cloud providers [2, 5, 6, 9, 10, 15] and open source platforms [8, 16, 18, 32]. However, the benefits of serverless computing for ML hinge on the ability to run ML algorithms efficiently. The main challenge in leveraging serverless computing is the significantly small local resource constraints (memory, cpu, storage, network) associated with lambda functions, which is fundamental to serverless computation because the fine-granularity of computation units enables scalability and flexibility. In contrast, existing ML systems commonly assume abundant resources, such as memory. For instance, Spark [51] and Bosen [49, 50] generally load all training data into memory.
Modified on	28/02/2023 13:20

Name	Cirrus~ a Serverless Framework for End-to-end ML Workflows Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Collaboration Challenges in Building ML-Enabled Systems~ Communication, Documentation, Engineering, and Process
Number of Coding References	7
Number of Codes Coding	3
Coverage	4.13%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	Three collaboration points surfaced as particularly challenging: (1) Identifying and decomposing requirements, (2) negotiating training data quality and quantity, and (3) integrating data science and software engineering work. We found that organizational structure, team composition, power dynamics, and responsibilities differ substantially, but also found common organizational patterns at specific collaboration points and challenges associated with them. Overall, our observations suggest four themes that would benefit from more attention when building ML-enabled systems: ■ Invest in supporting interdisciplinary teams to work together (including education and avoiding silos), ☐ Pay more attention to collaboration points and clearly document responsibilities and interfaces, ■ Consider engineering work as a key contribution to the project, and ☑ Invest more into process and planning. In summary, we make the following contributions: (1) We iden- tify three core collaboration points and associated collaboration challenges based on interviews with 45 practitioners, triangulated with a literature review, (2) We highlight the different ways in which teams organize, but also identify organizational patterns that associate with certain collaboration challenges, and (3) We identify recommendations to improve collaboration practices.
Modified on	16/02/2023 10:23

Name	Collaboration Challenges in Building ML-Enabled Systems~ Communication, Documentation, Engineering, and Process
Number of Coding References	7
Number of Codes Coding	3
Coverage	4.13%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	Through our interviews we identified three central collaboration points where organizations building ML-enabled systems face substantial challenges: (1) requirements and project planning, (2) training data, and (3) product-model integration. Other collaboration points surfaced, but were mentioned far less frequently (e.g., interaction with legal experts and operators), did not relate to problems between multiple disciplines (e.g., data scientists documenting their work for other data scientists), or mirrored conventional collaboration in software projects (e.g., many interviewees wanted to talk about unstable ML libraries and challenges interacting with teams building and maintaining such libraries, though the challenges largely mirrored those of library evolution generally [16, 31]). Data scientists and software engineers are certainly not the first to realize that interdisciplinary collaborations are challenging and fraught with communication and cultural problems [21], yet it seems that many organizations building ML-enabled systems pay little attention to fostering better interdisciplinary collaboration. Organizations differ widely in their structures and practices, and some organizations have found strategies that work for them (see recommendation sections). Yet, we find that most organizations do not deliberately plan their structures and practices and have little insight into available choices and their tradeoffs. We hope that this work can (1) encourage more deliberation about organization and process at key collaboration points, and (2) serve as a starting point for cataloging and promoting best practices. ICSE '22, May 21–29, 2022, Pittsburgh, PA, USA Beyond the specific challenges discussed throughout this paper, we see four broad themes that benefit from more attention both in engineering practice and in research: ■ Communication: Many issues are rooted in miscommunication between participants with different backgrounds. To facilitate interdisciplinary collaboration, education is key, including ML literacy for software engineers and managers (and even customers) but also training data scientists to understand software engineering concerns. The idea of T-shaped professionals [102] (deep expertise in one area, broad knowledge of others) can provide guidance for hiring and training. ☐ Documentation: Clearly documenting expectations between teams is important. Traditional interface documentation familiar to software engineers may be a starting point, but practices for documenting model requirements (Sec. 5.2), data expectations (Sec. 6.2), and assured model qualities (Sec. 7.3) are not well established. Recent suggestions like model cards [64], and FactSheets [8]are a good starting point for encouraging better, more standardized documentation ofML components. Given the interdisciplinary nature at these collaboration points, such documentation must be understood by all involved – theories of boundary objects [2] may help to develop better interface description mechanisms. ■ Engineering: With attention focused on ML innovations, many organizations seem to underestimate the engineering effort required to turn a model into a product to be operated and maintained reliably. Arguably adopting machine learning increases software complexity [48, 69, 87] and makes engineering practices such as data quality checks, deployment automation, and testing in production even more important. Project managers should ensure that the ML and the non-ML parts of the project have sufficient engineering capabilities and foster product and operations thinking from the start. ☑ Process: Finally, machine learning with its more science-like process challenges traditional software process life cycles. It seems clear that product requirements cannot be established without involving data scientists for model prototyping, and often it may be advisable to adopt a model-first trajectory to reduce risk. But while a focus on the product and overall process may cause delays, neglecting it entirely invites the kind of problems reported by our participants. Whether it may look more like the spiral model or agile [22], more research into integrated process life cycles for MLenabled systems (covering software engineering and data science) is needed.
Modified on	16/02/2023 10:22

Name	Collaboration Challenges in Building ML-Enabled Systems~ Communication, Documentation, Engineering, and Process
Number of Coding References	7
Number of Codes Coding	3
Coverage	4.13%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	Technical aspects such as testing ML components [10, 20], mis- use ofML libraries [43, 45], engineering challenges for developing ML components [3, 5, 18, 27, 40, 44, 60, 90], and automating learning and deployment processes for ML components [4, 13, 29, 34, 51], have received significant attention in research recently. However, human factors of collaboration during the development of software products supported by ML components, ML-enabled systems for short, have received less attention, including the need to separate and coordinate data science and software engineering work, to negotiate and document interfaces and responsibilities, and to plan the system's operation and evolution. Yet, those human collaboration challenges appear to be major hurdles in developing ML-enabled systems. In addition, past work has mostly been model-centric, focused on challenges of learning, testing, or serving models, but rarely focuses on the entire system, i.e., the product with many non-ML parts into which the model is embedded as a component, which requires coordinating and integrating work from multiple experts or teams.
Modified on	16/02/2023 10:20

Name	Collaboration Challenges in Building ML-Enabled Systems~ Communication, Documentation, Engineering, and Process
Number of Coding References	7
Number of Codes Coding	3
Coverage	4.13%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	The introduction ofmachine learning (ML) components in software projects has created the need for software engineers to collaborate with data scientists and other specialists. While collaboration can always be challenging, ML introduces additional challenges with its exploratory model development process, additional skills and knowledge needed, difficulties testing ML systems, need for continuous evolution and monitoring, and non-traditional quality requirements such as fairness and explainability
Modified on	16/02/2023 10:20

Name	Collaboration Challenges in Building ML-Enabled Systems~ Communication, Documentation, Engineering, and Process
Number of Coding References	7
Number of Codes Coding	3
Coverage	4.13%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	Three collaboration points surfaced as particularly challenging: (1) Identifying and decomposing requirements, (2) negotiating training data quality and quantity, and (3) integrating data science and software engineering work. We found that organizational structure, team composition, power dynamics, and responsibilities differ substantially, but also found common organizational patterns at specific collaboration points and challenges associated with them. Overall, our observations suggest four themes that would benefit from more attention when building ML-enabled systems: ■ Invest in supporting interdisciplinary teams to work together (including education and avoiding silos), ☐ Pay more attention to collaboration points and clearly document responsibilities and interfaces, ■ Consider engineering work as a key contribution to the project, and ☑ Invest more into process and planning. In summary, we make the following contributions: (1) We iden- tify three core collaboration points and associated collaboration challenges based on interviews with 45 practitioners, triangulated with a literature review, (2) We highlight the different ways in which teams organize, but also identify organizational patterns that associate with certain collaboration challenges, and (3) We identify recommendations to improve collaboration practices.
Modified on	16/02/2023 10:21

Name	Collaboration Challenges in Building ML-Enabled Systems~ Communication, Documentation, Engineering, and Process
Number of Coding References	7
Number of Codes Coding	3
Coverage	4.13%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	While some organizations have adopted better collaboration practices than others, many struggle setting up structures, processes, and tooling for effective collaboration among team members with different backgrounds when developing ML-enabled systems. To the best of our knowledge, and confirmed by the practitioners we interviewed, there is little systematic or shared understanding of common collaboration challenges and best practices for developing ML-enabled systems and coordinating developers with very different backgrounds (e.g., data science vs. software engineering). We find that smaller and new-to-ML organizations struggle more, but have limited advice to draw from for improvement.
Modified on	16/02/2023 10:21

Name	Collaboration Challenges in Building ML-Enabled Systems~ Communication, Documentation, Engineering, and Process
Number of Coding References	7
Number of Codes Coding	3
Coverage	4.13%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	04/06/2022 17:30
Coded Text	The introduction ofmachine learning (ML) components in software projects has created the need for software engineers to collaborate with data scientists and other specialists. While collaboration can always be challenging, ML introduces additional challenges with its exploratory model development process, additional skills and knowledge needed, difficulties testing ML systems, need for continuous evolution and monitoring, and non-traditional quality requirements such as fairness and explainability
Modified on	16/02/2023 10:19

Name	Collaboration Challenges in Building ML-Enabled Systems~ Communication, Documentation, Engineering, and Process Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Communication optimization strategies for distributed deep neural network training~ A survey
Number of Coding References	18
Number of Codes Coding	2
Coverage	21.52%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 23:01
Coded Text	Regarding good performance, DNNs tend to be deeper and more sophisticated and are trained with the larger datasets. The rapid increase of data volumes and model sizes has resulted in large number of computations, indicating that DNN training is time-consuming and can extend over several days or weeks. High-performance hardware, such as graphics processing units * Corresponding author. E-mail address: dong@nudt.edu.cn (D. Dong). https://doi.org/10.1016/j.jpdc.2020.11.005 0743-7315/© 2020 Elsevier Inc. All rights reserved. (GPU) [79] and tensor processing units (TPU) [67], are applied to accelerate the training time. Beyond using high-performance hardware, paralleling and deploying DNN training tasks on multiple nodes (consisting of one or more machines) is another practical approach. Under these conditions, each node only executes part of an entire computation task. However, due to the frequent communication requirements for exchanging large amounts of data among the different computation nodes, communications overhead is a critical bottleneck in distributed training. With the growth of the cluster scale, communications overhead has increased explosively. Such a phenomenon considerably diminishes the advantage of parallel training, as a majority of the training time is spent on transferring data. When high-performance hardware accelerators are used, the proportion of time spent on communication increases further because they only decrease the computation overhead, whereas the communications overhead is unchanged.
Modified on	28/02/2023 15:30

Name	Communication optimization strategies for distributed deep neural network training~ A survey
Number of Coding References	18
Number of Codes Coding	2
Coverage	21.52%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	2.4.1. Synchronous In the synchronous update, the server does not update the model until it receives gradients from all the workers at each iteration. In other words, faster workers will wait for slower workers. One well-known implementation of the synchronous update is bulk synchronous parallel (BSP) [110]. A characteristic of the synchronous mode is that the server will always receive the latest gradients of all the nodes, which does not affect the model's convergence. However, fast nodes idle when waiting for slow nodes, leading to a waste of resources and causing the straggler problem that slows the overall training time. 2.4.2. Asynchronous Asynchronous algorithms such as Hogwild! [92] overcome the above problems. In asynchronous updates, fast workers do not wait for slow workers. One worker may be sending its local gradients to the server while others are calculating their gradients, as shown in Fig. 3. For example, the first worker calculates and pushes its local gradients when t = 1 whereas the second worker pushes gradients when t = 2, although they both pull parameters from servers at the same timestamp (t = 0). The primary challenge raised by asynchronous updates is data staleness, because fast workers may always use stale parameters which jeopardizes model's convergence. In addition, the fastest worker is equivalent to update its local parameters by its subdataset, which causes the local model to deviate from the global model. To overcome the drawbacks of asynchronous updates, researchers have attempted to limit the staleness of the parameters. In bounded stale updates, fast workers will use stale parameters but the staleness (same as delay in Fig. 3) is limited [32,60]. The
Modified on	28/02/2023 15:31

Name	Communication optimization strategies for distributed deep neural network training~ A survey
Number of Coding References	18
Number of Codes Coding	2
Coverage	21.52%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	3.1.2. Periodic communication Recall that when training DNNs using a traditional distributed SGD, communication often occurs at the end of each iteration. Suppose that we parallel a DNN training task on K nodes with T iterations, the complexity of rounds of a vanilla parallel SGD is O(T). Due to such high complexity of communication rounds, many research studies suggest reducing the frequency of exchanging gradients and parameters. In model averaging, individual model parameters trained on local nodes are averaged periodically. We consider averaging operations occur at most for τ iterations, where τ is a factor of the period. Averaging occurs in every iteration, i.e., $\tau = 1$, which is identical to a vanilla parallel SGD. Conversely, averaging
Modified on	28/02/2023 15:31

Name	Communication optimization strategies for distributed deep neural network training~ A survey
Number of Coding References	18
Number of Codes Coding	2
Coverage	21.52%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	3.2. Gradients compression Each computation node requires communication operations to exchange gradients and model parameters during the distributed training. Using 32 bits variables to represent each element in the gradients and model parameters is the most commonly used configuration. When the size of the gradients and model parameters is large, the communication bottleneck caused by exchanging a large amount of 32 bits single-precision variables impairs the advantages of parallel SGDs. For example, consider training the BERT architecture for neural language processing tasks with approximately 340 million parameters, which implies that each full precision exchange between nodes is over 1.2 GB [38]. Therefore, we will discuss how to reduce message size in this section. Gradient compression reduces the communication traffic by compressing the gradients transmitted during training. It is easy to implement and performs very well on some DNN models. However, the model's performance will be affected by these lossy compression methods, which is unacceptable in some areas (i.e., recommendation systems) where the accuracy is sensitive. Currently there are mainly three bandwidth-efficient compression approaches: quantization, sparsification and decomposition. The first, substitutes 32 bits variables for low-precision variables (i.e., 8 bits, 4 bits, or even 1 bit), whereas the second only transports important variables to avoid unnecessary overhead. Since these two approaches are orthogonal, they can be combined to compress the communication further. The third is not as popular as the first two; it transmits small matrices instead of large matrices by a matrix decomposition. Fig. 7 compares various compression methods.
Modified on	28/02/2023 15:31

Name	Communication optimization strategies for distributed deep neural network training~ A survey
Number of Coding References	18
Number of Codes Coding	2
Coverage	21.52%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	Centralized and decentralized architectures The (logical) architecture of the computation nodes can affect the communication modes and the network's performance. Parameter server [73,74] is the most popular centralized architecture in distributed deep learning. A parameter server usually includes a server node and several worker nodes. The server maintains global model parameters, whereas each worker stores a local model replica. If the server node has more than one machine, each machine maintains a partition of the entire model parameters. For workers, each worker stores an entire model replica in data parallelism or a part of the model in model parallelism. Workers communicate with the server via a push/pull operation, whereas there is no communication between any workers. The drawback of the parameter server is the bandwidth bottleneck on the server-side with the increment of workers. Due to this drawback of the parameter server, decentralized architectures have attracted considerable research attention because they incur fewer communication traffic issues [76]. Similar to the parameter server, each node in a decentralized architecture holds an entire model replica, but they use collective communication operation Allreduce instead of push/pull to exchange gradients and parameters. Nevertheless, the Allreduce operation has a variety of implementations with notable differences in performance, which means that it may affect the communications overhead. We will discuss related issues in Section 4. 2.4. Synchronous and asynchronous updates Owing to differences in network bandwidth and computing power, some nodes may calculate gradients faster, while other nodes may be slower. The main challenge that comes with such circumstances is determining when to synchronize the gradients among multiple computation nodes. There are three different methods that reasonably solve this problem: synchronous, asynchronous, and bounded delay updates.
Modified on	28/02/2023 15:31

Name	Communication optimization strategies for distributed deep neural network training~ A survey
Number of Coding References	18
Number of Codes Coding	2
Coverage	21.52%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	Conclusion It has been shown that communications overhead is a sig- nificant obstacle to achieving a desirable performance for distributed DNN training. In this paper, we provide a comprehensive survey of the recent research on communication optimization techniques for distributed DNN training. Both theoretical and experimental studies are investigated. We divide these communication optimization strategies into two dimensions: algorithm level optimization and network level optimization. From the algorithm perspective, we elaborate on techniques to reduce communication volumes and computation–communication overlaps. In terms of network level optimization, we discussed the impact of different topologies and network protocols on distributed deep learning. Below, we highlight potential new research directions and challenges. Focusing on different deep learning tasks and neural network models. At present, there are a considerable number of communication optimizations that focus on image classification tasks, especially ResNet-50 trained on ImageNet dataset. Natural language processing and recommendation system related tasks have not received notable research attention. For example, the large embedding matrix in the recommendation system model may become a new bottleneck in communication optimization. Local SGD for nonconvex problems. For periodic communica- tions, there still remains some research opportunities for nonconvex problems. Despite many theoretical works on model averaging, the research on whether the linear speedup with $\tau > 1$ can be preserved for nonconvex optimizations is still unexplored. Individually, the lower bounds on the number of communication rounds for nonconvex optimizations to achieve linear speedup are an interesting research direction. Trade-off between model accuracy and compression ratio. The core challenge that requires consideration in gradient compression is the trade-off between model accuracy and the compression ratio. Conventional approaches to prevent models from diverging include error feedback (for quantization) and local gradient accumulation (for sparsification). More advanced methods such as squared error feedback [77] need to be explored in future works.
Modified on	28/02/2023 15:32

Name	Communication optimization strategies for distributed deep neural network training~ A survey
Number of Coding References	18
Number of Codes Coding	2
Coverage	21.52%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	Higher computation–communication overlap ratio. The ratio of computation to communication is essential for deploying pipeline training. For the sake of a higher overlap rating, various algorithms use different strategies to arrange communication operations and shrink communication time. However, these algorithms are generally heuristic and achieve nonoptimal solutions in such scheduling problems. Better optimization algorithms, such as dynamic programming, maybe be suitable for solving this problem. Large-scale DNN training on top of different datacenter network topology. The physical topology of the datacenter network has a significant impact on distributed deep learning. A recent work by Wang et al. [113] deployed a parameter server over BCube [51] instead of a traditional Fat-Tree [4] architecture and achieved good performance on LeNet-5 [72] and VGG-19 [103] training. Other network topologies, such as DCell [52], may further accelerate large-scale neural network training. Communications overhead analysis tools are required. Another critical research issue is the performance model and measurement tools of distributed DNN training. Performance models allow us to theoretically analyze the various costs of distributed training [90,123], and the measurement tools help us study communication behavior and find the bottlenecks in the distributed training tasks. Although some deep learning frameworks including Tensorflow [1] and MXNet [23] currently provide performance analysis tools, these tools cannot analyze network behavior. Advanced tools for network analysis such as Horovod timeline [100] and SketchDLC [120] are still required.
Modified on	28/02/2023 15:33

Name	Communication optimization strategies for distributed deep neural network training~ A survey
Number of Coding References	18
Number of Codes Coding	2
Coverage	21.52%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	In this paper, we primarily investigate how to deal with communications overhead for distributed DNN training. Because distributed deep learning is a cross-disciplinary field, both deep learning and distributed network communities have proposed communication optimization strategies from their own perspectives. In this survey, we bring together, classify, and compare the large body of work on communications optimization for
Modified on	28/02/2023 15:30

Name	Communication optimization strategies for distributed deep neural network training~ A survey
Number of Coding References	18
Number of Codes Coding	2
Coverage	21.52%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	In this section, we provide a brief introduction to large-scale distributed deep neural network training. Currently, the most popular method to train DNNs on a single-machine remains minibatch based stochastic gradient descent (SGD) algorithm [16,17] with error back-propagation [96]. When DNN training moves to parallelization, several problems need to be considered: (i) which part of the training task can be parallelized, (ii) the architectures of the computation nodes, and (iii) when to synchronize gradients. 53 2.1. Stochastic gradient descent SGD and its variants have become the workhorse algorithms for training DNN models. Suppose that we want to train a DNN model to minimize the loss function ℓ(f(x;W), y) averaged on a given dataset D, where f(x,W) represents the DNN's output with parameter W, and x is an input data instance corresponding with the true label y. The application-specific loss function ℓ(·) measures the difference between outputs and true labels. SGD will update model parameters and iteratively converge towards the minimum of the loss function as follows [16]: B Wt+1 = Wt − η × 1 B ∑ i=1 ∇ℓ (f (xi ;Wt) , yi) where B is the number of training instance in each mini-batch, η is learning rate and ∇ℓ (f (xi ;Wt) , yi) are the gradients of current mini-batch. 2.2. Data and model parallelism Data parallelism and model parallelism are two commonly training methods used in distributed deep learning. In data parallelism, the entire training dataset is randomly and equally divided into N parts and dispatched on N nodes, see in Fig. 2(a). Each node maintains a model replica along with its local parameters Wn. The training process is as follows: Step 1. Each node reads a mini-batch of the training data and executes forward and backward propagations to calculate its local gradients ∇ℓ ((= Wt − η N ∑N n=1 ∇ℓ f x;Wn ((t f x;Wn t))) , y . Step 2. Each node sends the local gradients to a master node. After receiving gradients from all the nodes, the master aggregates these gradients and updates the model by Wt+1) , y . Step 3. The master broadcasts the latest model parameters to all other nodes. Step 4. Repeat these three steps until the model converges. With respect to model parallelism [36,45], different parts of the DNN model are split into different nodes. Hence, the number of parameters on a single node is reduced. Nodes, where the input
Modified on	28/02/2023 15:31

Name	Communication optimization strategies for distributed deep neural network training~ A survey
Number of Coding References	18
Number of Codes Coding	2
Coverage	21.52%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	layer is located, are responsible for reading the entire dataset. For example, only Node 1 and Node 3 read input dataset in Fig. 2(b). Correspondingly, nodes where the output layer is located are responsible for outputting the predicted value of the DNN. The calculations between nodes are no longer independent of each other. Only neurons with connections that cross computation nodes (thick line in Fig. 2(b)) will need to exchange gradients and model parameters. Clearly, data parallelism is suitable for a large training dataset, while model parallelism is applicable when a model is too large to fit in a single node. In this paper, we primarily focus on data parallelism.
Modified on	28/02/2023 15:31

Name	Communication optimization strategies for distributed deep neural network training~ A survey
Number of Coding References	18
Number of Codes Coding	2
Coverage	21.52%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	limitation on staleness mitigates the straggler problem to some extent and increases training throughput. However, determining ways to choose the limitation on staleness is a question that is worth discussing because an excessively large value means completely asynchronous updates, while a small value is similar to synchronous updates. 3. Algorithm level optimization In this section, we demonstrate how to reduce the communications overhead in distributed DNN training from an algorithm perspective. Algorithm optimizations include reducing communication rounds and volumes as well as increasing the computation–communication overlap ratio. These optimizations are compatible with the underlying network, and most of these can run on top of various network infrastructures and protocols. 3.1. Communication rounds When using SGD to train deep neural networks, the entire training process usually consists of multiple epochs and iterations. In regard to parallelization, nodes often exchange data at the end of every iteration. One intuitive way to cut down the communication overhead is reducing the number of data exchanges, or the communication rounds. The number of rounds for a node is related to the batch size and the communication period. Larger batch sizes and more extended communication periods can both reduce the number of data exchanging rounds. 3.1.1. Large batch training Batch size is a significant hyperparameter that controls the amount of data read by a node in each iteration. A large batch size usually better approximates the distribution of the input data and introduces fewer variances into the gradient estimates than a small batch. Additionally, a large batch of data will take a longer time to process and incur fewer model parameter update; this finding is primarily due to the relationship between batch size, the number of iterations, and training data size. The following equation shows that using a large batch size leads to a reduction in the number of iterations; hence, the parameter updates are infrequent. Batch Size × # of iterations # of epochs = training data size parameters only depend on the DNN itself, the single iteration communicated message size remains constant, as changing the batch size does not change their shape and size. Therefore, increasing batch sizes reduces the number of iterations and the communications rounds. Take ResNet-50 as an example, if we fix the number of epochs at 100 and set the batch size to 1024 on two machines, the entire training process requires 250,000 iterations, in contrast, with a batch size of 8192 on 16 machines, only 15625 iterations are required [129]. Table 1 shows ResNet-50 training results with different configurations. Nevertheless, by directly deploying a parallel SGD with a huge batch size, in practice it will likely suffer generalization ability degradations compared with training a small batch [48,69]. Fig. 4 illustrates this situation in which, when the batch size exceeds 8k (we use 1k to denote 1024 samples), the error on the validation set increases dramatically as the batch size increases. The reason of this finding is that large batch methods tend to converge to a sharp minima of the training function [69]. These minima are characterized by a significant number of large positive eigenvalues in the Hessian matrix, indicating that the curvature around these minima is large. As shown in Fig. 5, sharp minima often generalize less well. Conversely, small batch method converge to flat minima, which is characterized by having many small eigenvalues for the Hessian matrix and a small curvature. Observations show that the loss function landscape of a DNN is such that largebatch methods are attracted to regions with sharp minima and are unable to escape from them [69]. Many methods have been proposed to prevent models from converging into sharp minima, including learning rate scaling rules [48,70], various warm-up schemes [48,126], and layerwise adaptive rate scaling (LARS) [125,128,129]. The scaling rules dictate that the learning rate should increase to prevent a loss of accuracy with an increase of the batch size. There are two rules of increasing learning rate: Sqrt Scaling Rule, [70] and Linear Scaling Rule, [48]. When the batch size is multiplied by k, the first rule multiplies the learning rate by √ k to keep the variance of the (1) In distributed training circumstances with a data parallelism

	scheme, the batch size is the sum of all the local batch sizes of each node. Recall that in conventional distributed deep learning, nodes exchange gradients and model parameters at the end of each iteration. Since the shape and size of gradients and model
Modified on	28/02/2023 15:31
Name	Communication optimization strategies for distributed deep neural network training~ A survey
Number of Coding References	18
Number of Codes Coding	2
Coverage	21.52%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	Matrix decomposition In the field of compression, an emerging method is to decom- pose a large gradient matrix into several small matrices before transmission, and then reconstruct after receiving it. Transmitting two small matrices has less communication overhead than transmitting one huge matrix. This method is feasible because of the correlation between gradients. GradiVeQ [130] exploited such linear correlations between CNN gradients and applied principal component analysis (PCA) [117] to reduce the gradients' dimensions. In addition, the proposed method GradiVeQ also enables direct aggregation of compressed gradients. ATOMO [114] is a general compression method built on top of a singular value decomposition (SVD) [46]. For a given gradient, ATOMO will produce a random unbiased gradient with minimal variance [114]. PowerSGD [112] performs a low rank decomposition with error feedback, and avoids the computationally expensive SVD step in ATOMO [114] to achieve better scalability. 3.3. Computation–communication overlap Since the gradient is generated in order from the last layer to the first layer during back propagation, there is no need to wait for the calculation of the previous layer to complete before sending the gradient of the later layer. In other words, former layers' computation is independent of the latter layers' communication, and the latter layers' parameters update is independent of the former layers [135]. Hence, we can transmit the gradient of the Lth layer while computing the (L − 1)th layer's gradient. In fact, computation–communication overlap does not really reduce the communication time, it just performs computation and communication simultaneously as possible. Therefore, this approach can combined easily with other optimization strategies. Poseidon [135] provided a wait-free backward propagation (WFBP) scheduling algorithm. WFBP makes each layer start its communication once its gradients are calculated after backward propagation. However, different layers may have different computation and communication times, which means Poseidon may not outperform than the FIFO scheduling on some specific network models.
Modified on	28/02/2023 15:32

Name	Communication optimization strategies for distributed deep neural network training~ A survey
Number of Coding References	18
Number of Codes Coding	2
Coverage	21.52%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	Network level optimization In this section, we mainly concentrate on optimizing low-level network infrastructures, including advanced centralized and decentralized architectures, messaging libraries, and network protocols. The benefits of network level optimization are intuitive. Modifying the low-level communication protocol will not have too much impact on the high-level training algorithm. However, network level optimization maybe not easy to implement, and the performance depends on the design of the distributed training system. 4.1. Logical architectures Centralized and decentralized architectures have different communication patterns and performance. We will discuss modern and advanced architectures in this subsection. 59 4.1.1. Parameter server Fig. 9(a) depicts the traditional parameter server architecture, in which the server is responsible for storing and updating global model parameters. The server is prone to network bottlenecks, especially when there are many working nodes. Tree-based parameter servers [53,62,81] alleviate such bottlenecks to a certain extent. Mai et al. [81] treat each server as the root and build a spanning tree connecting all workers. All workers are leaf nodes in the spanning tree, whereas other nodes in the tree are servers. Each worker pushes gradients to their parents. The parents aggregate all received gradients and push the results upstream towards the root where the last step of aggregation is performed. The global weights are multicasted from the top root server down to the leaf worker. Through such a spanning tree, the communication overhead, in both push and pull operations is reduced. Gupta et al. [53] proposed similar tree-based architectures like [81]. To further alleviate the network traffic, the root server broadcasts the global weights directly down a tree constructed within all workers, see in Fig. 9(b). Heish et al. [62] considered the physical distance of the parameter server and employed an intelligent communication mechanism over wide area networks (WAN) to efficiently utilize the bandwidth. In addition, some works such as Project Adam [25] and Geeps [35] improve the throughput of parameter servers by caching and by isolated communications. 4.1.2. Allreduce The key issue with traditional communication strategies is that, as the number of GPUs increases, the communication costs increase linearly. A classical implementation of Allreduce is a combination of Reduce operation followed by Broadcast which sends the result from the root to all processes. This implies a bottleneck on the root process. The optimized algorithms are based on a few principles: recursive vector halving, recursive vector doubling, recursive distance halving, recursive distance doubling, binary blocks, and ring [91]. To the best of our knowledge, Baidu first introduced a ring-based Allreduce into distributed deep learning [11]. A Ring Allreduce is made of two phases: Reduce-Scatter and Allgather; each phase includes p −1 communication steps when we use p GPUs, see in Fig. 10. Each GPU maintains its local gradients, which are equally divided into p chunks. In the reduce-scatter phase, each node sends and receives different chunks of a stored tensor. For the received chunk, each node adds it to the corresponding position in the buffer. After p−1 steps, each node holds a different part of the global result. In the all-gather phase, each node sends the part of the global result maintained by itself and receives other parts of the global result from other nodes. Each node holds a complete global result after p − 1 steps. Hence, Ring Allreduce needs a total of 2(p − 1) communication steps. The complete communication process is described in Fig. As early as 2009,
Modified on	28/02/2023 15:32

Name	Communication optimization strategies for distributed deep neural network training~ A survey
Number of Coding References	18
Number of Codes Coding	2
Coverage	21.52%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	Network protocols Traditional message level communication libraries are imple- mented based on the TCP/IP protocol, which handles data sending and receiving by sockets. Each node must create a socket object and establish a connection to the receiver before sending data. Data are processed by the operating system and encapsulated with different protocol headers until its copied into the network interface controller (NIC) buffer, as shown in Fig. 13(a). Such operations are wasteful in distributed training that needs low network latency. Therefore, high performance and low latency networks that run on top of associated network hardware (i.e., InfiniBand) attract much research attention; the two most common are remote direct memory access (RDMA) [93] and internet protocol over InfiniBand (IPoIB) [30]. As illustrated in Fig. 13(c), RDMA allows one machine to directly read and write the memory of another machine without involving the operating system, which enables high performance and low latency networking. The network interface in RDMA is called verbs, which provide two types of communication paradigms: message and memory. IPoIB, as the name implies, encapsulates IP datagrams over an InfiniBand card and enables TCP/IP applications to run on top of InfiniBand without any code modification [30]. However, as shown in Fig. 13(b), IPoIB cannot bypass the host operating system like RDMA in Fig. 13(c). With the advent of RDMA and IPoIB, tremendous works have been done on improving the performance of distributed training systems, including MXNet [75,80], TensorFlow [15,65,121], CNTK [12,31], Caffe [9] and the IBM deep learning platform [94], to leverage its high bandwidth and low latency. The memory communication paradigm is the most popular method due to its low memory demands [94,121]. Some works [31,94,121] consider communications between GPUs and explore GPU direct RDMA (GDR), which allows an RDMA NIC to access GPU memory directly without going through host memory. Gradients are aggregated in GPUs by using the GDR technique. Furthermore, Biswas et al. [15] designed an adaptive RDMA-based gRPC to dynamically adjust communication mechanisms for different message sizes in a deep learning workload. According to experimental results [65,75,121], using RDMA and IPoIB to replace the TCP/IP protocol can significantly accelerate training. In addition, RDMA performs better than IPoIB in distributed training [75,80]. Experiments by Liu et al. [80] report that compared with IPoIB (53%), the RDMA-capable network achieves near-linear (96%) speedup when scaling Inception-v3
Modified on	28/02/2023 15:32

Name	Communication optimization strategies for distributed deep neural network training~ A survey
Number of Coding References	18
Number of Codes Coding	2
Coverage	21.52%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	Patarasuk et al. [88] proved that a ring-based Allreduce has the optimal bandwidth of the Allreduce algorithms. Mikami et al. [85] proposed a 2D-Torus Allreduce topology in which the GPUs are arranged in a 2D grid. Each row contains ph GPUs while each column contains pv GPUs. There are three phases in a 2D-Torus Allreduce: Reduce-Scatter, vertical Allreduce, and Allgather. Although a 2D-Torus Allreduce has one phase more than a Ring Allreduce, its overall communication overhead is still smaller because ph and pv are less than p [85]. A similar study by Ying et al. [124] aggregates gradients in two phases with a 2D-Mesh topology, which utilizes two parallel ringbased reductions, each summing different halves of the payload along the horizontal and the vertical dimension, see in Fig. 11. The 2D-Mesh algorithm has twice the throughput of gradient aggregations than the 1D Ring Allreduce [124]. Jia et al. [66] proposed hierarchical Allreduce to resolve the problem for a small tensor communication. These researchers split all p GPUs into several groups and conducted a threephase reduction. Fig. 12 shows the three-phase operations. The first phase is a separate ring Allreduce operation in independent groups, each of which consists of k GPUs. In the second phase, a master node from each group operates Ring Allreduce to get a global result. Finally, the master node in each group broadcasts the global result to every GPU in its group. Compared with a Ring Allreduce, this three-phase hierarchical Allreduce decreases the running steps from 2(p − 1) to 4(k − 1) + 2(p/k − 1). 60 4.2. Message level libraries The performance of different parameter server systems and various Allreduce algorithms partially depends on the implementation of the message communication libraries. The parameter server is usually run on top of ZeroMQ [59] or gRPC [47]. ZeroMQ is a high performance and low latency asynchronous messaging library that supports multiple communication patterns, and gRPC is a high-performance remote procedure call (RPC) framework developed by Google. As far as we know, the most commonly used message communication libraries in the parameter server are still ZeroMQ and gRPC. At present, there are many message level communication libraries that implement various Allreduce algorithms and other collective communication algorithms efficiently, including MPI [49], Gloo [43], NCCL [64], Baidu Allreduce [11], Aluminum [40] and BlueConnect [27]. Thanks to the high performance of MPI, there are many optimizations based on MPI_Allreduce, including Horovod [100], MXNet-MPI [82] and TensorFlow-MPI [111]. To reduce communications, Horovod uses tensor fusion that sends several small tensors simultaneously [100]. NCCL implements multi-GPU and multinode collective communication primitives that are optimized for NVIDIA GPUs [64]. A series of studies by Awan et al. [8,10] optimized Bcast operation based on NCCL and CUDA-Aware MPI, respectively. BlueConnect [27] decomposes a
Modified on	28/02/2023 15:32

Name	Communication optimization strategies for distributed deep neural network training~ A survey
Number of Coding References	18
Number of Codes Coding	2
Coverage	21.52%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	Quantization As a low-precision data representation method, quantization first discretizes continuous values and maps them to different integers in a range. A standard method of quantization is to use the sign of each element to represent the gradient. As far as we know, one-bit SGD [99] has pioneered the research of communication quantization for distributed deep learning. Recently, a similar algorithm called signSGD [14] was proposed for nonconvex optimization. Because the difference between gradients processed by sign-based quantization and original gradients is too large, the quantized gradient is often a biased estimate of the original gradient [68], which makes the model converge slowly with a significant accuracy loss [50]. To address this issue, we use the error feedback technique to correct the deviations of direction accumulated in the previous iterations. Error feedback maintains a vector e to store the accumulated difference between the quantized and the original gradients. The following equations give the rationale of the error feedback, where subscript t represents the t th iteration, β is a decaying factor [118] and Q(·) is the quantizer:
Modified on	28/02/2023 15:32

Name	Communication optimization strategies for distributed deep neural network training~ A survey
Number of Coding References	18
Number of Codes Coding	2
Coverage	21.52%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	Recent trends in high-performance computing and deep learning have led to the proliferation of studies on large-scale deep neural network training. However, the frequent communication requirements among computation nodes drastically slow the overall training speeds, which causes bottlenecks in distributed training, particularly in clusters with limited network bandwidths. To mitigate the drawbacks of distributed communications, researchers have proposed various optimization strategies. In this paper, we provide a comprehensive survey of communication strategies from both an algorithm viewpoint and a computer network perspective. Algorithm optimizations focus on reducing the communication volumes used in distributed training, while network optimizations focus on accelerating the communications between distributed devices. At the algorithm level, we describe how to reduce the number of communication rounds and transmitted bits per round. In addition, we elucidate how to overlap computation and communication. At the network level, we discuss the effects caused by network infrastructures, including logical communication schemes and network protocols. Finally, we extrapolate the potential future challenges and new research directions to accelerate communications for distributed deep neural network training.
Modified on	28/02/2023 15:30

Name	Communication optimization strategies for distributed deep neural network training~ A survey
Number of Coding References	18
Number of Codes Coding	2
Coverage	21.52%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	Sparsification A limitation of quantization is that it can only compress a communication message up to 32 times, because we need at least one bit to represent numbers. However, sparsification does not have such a limitation; it only transmits values that play essential roles in the model update. A zero in the gradient means that the associated parameter is rarely updated, meaning that such gradient values have little effect on parameter updates. Therefore, it is a waste of bandwidth to transmit gradients that contains many zeros. As far as we know, Strom et al. [107] were the first to clip gradients via a static threshold, the element will not be transmitted if its absolute value below this static threshold. However, it is not very easy to select a reasonable threshold for various DNNs. To remedy this issue, subsequent works [20,41] applied local selections and dynamic threshold instead of a static threshold. A variant of threshold sparsification is the Top-K method, in which each node only transmits k largest (absolute) values [2,78,95,98]. Theoretical work [6] gave a convergence analysis of Top-K sparsification under some assumptions, and pointed out that sparsification is equivalent to a stale update. To ensure model convergence, we usually select values from gradient residuals, rather than manipulating the original gradients directly. The gradient residual is the sum of all previous gradients accumulated locally at each node [107]. Stich et al. [106] proved that with accumulated gradients, sparsified SGD had the same convergence rates as vanilla parallel SGD. Deep gradient compression (DGC) [78] introduced momentum correction, local gradient clipping, momentum factor masking, and warm-up training to achieve better performance. By these methods, DGC compresses the gradient size of ResNet-50 from 97 MB to 0.35 MB without accuracy loss [78]. Wangni et al. [115] proposed random sparsification, which drops out indices of gradients randomly. To guarantee the sparsified vector is unbiased, the remaining part is appropriately amplified. Experiments show that random dropping causes little accuracy losses of 3-layer CNNs on CIFAR-10 datasets [71]. Since quantization and sparsification are two orthogonal methods, they can be integrated for deep compression. The integration is straightforward based on a centralized architecture (Parameter Server). However, there is a challenge that needs to be resolved for sparsification in a decentralized setting. Recall that sparsification only transmits ‘‘important’’ values in the gradient, which means that each node may receive different nonzero indices (dimensions) for the same gradient [13]. Fortunately, there
Modified on	28/02/2023 15:32

Name	Communication optimization strategies for distributed deep neural network training~ A survey Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	CompareML~ A Novel Approach to Supporting Preliminary Data Analysis Decision Making
Number of Coding References	7
Number of Codes Coding	1
Coverage	8.44%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	CompareML makes it easy to compare the performance of different models by using well-known classification and regression algorithms already made available by some of the most widely used providers. It facilitates the practical application of methods and techniques of artificial intelligence that let a practising engineer decide whether they might be used to resolve hitherto intractable problems. Thus, researchers and engineering practitioners can uncover the potential of their datasets for the inference of new knowledge by selecting the most appropriate machine learning algorithm and determining the provider best suited to their data.
Modified on	18/02/2023 16:13

Name	CompareML~ A Novel Approach to Supporting Preliminary Data Analysis Decision Making
Number of Coding References	7
Number of Codes Coding	1
Coverage	8.44%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	CompareML provides researchers with the possibility of creating models of their data using the following well-known algorithms: • Linear Regression. This assumes a linear relationship between the input variables and a single output variable. The model learns by estimating the values of the coefficients used in the representation from the data available. The linear regression can be expressed as y = ax + b, where a and b are the aforementioned coefficients. • Decision Tree. Decision tree algorithms build a tree-like structure in which each node represents a question concerning an attribute. The responses to that question create new branches, expanding the structure until the end of the tree is reached, with the leaf node being the one that indicates the predicted class. • Boosted Decision Tree. This is a general method, not limited to decision trees, which consists of applying a boosting method to combine many classifiers into a new and stabler one with a smaller error. In the boosting, the predictors are made sequentially rather than independently, applying the rationale that the subsequent predictors learn from the mistakes of the previous ones. • Random Forest. This algorithm is an improvement that creates several decision trees, using bagging or some other technique, and votes for the most popular output that the trees yield. Usually, most implementations do not count the outputs directly, but sum their normalized frequencies to get the label with greatest likelihood. • Logistic Regression. This algorithm uses a more complex cost function – the 'sigmoid' or 'logistic' function – than the Linear Regression model. Input values are combined linearly using weights or coefficients to predict an output value. A key difference with Linear Regression is that the output being modeled is dichotomous (a 0 or 1) rather than numerical. • Support-Vector Machine. The data are mapped onto a highdimensional feature space so that the data points can be categorized even when the data are not otherwise linearly separable. Then, a separator is estimated for the data. The data should be transformed in such a way that a separator can be drawn as a hyperplane. As there are many possible hyperplanes, the Support Vector Machine algorithm finds a hyperplane that represents the largest separation, or margin, between classes. Of particular importance are the libraries and services from different machine learning providers that CompareML supports: • Turi Create [15], a Python™ [16] package that allows programmers to perform end-to-end large-scale data analysis and data product development. It is a distributed computation framework written in C++, developed at Carnegie Mellon University, and acquired in 2016 by Apple Inc.
Modified on	18/02/2023 16:15

Name	CompareML~ A Novel Approach to Supporting Preliminary Data Analysis Decision Making
Number of Coding References	7
Number of Codes Coding	1
Coverage	8.44%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	In addition to the software analysed, one can find in the literature related work worthy of mention with several tools and libraries, such as: • LEAC [12], an efficient library for clustering with evolutionary algorithms in the neural networks field. • Ruta [13], which implements autoencoders, neural networks that perform feature learning on data. • AutoML-Zero [14], a specific-purpose tool that simultaneously searches for all aspects of a machine learning algorithm, using basic maths operations, with the objective of reducing human bias in the search space. However, as noted above, to the best of our knowledge, there is no solution that is able to compare algorithm implementations from different providers while requiring no depth of machine learning knowledge on the part of the user.
Modified on	18/02/2023 16:15

Name	CompareML~ A Novel Approach to Supporting Preliminary Data Analysis Decision Making
Number of Coding References	7
Number of Codes Coding	1
Coverage	8.44%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	It can be applied by means of Python scripts or through a visual programming interface in which users can make use of its functionalities. The software is developed by the University of Ljubljana, and is open-source released under a GPL licence. • RapidMiner [4], [5], a data science software platform that implements data mining algorithms as operators that users can visually drag and drop to create customized dataflows. It is proprietary software developed by the RapidMiner firm, although previous versions were open source. • Knime [4], a data mining tool integrated in Eclipse (the Java Integrated Development Environment) with a visual interface in which users make use of blocks, connecting them to create dataflows that visualize, deploy, and manage machine learning models. It is open source software developed by the University of Konstanz. Table I presents a comparison of the work mentioned in this section in such aspects as: a) Support for Regression algorithms; b) Support for Classification algorithms; c) Support for Clustering algorithms; d) Support for Deep Learning algorithms; e) Model Comparison; f) Data Visualization; g) Implementation of Multiple Providers' algorithms; h) Machine Learning Knowledge required. A more detailed and specific comparison of these software packages and the Scikit-Learn library and R Programming Language can be found in [6], where the authors compare software and services that are beyond the scope of this present work. Our approach follows AutoML or Automated Machine Learning principles. AutoML [7] is an idea that consists of automating the entire pipeline or a part of a machine learning project. This is a hot topic of interest in both industry and academia, and the evaluation of its results [8] has led to several AutoML approaches and tools emerging. Some of those most widely used are: • Automated Machine Learning in PowerBI [9] was launched by Microsoft in 2019 to allow business analysts, economists, and PowerBI users in general to build machine learning models to solve business problems without their needing to have a strong background in machine learning. • PyCaret [10] is an open-source machine learning library in Python developed by Moez Ali and launched in 2019. Its main characteristic is its low-code orientation, making it simple and easy to use. • Cloud AutoML [11] is the AutoML platform launched by Google in 2018 that trains custom machine learning models for image, video, and tabular data, as well as making use of pretrained models to create natural language processing, image classification, video recognition, or structured data discovery applications.
Modified on	18/02/2023 16:15

Name	CompareML~ A Novel Approach to Supporting Preliminary Data Analysis Decision Making
Number of Coding References	7
Number of Codes Coding	1
Coverage	8.44%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	There are a large number of machine learning algorithms as well as a wide range of libraries and services that allow one to create predictive models. With machine learning and artificial intelligence playing a major role in dealing with engineering problems, practising engineers often come to the machine learning field so overwhelmed with the multitude of possibilities that they find themselves needing to address difficulties before actually starting on carrying out any work. Datasets have intrinsic properties that make it hard to select the algorithm that is best suited to some specific objective, and the ever-increasing number of providers together make this selection even harder. These were the reasons underlying the design of CompareML, an approach to supporting the evaluation and comparison of machine learning libraries and services without deep machine learning knowledge.
Modified on	18/02/2023 16:13

Name	CompareML~ A Novel Approach to Supporting Preliminary Data Analysis Decision Making
Number of Coding References	7
Number of Codes Coding	1
Coverage	8.44%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	This section reviews some related work, describes the fundamentals of the machine learning algorithms compared by CompareML, and presents the state-of-the-art of the libraries and services available on the market, describing those supported by CompareML. A. Related Work There are general-purpose machine learning tools that have dealt with similar problems: • Weka [1], a machine learning software package ideally suited for teaching and research which allows, inter alia, the comparison of algorithms for a dataset. It has recently been enriched with WekaLearning4j [2], a deep-learning package based on Deeplearning4j. The software is free under GNU GPL 3 for noncommercial purposes. • Orange [3], a machine learning software package that allows data analysis and data visualization to be performed on datasets.
Modified on	18/02/2023 16:14

Name	CompareML~ A Novel Approach to Supporting Preliminary Data Analysis Decision Making
Number of Coding References	7
Number of Codes Coding	1
Coverage	8.44%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	With the aim of answering these questions, we present CompareML, an approach that allows practitioner or research engineers to make a preliminary analysis of their data by testing different machine learning algorithms from different providers. The main goal of this approach is to allow such users to select the most suitable environment for their datasets without requiring any in-depth knowledge about machine learning. Thus, using CompareML, engineers can quickly and effortlessly compare the performance of different algorithms and providers applied to their specific datasets before deciding which to employ in their machine learning models. The resulting software supporting the CompareML approach is licensed under an MIT permissive free software licence for it to be useful, reusable, and customizable in machine learning research areas. It is readily accessible at the following URL: https://compareml.io/. The software is already very intuitive, but nevertheless there is a User Manual1 available for other researchers to answer any doubts they may have when using CompareML. Finally, scripts are available to enable full automated portability of the software. The rest of this paper is organized as follows. Section II summarizes other approaches, and lists the algorithms and providers supported by CompareML. Section III describes the approach. Section IV details the implementation of the approach and the software architecture. Section V provides an illustrative example using CompareML. Finally, Section VI draws some conclusions and describes future work.
Modified on	18/02/2023 16:14

Name	CompareML~ A Novel Approach to Supporting Preliminary Data Analysis Decision Making Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Concept drift detection for distributed multi-model machine learning systems
Number of Coding References	6
Number of Codes Coding	2
Coverage	10.46%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 20:46
Coded Text	III. PROPOSED FRAMEWORK A. Problem statement Given a system where streams of data are fed into multiple models that share intersections of features, we aim to develop a method to accomplish the following: 1) Detect drift for each model (local context) 2) Augment the labels based on shared features 3) Make the holistic features of the system more resistant to drift (improve global context) B. Detect Drift for each model Drift Detection on Distributed Datasets (QuaD) is comprised of two classical drift detection methods, DDM and KS test. It combines both methods in order to detect varying types of concept drifts and switch between labeled and unlabeled data streams. DDM is a performance-based method for drift detection and tests for the statistical distribution of its model's performance. It is measured by its error rate, p and standard deviation, s (3) and detects drift using thresholds, such as the warning level (4) and the drift level (5). The values, smin and pmin are defined in the training phase and are updated if the sample, i at time t achieves (6).
Modified on	16/02/2023 12:10

Name	Concept drift detection for distributed multi-model machine learning systems
Number of Coding References	6
Number of Codes Coding	2
Coverage	10.46%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 20:46
Coded Text	Many works focus on optimizing machine learning models during their training phase, but fail to account how these models adapt into their model-serving phase once they are deployed into real world applications. In this phase models must process through streams of data that can evolve over time and distort the relationship between incoming data, causing concept drift. This paper proposes leveraging the advantages of emerging features stores in order to improve concept drift detection on unlabeled, dynamic data streams across multiple models. Firstly, we introduce Drift Detection on Distributed Datasets (QuaD), which combines classical drift detectors to make use of labeled and unlabeled data, and create local context (i.e. per live model) and global context (i.e. across multiple models). Secondly, we propose using feature store entities, SHAP values, and Collaborative Filtering (CF) to augment unlabeled data across multiple models. To the best of our knowledge, QuaD is the first work that examines the collective behavior of concept drift across multiple models and discerns associations between models that may share a susceptibility in a dynamic setting. QuaD uses a combination of performance-based and data distribution-based drift detectors and CF to capture varying types of concept drifts for labeled and unlabeled data streams and is modeled around the data abstraction provided by emerging feature stores.
Modified on	16/02/2023 12:09

Name	Concept drift detection for distributed multi-model machine learning systems
Number of Coding References	6
Number of Codes Coding	2
Coverage	10.46%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 20:46
Coded Text	of this work is two-fold. Firstly, we introduce Drift Detection on Distributed Datasets (QuaD), which combines classical drift detectors to make use of labeled and unlabeled data, and create local context (i.e. per live model) and global context (i.e. across multiple models). Secondly, we propose using feature store entities, SHAP values, and Collaborative Filtering (CF) to augment unlabeled data across multiple models. This paper is organized as follows. Section II reviews background information related to drift detection methods, feature store, SHAP values, and CF. Section III describes our framework, QuaD. Section IV discusses metrics to evaluate our framework and Section V details future work.
Modified on	16/02/2023 12:10

Name	Concept drift detection for distributed multi-model machine learning systems
Number of Coding References	6
Number of Codes Coding	2
Coverage	10.46%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 20:46
Coded Text	This paper proposes leveraging the advantages of emerging features stores in order to improve drift detection on unlabeled, dynamic data streams across multiple ML models. The purpose 2022
Modified on	16/02/2023 12:10

Name	Concept drift detection for distributed multi-model machine learning systems
Number of Coding References	6
Number of Codes Coding	2
Coverage	10.46%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 20:46
Coded Text	Many works focus on optimizing machine learning models during their training phase, but fail to account how these models adapt into their model-serving phase once they are deployed into real world applications. In this phase models must process through streams of data that can evolve over time and distort the relationship between incoming data, causing concept drift.
Modified on	16/02/2023 12:08

Name	Concept drift detection for distributed multi-model machine learning systems
Number of Coding References	6
Number of Codes Coding	2
Coverage	10.46%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 20:46
Coded Text	Rather than measuring classifier performance metrics, data distribution-based approaches track changes in location, density, and range of the data itself. These approaches have the advantage of being able to process both labeled and unlabeled data, but are limited in the types of drift they can detect. For example, they cannot detect fixed space drift for unlabeled data without combining multiple approaches together. Hence, the trade off is that the ability to detect all types of drift is dependent on whether unlabeled data or labeled data are used. Interestingly, these drift detectors only demonstrate how to react to drift acting on a single model and/or single pair of source and target stream. They do not take into account the possibility of multiple live models acting on different streams. In the real world setting, multiple models can run simultaneously across streams and share subsets of training data. This idea of managing offline training data and online streams for multiple models and creating logically centralized features based on physically distributed data has been gaining popularity, so much so that it has given rise to several feature stores [2]–[4]. The purpose of these feature stores are to create an abstraction layer between the offline and online data and promote data reuse by removing data silos among models, reproducibility in training data, and mitigate training-serving skews. Additionally, if feature stores are for maintaining and deploying ML models in production, then SHAP values are a means to promote explainable ML. SHAP values apply cooperative game theory to distinguish each feature’s contribution to a model.
Modified on	16/02/2023 12:09

Name	Concept drift detection for distributed multi-model machine learning systems Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Continuous validation for data analytics systems
Number of Coding References	3
Number of Codes Coding	1
Coverage	4.57%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	06/01/2022 16:25
Coded Text	Data analytics systems also have a new validation goal: model accuracy, also called statistical validity. Does a model created by ML really reflect the situation in the world? When reusing data, is sample population and data collection instruments that were used still appropriate? Accuracy is fundamental to validating user needs, but is also critical for ethical assessment and legal probity. Validating model accuracy can be complicated by difficulties with interpretation. In statistics, Simpson's paradox [14] is a well-known example where associations between variables can be reversed under different groupings. These threats can defeat validation.
Modified on	23/02/2022 20:26

Name	Continuous validation for data analytics systems
Number of Coding References	3
Number of Codes Coding	1
Coverage	4.57%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	06/01/2022 16:25
Coded Text	SE is increasingly specialised [12]. A clear example of this is in the development of data analytics systems1 ('SE4ML'). Statistical machine learning (hence 'ML') lead in the integration with development practices for data analytics systems, but is now often combined with techniques from operations research and AI. As ML moved from research to widespread industrial application, there was a realisation that the bespoke algorithms written for academic publication were not necessarily scalable for large data sets nor maintainable for evolving data schemas and analysis purposes. Moreover, in industrial application there are new development artefacts to be managed, including learned statistical models, and training data sets. Since 2015, SE4ML has adapted conventional SE practices and technologies, and created new ones
Modified on	23/02/2022 20:25

Name	Continuous validation for data analytics systems
Number of Coding References	3
Number of Codes Coding	1
Coverage	4.57%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	06/01/2022 16:25
Coded Text	This trend continues through the lifecycle, into what we call 'devUsage': continuous usage validation. In addition to ensuring systems meet user needs, organisations continuously validate their legal and ethical use. The rise of end-user programming and multi-sided platforms exacerbate validation challenges. A separate trend is the specialisation of software engineering for technical domains, including data analytics. This domain has specific validation challenges. We must validate the accuracy of statistical models, but also whether they have illegal or unethical biases. Usage needs addressed by machine learning are sometimes not specifiable in the traditional sense, and statistical models are often 'black boxes'. We describe future research to investigate solutions to these devUsage challenges for data analytics systems.
Modified on	07/02/2022 23:27

Name	Curious Containers~ A framework for computational reproducibility in life sciences with support for Deep Learning applications
Number of Coding References	10
Number of Codes Coding	4
Coverage	6.83%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	In order to reproduce a computational experiment, the full stack of data, software and runtime environments must be replicated [6] as well. This has also been stated in context of FAIR data management, where the authors note, that these ‘‘[...] principles apply not only to ‘data’ in the conventional sense, but also to the algorithms, tools, and workflows that led to that data’’ [5]. It has been shown in different contexts, for example in medical image processing [7], that results may vary even if the exact same binary files are executed in different runtime environments. However, the more complex the data processing pipeline gets in terms of processing steps, tools and libraries, specific hardware requirements and data volume, the more complex gets the task to ensure reproducibility. In particular Deep Learning (DL) of medical imaging, that is currently discussed as one of the main advances in computer assisted diagnosis, shows these characteristics [8,9]. In the following sections existing software solutions are discussed, that each cover different aspects of reproducibility, some with support for Deep Learning applications. Further we introduce two complex use cases, where the goal is a platform independent reproduction of the presented experiments.
Modified on	11/02/2022 14:09

Name	Curious Containers~ A framework for computational reproducibility in life sciences with support for Deep Learning applications
Number of Coding References	10
Number of Codes Coding	4
Coverage	6.83%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	03/02/2022 14:50
Coded Text	In clinical scenarios, there is an increasing interest in complex computational experiments, as for example the training of Deep Learning models. Reproducibility is an essential property of such experiments, especially if the result contributes to a patient’s treatment. This paper introduces Curious Containers, a software framework for computational reproducibility that treats data, software and runtime environment as decentralized network resources. All experiment resources are described in a single file, using a new format that is compatible with a subset of the Common Workflow Language. Docker is used to deploy the experiment software in a container image, including arbitrary data transmission programs to connect with existing storage solutions. The framework supports Deep Learning applications, that have a high demand in storage and processing capabilities. Large datasets can be mounted inside containers via network filesystems like SSHFS based on the filesystem in userspace technology. The Nvidia-Container-Toolkit enables GPU usage. Curious Containers has been tested in two biomedical scenarios. The first use case is a Deep Learning application for tumor classification in images that requires a large dataset and a GPU. In this context, a prototypical integration of the framework with the existing Data Version Control system for exploratory Deep Learning modeling has been developed. The second use case extends an existing container image, including a scientific workflow for detection and comparison of human protein in mass spectrography data. The container image was originally developed for an archiving platform and could be extended to be compatible with both Curious Containers and cwltool,
Modified on	11/02/2022 14:03

Name	Curious Containers~ A framework for computational reproducibility in life sciences with support for Deep Learning applications
Number of Coding References	10
Number of Codes Coding	4
Coverage	6.83%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	In clinical scenarios, there is an increasing interest in complex computational experiments, as for example the training of Deep Learning models. Reproducibility is an essential property of such experiments, especially if the result contributes to a patient's treatment. This paper introduces Curious Containers, a software framework for computational reproducibility that treats data, software and runtime environment as decentralized network resources. All experiment resources are described in a single file, using a new format that is compatible with a subset of the Common Workflow Language. Docker is used to deploy the experiment software in a container image, including arbitrary data transmission programs to connect with existing storage solutions. The framework supports Deep Learning applications, that have a high demand in storage and processing capabilities. Large datasets can be mounted inside containers via network filesystems like SSHFS based on the filesystem in userspace technology. The Nvidia-Container-Toolkit enables GPU usage. Curious Containers has been tested in two biomedical scenarios. The first use case is a Deep Learning application for tumor classification in images that requires a large dataset and a GPU. In this context, a prototypical integration of the framework with the existing Data Version Control system for exploratory Deep Learning modeling has been developed. The second use case extends an existing container image, including a scientific workflow for detection and comparison of human protein in mass spectrography data. The container image was originally developed for an archiving platform and could be extended to be compatible with both Curious Containers and cwltool,
Modified on	11/02/2022 14:03

Name	Curious Containers~ A framework for computational reproducibility in life sciences with support for Deep Learning applications
Number of Coding References	10
Number of Codes Coding	4
Coverage	6.83%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	
Modified on	11/02/2022 14:10

Name	Curious Containers~ A framework for computational reproducibility in life sciences with support for Deep Learning applications
Number of Coding References	10
Number of Codes Coding	4
Coverage	6.83%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	
Modified on	11/02/2022 14:10

Name	Curious Containers~ A framework for computational reproducibility in life sciences with support for Deep Learning applications
Number of Coding References	10
Number of Codes Coding	4
Coverage	6.83%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03

Coded Text	Containerized applications Executable computer programs often require a complex set of dependencies, ranging from third party programming libraries, as well as interpreters and runtimes, to low-level APIs provided by the operating system. Ensuring compatibility of programs with other computers outside the original programming environment is therefore difficult. With container technologies like Docker, a program executable and all its dependencies, often including the userspace libraries of a certain Linux distribution, can be installed in a container image filesystem. Given the container image, the common interface for any containerized process is therefore reduced to a Linux kernel and a docker-engine to manage the container. There are two ways to publish such an appliance: On one hand, the Dockerfile, that serves as the building recipe for the container image, can be published as source code, or the built image can be uploaded to a container registry. For all experiments, both methods are provided. A specific feature is the layered structure of container images. New images can be created based on existing images, making it easy to extend the appliance by further use case specific components.
Modified on	11/02/2022 14:11

Name	Curious Containers~ A framework for computational reproducibility in life sciences with support for Deep Learning applications
Number of Coding References	10
Number of Codes Coding	4
Coverage	6.83%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Containerized applications are the current state-of-the-art method for replication of a certain runtime environment [10– 13]. FAIR principles are applicable to container images, because they are referenced by a URI when stored in a container registry. Applications installed in a container image are typically published in open source repositories such as GitLab1 or GitHub2 and can also be referenced by the URL of the specific version. However, both container registries and open source repositories offer only a very limited set of structured metadata [14]. It is up to the user to add their own metadata description as a file. For this purpose, RO-Crate3 is a metadata standard, that allows for description and linking of experiment resources. Applications with a command line interface (CLI) can be consistently described using the command line tool specification of the Common Workflow Language (CWL) [15]. CWL is an established standard in the bioinformatics domain and various runtime engines already support it. The reference implementation cwltool4 executes experiments on a local computer with specific input files and parameters defined in a job description. Experiments are either standalone command line tools or scientific workflows composed of multiple processing steps. The execution is performed locally, while input data and containers might be downloaded from remote sites. They are defined in the so-called job description, also known as the input object. CWL-based research platforms like Arvados5 or REANA [16] are more advanced solutions. They include multiple server-side components for dedicated data storage, workflow engines connected to compute clusters and graphical management interfaces. Both solutions make use of container technologies and aim for reproducibility, but data resources for an experiment are described offline. For this purpose, REANA defines a custom YAML file format to reference local file paths of the data and the CWL document. Client tools then upload the data to the respective platform
Modified on	11/02/2022 14:09

Name	Curious Containers~ A framework for computational reproducibility in life sciences with support for Deep Learning applications
Number of Coding References	10
Number of Codes Coding	4
Coverage	6.83%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	In clinical scenarios, there is an increasing interest in complex computational experiments, as for example the training of Deep Learning models. Reproducibility is an essential property of such experiments, especially if the result contributes to a patient's treatment. This paper introduces Curious Containers, a software framework for computational reproducibility that treats data, software and runtime environment as decentralized network resources. All experiment resources are described in a single file, using a new format that is compatible with a subset of the Common Workflow Language. Docker is used to deploy the experiment software in a container image, including arbitrary data transmission programs to connect with existing storage solutions. The framework supports Deep Learning applications, that have a high demand in storage and processing capabilities. Large datasets can be mounted inside containers via network filesystems like SSHFS based on the filesystem in userspace technology. The Nvidia-Container-Toolkit enables GPU usage. Curious Containers has been tested in two biomedical scenarios. The first use case is a Deep Learning application for tumor classification in images that requires a large dataset and a GPU. In this context, a prototypical integration of the framework with the existing Data Version Control system for exploratory Deep Learning modeling has been developed. The second use case extends an existing container image, including a scientific workflow for detection and comparison of human protein in mass spectrography data. The container image was originally developed for an archiving platform and could be extended to be compatible with both Curious Containers and cwltool,
Modified on	11/02/2022 14:03

Name	Curious Containers~ A framework for computational reproducibility in life sciences with support for Deep Learning applications
Number of Coding References	10
Number of Codes Coding	4
Coverage	6.83%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	The Curious Containers (CC) software project17 aims to sup- port reproducible experiments [14]. Four main design criteria can be formulated: (a) The applications are provided as container images, so-called appliances, (b) data is loaded directly into the running container, (c) user authentication on the components like data storage and compute environment is supported, and (d) all information required to describe the experiment is stored in a single file, that can be published and shared. Since its first release, the components have been completely redesigned to better fulfill the FAIR guiding principles and to be easier to employ. The new execution engines CC-FAICE and CC-Agency are described in Section 3.2. For the implementation of CC the Docker container technology has been chosen, because it provides a mature implementation with widespread industry support, for example as a base technology of Kubernetes.18 Most importantly, it is supported by CWL (cf. Section 2.1) and Nvidia (cf. Section 2.5.4). In addition, Docker uses a client–server model that allows CC-FAICE and CC-Agency to
Modified on	11/02/2022 14:10

Name	Curious Containers~ A framework for computational reproducibility in life sciences with support for Deep Learning applications
Number of Coding References	10
Number of Codes Coding	4
Coverage	6.83%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	where the experiment is executed remotely. With respect to reproducibility, these solutions have complementary strengths and weaknesses. Local file paths, as used in the latter platforms, do not fullfill the FAIR requirement of unique identifiers. A complete reproduction of an experiment is tied to the platform. On the other hand, cwltool allows for local execution, so that everyone could reproduce an experiment that uses remote data and container images. However, the support of specific transmission and authentication protocols depends on the CWL runtime implementation and therefore might not be portable. For example, the Extensible Neuroimage Archive Toolkit6 (XNAT) [17], an established storage solution for collaborative image-based biomedical research, provides a complex HTTP API, that usually requires multiple requests to handle data access. A typical CWL runtime would not implement a data transfer mechanism for this type of specialized API. A specialized data transmission program could be part of a larger workflow, but there would be no dedicated way of handing over sensitive authentication credentials to this program. This is of high interest, as for example many datasets available on XNAT Central – one of the largest biomedical imaging repositories worldwide – require access granted [18]. In our earlier work we have introduced the Curious Con- tainer (CC) software project. It included the scheduling software CC-Server to run containerized experiments in an OpenStack cluster [19]. CC introduced an extensible data connector concept, where data from external sources was transferred into a running application container via a network. Based on this technology, an integration with XNAT for analysis of multidimensional biosignal data in sleep research (~300 MB per file) was demonstrated [20]. The FAICE tool-suite [14] was an approach to provision experiment runtimes like CC-Server or cwltool in virtual machines, such that experiments could be reproduced locally in a controlled environment. While this tool has increased the reproducibility of the addressed experiments, it was built around a custom description format that did not follow community standards. While the data transfer was suitable for small files, access to large image datasets could not be supported. For DL applications, specific tools for reproducible training are available. A lightweight solution to support arbitrary modifications in the pipeline and the input data is Data Version Control (DVC)7. It is an open source version control system for machine learning projects that approaches reproducibility of data-driven applications from a different perspective. It integrates with git8, a well-known version control system for source code, and extends the concept for data versioning. Different processing steps, such as data preparation, training and evaluation, are linked by their input and output data. The resulting graph of dependencies is used to only run a processing step if dependencies or outputs of this step are missing or outdated, based on checksums. However, DVC is not based on CWL, but defines its own data format; and is not suited to describe runtime environments. Another solution to support DL applications is offered by Hopsworks9, a software stack for centralized platforms similar to Arvados and REANA. Hopsworks tightly integrates with a wide variety of technologies for storage, processing and user management. It supports certain DL software libraries like Tensorflow10 and PyTorch11 Applications must be prepared to integrate with the ecosystem, for example by importing the Hops-Py12 package for Python. This
Modified on	11/02/2022 14:10

Name	Data Cleaning for Accurate, Fair, and Robust Models~ A Big Data - AI Integration Approach
Number of Coding References	15
Number of Codes Coding	4
Coverage	16.86%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	06/01/2022 15:37
Coded Text	A popular solution is to make the model training more robust. Another approach that is gaining interest is sanitizing the poisoned data before it is used in training. Data poisoning attacks have recently become more sophisticated [9], and
Modified on	14/02/2022 14:33

Name	Data Cleaning for Accurate, Fair, and Robust Models~ A Big Data - AI Integration Approach
Number of Coding References	15
Number of Codes Coding	4
Coverage	16.86%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	06/01/2022 15:37
Coded Text	As a running example, suppose we are cleaning a set of training examples in Table 1 (small for illustration purposes). This data is not clean in the sense that e2 and e3 refer to the same person because their ages are the same and Joe is an abbreviation of Joseph. (In comparison, e4 and e5 are not the same person because they have very different ages.) In addition, e6 has an unusually-high age, which can be viewed as an incorrect value. Hence, cleaning this data may involve merging e2 and e3 to a single example e23 and fixing or removing e6's age.
Modified on	14/02/2022 14:33

Name	Data Cleaning for Accurate, Fair, and Robust Models~ A Big Data - AI Integration Approach
Number of Coding References	15
Number of Codes Coding	4
Coverage	16.86%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	06/01/2022 15:37
Coded Text	Beyond removing duplicates, data cleaning can be any general process like HoloClean [11]. Data sanitization can also employ more sophisticated defenses [9]. Of course, one should carefully analyze the possible interactions between each cleaning and sanitization combination.
Modified on	14/02/2022 14:29

Name	Data Cleaning for Accurate, Fair, and Robust Models~ A Big Data - AI Integration Approach
Number of Coding References	15
Number of Codes Coding	4
Coverage	16.86%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	06/01/2022 15:37
Coded Text	MLCLEAN Since data cleaning, unfairness mitigation, and data sanitization are ultimately preprocessing the same dataset, it makes sense to unify them. The naïve approach of applying each technique independently in any sequence can be problematic for several reasons. Simply ignoring the dependencies between preprocessing techniques may result in incorrect results. For example, ifwe reweigh examples and then attempt to remove duplicates, then the reweighing may need to be done again to ensure fairness. Moreover, running one operation at a time may have efficiency issues due to redundant operations on the data. 3.1 Basic Architecture
Modified on	14/02/2022 14:29

Name	Data Cleaning for Accurate, Fair, and Robust Models~ A Big Data - AI Integration Approach
Number of Coding References	15
Number of Codes Coding	4
Coverage	16.86%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	06/01/2022 15:37
Coded Text	More recently, there are mitigation techniques for fixing unfairness, which can be done before (pre-processing), during (in-processing), or after (post-processing) model training [2]. These techniques typically tradeoff some model accuracy in order to improve model fairness. Among them, we focus on the pre-processing approach where the example weights are adjusted (i.e., reweighed [3]) to maximize fairness. For example, a simplified reweighing technique for demographic parity is to increase the weights of positivelylabeled examples in sensitive groups whose ratio ofweighted positive labels is lower than other groups.
Modified on	14/02/2022 14:33
Name	Data Cleaning for Accurate, Fair, and Robust Models~ A Big Data - AI Integration Approach
Number of Coding References	15
Number of Codes Coding	4
Coverage	16.86%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	06/01/2022 15:37
Coded Text	We contend that it is time to extend the notion of data cleaning for modern machine learning needs. We identify dependencies among the data preprocessing techniques and propose MLClean, a unified data cleaning framework that integrates the techniques and helps train accurate and fair models. This work is part of a broader trend of Big data – Artificial Intelligence
Modified on	07/02/2022 23:21

Name	Data Cleaning for Accurate, Fair, and Robust Models~ A Big Data - AI Integration Approach
Number of Coding References	15
Number of Codes Coding	4
Coverage	16.86%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	06/01/2022 15:36
Coded Text	
Modified on	07/02/2022 23:22

Name	Data Cleaning for Accurate, Fair, and Robust Models~ A Big Data - AI Integration Approach
Number of Coding References	15
Number of Codes Coding	4
Coverage	16.86%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	06/01/2022 15:36

Coded Text	. While many techniques have been proposed to improve the model training process (in-processing approach) or the trained model itself (post-processing), we argue that the most effective method is to clean the root cause of error: the data the model is trained on (pre-processing). Historically, there are at least three research communities that have been separately studying this problem: data management, machine learning (model fairness), and security. Although a significant amount of research has been done by each community, ultimately the same datasets must be preprocessed, and there is little understanding how the techniques relate to each other and can possibly be integrated.
Modified on	07/02/2022 23:20

Name	Data Cleaning for Accurate, Fair, and Robust Models~ A Big Data - AI Integration Approach
Number of Coding References	15
Number of Codes Coding	4
Coverage	16.86%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	06/01/2022 15:36
Coded Text	Data Sanitization The machine learning and security communities are actively studying the problem of robust machine learning against adversarial data in critical applications including spam filtering, autonomous driving, and cybersecurity. A major problem is that the training data is often collected from external data sources, which are vulnerable to attacks by malicious actors [9]. A popular solution is to make the model training more robust. Another approach that is gaining interest is sanitizing the poisoned data before it is used in training. Data poisoning attacks have recently become more sophisticated [9], and there is an arms race on developing better defenses to stop them as well. Data poisoning can also be done on the test data where the same sanitization techniques can apply. Data sanitization may conflict with data cleaning.
Modified on	07/02/2022 23:21
Name	Data Cleaning for Accurate, Fair, and Robust Models~ A Big Data - AI Integration Approach
Number of Coding References	15
Number of Codes Coding	4
Coverage	16.86%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	06/01/2022 15:36
Coded Text	MLCLEAN Since data cleaning, unfairness mitigation, and data sanitization are ultimately preprocessing the same dataset, it makes sense to unify them. The naïve approach of applying each technique independently in any sequence can be problematic for several reasons. Simply ignoring the dependencies between preprocessing techniques may result in incorrect results. For example, ifwe reweigh examples and then attempt to remove duplicates, then the reweighing may need to be done again to ensure fairness. Moreover, running one operation at a time may have efficiency issues due to redundant operations on the data. 3.1 Basic Architecture We present MLClean, an extended data cleaning framework that takes into account the dependencies of the three preprocessing techniques and integrates them to produce clean, unbiased, and sanitized data (see architecture in Figure 1). Data sanitization can be viewed as an extreme version ofdata cleaning and thus be executed together in one component. The unfairness mitigation component comes afterwards because, while data sanitization and cleaning may affect the bias of data, reweighing examples only changes the example weights and does not effect the correctness of sanitization and cleaning on the other features.
Modified on	07/02/2022 23:22

Name	Data Cleaning for Accurate, Fair, and Robust Models~ A Big Data - AI Integration Approach
Number of Coding References	15
Number of Codes Coding	4
Coverage	16.86%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	06/01/2022 15:36
Coded Text	We compare and identify dependencies among the three data preprocessing techniques and discuss how data cleaning can possibly be extended to the other preprocessing techniques. 2.1 Traditional Data Cleaning Data cleaning [4] originates from the data management community and has been studied for decades. Traditionally, there is a focus on cleaning structured data with schema at scale where integrity constraints, denial constraints, and functional dependencies need to be satisfied. In addition, duplicates must be removed, and values need to be corrected to be within certain ranges or to exist in external data sources. More recently, there are efforts to improve machine learning accuracy [5] and data validation techniques for machine learning pipelines [10]. However, these techniques do not resolve the pressing issues of model fairness or model robustness against adversarial data.
Modified on	07/02/2022 23:21

Name	Data Cleaning for Accurate, Fair, and Robust Models~ A Big Data - AI Integration Approach
Number of Coding References	15
Number of Codes Coding	4
Coverage	16.86%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	06/01/2022 15:36
Coded Text	We contend that it is time to extend the notion of data cleaning for modern machine learning needs. We identify dependencies among the data preprocessing techniques and propose MLClean, a unified data cleaning framework that integrates the techniques and helps train accurate and fair models. This work is part of a broader trend of Big data – Artificial Intelligence
Modified on	07/02/2022 23:20

Name	Data Cleaning for Accurate, Fair, and Robust Models~ A Big Data - AI Integration Approach
Number of Coding References	15
Number of Codes Coding	4
Coverage	16.86%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Research gap in Maintaining ML systems
Created on	14/02/2022 14:34
Coded Text	Unfortunately, little is known how the different data preprocessing approaches depend on each other and can be used together when a dataset is dirty, biased, and adversarial. In addition, we cannot assume that solving one problem will automatically solve the others. From a data management standpoint, we contend that it is a good time to extend the data cleaning problem for the pressing needs of modern machine learning for accurate, fair, and robust model training.
Modified on	14/02/2022 14:34
Name	Data Cleaning for Accurate, Fair, and Robust Models~ A Big Data - AI Integration Approach
Number of Coding References	15
Number of Codes Coding	4
Coverage	16.86%
Folder Location	Codes\\Snowballing
Parent Name	Codes\\Snowballing\\Snowballing
Created on	14/02/2022 14:36
Coded Text	[4] Xu Chu, Ihab F. Ilyas, Sanjay Krishnan, and Jiannan Wang. 2016. Data Cleaning: Overviewand Emerging Challenges. In SIGMOD. 2201–2206.
Modified on	14/02/2022 14:39

Name	Data Cleaning for Accurate, Fair, and Robust Models~ A Big Data - AI Integration Approach
Number of Coding References	15
Number of Codes Coding	4
Coverage	16.86%
Folder Location	Codes\\Snowballing
Parent Name	Codes\\Snowballing\\Snowballing
Created on	14/02/2022 14:36
Coded Text	[9] Pang Wei Koh, Jacob Steinhardt, and Percy Liang. 2018. Stronger Data Poisoning Attacks Break Data Sanitization Defenses. CoRR abs/1811.00741 (2018). [10] Neoklis Polyzotis, Sudip Roy, Steven Euijong Whang, and Martin Zinkevich. 2017. Data Management Challenges in Production Machine Learning. In SIGMOD. 1723–1726. [11] Theodoros Rekatsinas, Xu Chu, Ihab F. Ilyas, and Christopher Ré. 2017. HoloClean: Holistic Data Repairs with Probabilistic Inference. PVLDB 10, 11 (2017), 1190–1201.
Modified on	14/02/2022 14:39

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	06/01/2022 15:37
Coded Text	We cover the recent CleanML [8] work, which systematically studies the impact of data cleaning on the accuracy of the model trained on that data.
Modified on	14/02/2022 14:43

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	06/01/2022 16:31
Coded Text	While there is a plethora of data visualization techniques, we focus on the ones that are most relevant to machine learning. Facets, a component of TFX, shows various statistics of datasets that are relevant for machine learning. More advanced tools include SeeDB [17], which can repeatedly generate possible visualizations that are of interest. This approach has the problem of false positives, so hypothesis testing started to be used in systems like CUDE [19] to guarantee the statistical significance of the findings. Data validation focuses on finding problems in the data that affect the machine learning pipeline. TensorFlow Data
Modified on	05/06/2022 17:52

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	10/02/2022 12:05
Coded Text	data discovery, data augmentation, and data generation. Data discovery is the problem of indexing and searching datasets that may exist in corporate data lakes or the Web. Data augmentation is related, but focuses on complementing existing datasets by integrating them with external data. If there is not enough data around, the last resort is to take matters in one’s hand and create datasets using crowdsourcing or synthetic data generation techniques.
Modified on	05/06/2022 17:35

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	10/02/2022 12:05
Coded Text	First, data acquisition is the problem of finding the right datasets for training models. For example, as the number of datasets around us is increasing rapidly, searching for the right ones itself becomes a challenge. Second, data labeling is necessary in all supervised learning applications. Since manual labeling can be expensive, various scalable techniques have been proposed using semi-supervised learning, crowdsourcing, and weak supervision. Finally, one can also improve the quality of existing data or use transfer learning to re-use existing models instead of training from scratch.
Modified on	05/06/2022 17:34

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	10/02/2022 12:05
Coded Text	The traditional approach for labeling is semi-supervised learning where the idea is to use existing labels to predict the other labels as much as possible. Other approaches include crowdsourcing where workers manually label examples or more advanced techniques like active learning where the manual labeling is done only for examples that are most likely to benefit the model’s accuracy. Most recently, weak supervision is on the rise where the idea is to (semi-)automatically generate labels that are not perfect (therefore called “weak” labels), but at scale where the larger volume may compensate for the lower label quality. Weak supervision is useful in applications where there are few or no labels to start with.
Modified on	05/06/2022 17:37

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering\Data cleaning
Created on	05/06/2022 18:28
Coded Text	It is common for the training data to contain various er- rors. Machine learning platforms like TensorFlow Extended (TFX) [1] have data validation components to detect such data errors in advance using data visualization and schema generation techniques.
Modified on	05/06/2022 17:46
Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering\Data cleaning
Created on	05/06/2022 18:28
Coded Text	Missing Data. Because missing data can reduce the statistical power and produce biased estimates, data imputation has been an active research topic in statistics and machine learning.
Modified on	05/06/2022 18:26

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering\Data cleaning
Created on	05/06/2022 18:28
Coded Text	Noisy or Missing Labels. Regarding noisy labels, recent techniques are mainly categorized into loss correction and sample selection. The former estimates the confidence of a label for each sample and adjusts the loss for the sample based on its label confidence during backward propagation. The latter also estimates the confidence of a label for each sample and includes the samples in training only if their label confidence is above some threshold. Recently, the sample selection approach becomes dominant, and a hybrid of the two approaches has been proposed [15]. Regarding missing labels, semi-supervised learning builds a model from a mixture of labeled and unlabeled data, by adopting unsupervised loss or collaborating with mix-up augmentation for unlabeled data.
Modified on	05/06/2022 18:25
Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering\Data cleaning
Created on	05/06/2022 18:28
Coded Text	we focus on the deep learning 3431 techniques for time-series data. Because the recurrent neural network (RNN) is typically used for time-series data, the RNN family has been extended to receive the context about missing values. The most well-known technique is GRU-D [3] developed for medical data analysis, and many variations such as Temporal Belief Memory are available.
Modified on	05/06/2022 18:26

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering\Data cleaning
Created on	05/06/2022 18:28
Coded Text	While there is a plethora of data visualization techniques, we focus on the ones that are most relevant to machine learning. Facets, a component of TFX, shows various statistics of datasets that are relevant for machine learning. More advanced tools include SeeDB [17], which can repeatedly generate possible visualizations that are of interest. This approach has the problem of false positives, so hypothesis testing started to be used in systems like CUDE [19] to guarantee the statistical significance of the findings.
Modified on	05/06/2022 17:55

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	05/06/2022 18:19
Coded Text	More recently, another line of research is to mitigate the unfairness, which can be done in largely three places: before model training on the data (pre-processing), during model training (in-processing), and after model training (post-processing). All of these approaches typically have a trade-off between accuracy and fairness, as the two objectives do not necessarily align. In this tutorial, we focus on pre-processing and in-processing approaches where examples are re-weighted to improve model fairness. For example, demographic parity can be improved by increasing the weights of positive examples of a sensitive group that has a lower positive prediction rate than other groups.
Modified on	05/06/2022 18:19

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	06/01/2022 15:36
Coded Text	Compared to traditional machine learning, there is less need for feature engineering, but more need for significant amounts of data. We thus go through stateof-the-art data collection techniques for machine learning. Then, we cover data validation and cleaning techniques for improving data quality. Even if the data is still problematic, hope is not lost, and we cover fair and robust training techniques for handling data bias and errors. We believe that the data management community is well poised to lead the research in these directions. The presenters have extensive experience in developing machine learning platforms and publishing papers in top-tier database, data mining, and machine learning venues.
Modified on	07/02/2022 23:17

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	06/01/2022 15:36
Coded Text	Data cleaning has a long history of removing various well-defined errors by satisfying integrity constraints including key constraints, domain constraints, referential integrity constraints, and functional dependencies. Unfortunately, only focusing on fixing the data does not necessarily guarantee the best model accuracy. We cover the recent CleanML [8] work, which systematically studies the impact of data cleaning on the accuracy of the model trained on that data. The conclusions are twofold: data cleaning does not necessarily improve the model accuracy, and performing model selection can at least reduce any negative effects where the data cleaning may harm model accuracy. Hence, we cover recent data cleaning techniques that are specifically geared towards improving model accuracy.
Modified on	05/06/2022 18:05

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	06/01/2022 15:36
Coded Text	Data poisoning has recently become a serious issue be- cause changing a fraction of the training data, which may come from an untrusted source, may alter the model's behavior. Compared to dirty data, there is a malicious intention of making the model fail. Early work focused on specific applications like spam detection and sensors. More recent studies are more general, but still tend to focus on specific models. It is unclear if there will be anything close to a unifying solution. The notion of data sanitization was introduced in 2008 [4] where attacks were assumed to occur in relatively confined time intervals, and the sanitization techniques used training metadata. More recently, adversarial machine learning, which attempts to fool models through malicious inputs (e.g., adversarial images), has become one of the most popular topics in machine learning.
Modified on	14/02/2022 14:07

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	06/01/2022 15:36
Coded Text	While there is a vast literature on data cleaning, not all of the techniques are beneficial to machine learning [8]. In addition, recent machine learning issues including data poisoning need to be addressed as well. Even after carefully preparing the data, the data quality may still be problematic, and we need to cope with biased, dirty, or missing data using fair and robust model training [14, 15].
Modified on	14/02/2022 14:42

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	03/02/2022 10:47
Coded Text	Compared to traditional machine learning, there is less need for feature engineering, but more need for significant amounts of data. We thus go through stateof-the-art data collection techniques for machine learning. Then, we cover data validation and cleaning techniques for improving data quality. Even if the data is still problematic, hope is not lost, and we cover fair and robust training techniques for handling data bias and errors. We believe that the data management community is well poised to lead the research in these directions. The presenters have extensive experience in developing machine learning platforms and publishing papers in top-tier database, data mining, and machine learning venues.
Modified on	07/02/2022 23:17

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	03/02/2022 10:47
Coded Text	data becomes a first class citizen, on par with code and 2) machine learning models become the new software. As a result, we need to rethink how to develop software. It is well known that 80–90% of the time spent on machine learning development is data preparation. Unfortunately, the opposite effort is spent on machine learning algorithms instead of the actual bottleneck [16].
Modified on	04/06/2022 17:29

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	03/02/2022 10:47
Coded Text	Data collection for machine learning [13]. The techniques in the leaf nodes that are at least partially proposed by the data management community are highlighted in italic blue font. A key observation is that there is an convergence of techniques between the data management and machine learning communities, so one needs to know both sides to understand the overall research landscape.
Modified on	14/02/2022 14:12
Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	04/06/2022 17:30
Coded Text	data quality has a profound impact on model accuracy where even the best machine learning algorithms cannot perform well without good data or at least handling dirty data during training.
Modified on	04/06/2022 17:30

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	04/06/2022 17:30
Coded Text	However, recent studies [8] show that only focusing on improving the data does not necessarily benefit machine learning accuracy. In addition, in a machine learning point of view, we also need to address critical issues including data sanitization against poisoning and model fairness, which are actively studied in the security and machine learning communities, respectively.
Modified on	05/06/2022 17:48

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	04/06/2022 17:30
Coded Text	In comparison to traditional machine learning, feature engineering is less of a concern, but there is instead a need for large amounts of training data. Unfortunately, the lack of data is one of the main reasons many industries are reluctant to adopt deep learning [16] (the other reason being lack of explainability).
Modified on	04/06/2022 17:31

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	04/06/2022 17:30
Coded Text	It is widely agreed that real-world datasets are dirty and erroneous despite the data cleaning process. For example, because data labeling is done manually in many cases, incorrect or missing labels are, in fact, very common; the proportion of incorrect labels is reported to be 8–38% in several real-world datasets [15]. Besides, especially in multivariate time-series data, missing values are unavoidable because of its high input rate and sensor malfunction. Thus, many deep learning techniques have been developed to consider the existence of data noises and errors, which are more critical in deep learning than in conventional machine learning as a deep neural network (DNN) completely memorizes such noises and errors because of its high expressive power.
Modified on	05/06/2022 18:21

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	04/06/2022 17:30
Coded Text	While there is a vast literature on data cleaning, not all of the techniques are beneficial to machine learning [8]. In addition, recent machine learning issues including data poisoning need to be addressed as well. Even after carefully preparing the data, the data quality may still be problematic, and we need to cope with biased, dirty, or missing data using fair and robust model training [14, 15].
Modified on	04/06/2022 17:34

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering\Data cleaning
Created on	05/06/2022 18:08
Coded Text	Compared to dirty data, there is a malicious intention of making the model fail. Early work focused on specific applications like spam detection and sensors. More recent studies are more general, but still tend to focus on specific models. It is unclear if there will be anything close to a unifying solution. The notion of data sanitization was introduced in 2008 [4] where attacks were assumed to occur in relatively confined time intervals, and the sanitization techniques used training metadata. More recently, adversarial machine learning, which attempts to fool models through malicious inputs (e.g., adversarial images),
Modified on	05/06/2022 18:13

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering\Data cleaning
Created on	05/06/2022 18:08
Coded Text	Unfortunately, only focusing on fixing the data does not necessarily guarantee the best model accuracy. We cover the recent CleanML [8] work, which systematically studies the impact of data cleaning on the accuracy of the model trained on that data. The conclusions are twofold: data cleaning does not necessarily improve the model accuracy, and performing model selection can at least reduce any negative effects where the data cleaning may harm model accuracy.
Modified on	05/06/2022 18:08

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering\Data validation
Created on	05/06/2022 17:57
Coded Text	Data validation focuses on finding problems in the data that affect the machine learning pipeline. TensorFlow Data 3430 Validation [2] automatically generates database schemas from previous datasets and validates future datasets with the schema. There are largely three types of data errors that are validated. First, the data may be dirty; e.g., there may be duplicate country codes that are in upper and lower case letters. Second, the data may change as the data source itself evolves; e.g., the unit of a numeric feature may change from days to hours. Third, the data may simply be missing due to possible bugs in the data source; e.g., the title information of documents may be missing for a large portion of examples. For each schema violation, the user can either fix the data or the schema itself.
Modified on	05/06/2022 17:57

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering\Model training
Created on	05/06/2022 18:15
Coded Text	Even after collecting the right data and cleaning it, data quality may still be an issue during model training. We present three directions of research. First, we cover fair model training techniques that can address data bias, which results in discriminative model behavior. Next, we cover robust model training techniques that cope with dirty data and still produce accurate results. Finally, we explore representation learning techniques for transforming the data in the best embedding format for model training.
Modified on	05/06/2022 18:15

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Research gap in Maintaining ML systems
Created on	14/02/2022 14:34
Coded Text	Data cleaning has a long history of removing various well-defined errors by satisfying integrity constraints including key constraints, domain constraints, referential integrity constraints, and functional dependencies. Unfortunately, only focusing on fixing the data does not necessarily guarantee the best model accuracy.
Modified on	14/02/2022 14:44

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Research gap in Maintaining ML systems
Created on	14/02/2022 14:34
Coded Text	Data poisoning has recently become a serious issue be- cause changing a fraction of the training data, which may come from an untrusted source, may alter the model's behavior.
Modified on	14/02/2022 14:45

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Research gap in Maintaining ML systems
Created on	14/02/2022 14:34
Coded Text	We cover the recent CleanML [8] work, which systematically studies the impact of data cleaning on the accuracy of the model trained on that data.
Modified on	14/02/2022 14:44

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Snowballing
Parent Name	Codes\\Snowballing\\Snowballing
Created on	14/02/2022 14:36
Coded Text	[8] P. Li, X. Rao, J. Blase, Y. Zhang, X. Chu, and C. Zhang. CleanML: A benchmark for joint data cleaning and machine learning. CoRR, abs/1904.09483, 2019.
Modified on	14/02/2022 14:45

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Snowballing
Parent Name	Codes\\Snowballing\\Snowballing
Created on	14/02/2022 14:36
Coded Text	Even after carefully preparing the data, the data quality may still be problematic, and we need to cope with biased, dirty, or missing data using fair and robust model training [14, 15].
Modified on	14/02/2022 14:42

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Snowballing
Parent Name	Codes\\Snowballing\\Snowballing
Created on	14/02/2022 14:36
Coded Text	We cover the recent CleanML [8] work, which systematically studies the impact of data cleaning on the accuracy of the model trained on that data.
Modified on	14/02/2022 14:43

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Snowballing
Parent Name	Codes\\Snowballing\\Snowballing
Created on	14/02/2022 14:46
Coded Text	2] E. Breck, M. Zinkevich, N. Polyzotis, S. Whang, and S. Roy. Data validation for machine learning. In MLSys, 2019
Modified on	14/02/2022 14:46

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Snowballing
Parent Name	Codes\\Snowballing\\Snowballing
Created on	14/02/2022 14:46
Coded Text	A. Kumar, M. Boehm, and J. Yang. Data management in machine learning: Challenges, techniques, and systems. In SIGMOD, pages 1717–1722, 2017.
Modified on	14/02/2022 14:48

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Snowballing
Parent Name	Codes\\Snowballing\\Snowballing
Created on	14/02/2022 14:46
Coded Text	N. Polyzotis, S. Roy, S. E. Whang, and M. Zinkevich. Data management challenges in production machine learning. In SIGMOD, pages 1723–1726, 2017.
Modified on	14/02/2022 14:47

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Snowballing
Parent Name	Codes\\Snowballing\\Snowballing
Created on	14/02/2022 14:46
Coded Text	Y. Roh, G. Heo, and S. E. Whang. A survey on data collection for machine learning: a big data - AI integration perspective. IEEE TKDE, 2019.
Modified on	14/02/2022 14:49

Name	Data collection and quality challenges for deep learning
Number of Coding References	38
Number of Codes Coding	16
Coverage	25.55%
Folder Location	Codes\\Snowballing
Parent Name	Codes\\Snowballing\\Snowballing
Created on	05/06/2022 17:40
Coded Text	M. Stonebraker and E. K. Rezig. Machine learning and big data: What is important? IEEE Data Eng. Bull., 2019.
Modified on	05/06/2022 17:40

Name	DeepSpeed-MoE~ Advancing Mixture-of-Experts Inference and Training to Power Next-Generation AI Scale
Number of Coding References	7
Number of Codes Coding	2
Coverage	10.74%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	However, sustaining the growth in model size is getting more and more difficult due to the increasing compute requirements. For example, the largest single dense model in existence as of Dec 2021, the Megatron-Turing NLG 530B model, took around 3 months to train on over 2000 A100 GPUs on the NVIDIA Selene Supercomputer, consuming over 3 million GPU hours (Microsoft & Nvidia, 2021). Another 3 to 5 times of increase in dense model size would be infeasible within a reasonable timeframe. Given the exorbitant compute resources required to train the state-of-art models, a natural question to ask is: “Is it possible to make non-trivial improvement to model quality without increasing the compute cost?” Or equivalently, “Is it possible to produce model with similar quality using 3 to 5 times less resources?” There have been numerous efforts to reduce the compute requirements to train large models without sacrificing model quality. To this end, architectures based on Mixture-ofExperts (MoE) (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2021a) have paved a promising path, enabling sub-linear compute requirements with respect to the model parameters and allowing for improved model quality without increasing training cost. However, MoE based models have their own set of challenges that limit their use in a wide range of real world scenarios: • Limited Scope The scope of MoE based models in the NLP area is primarily limited to encoder-decoder models and sequence-to-sequence tasks, with limited work done in exploring its application in other domains. Application of MoE to auto-regressive natural language generation (NLG) like GPT-3 and MT-NLG 530B, where the compute cost of training state-of-art language models can be orders of magnitude higher than for encoder-decoder models, is less explored. • Massive Memory Requirements While MoE models require less compute to achieve the same model qualAuthor contributions listed at Appendix A. DeepSpeed-MoE ity as their dense counterparts, they need significantly more number of parameters. For example, the MoE based Switch-Base model has 10x more parameters than T5large (7.4B vs 0.74B) and still it does not have the same model quality when compared across a wide range of downstream tasks (Fedus et al., 2021a). In other words, MoE based models have a much lower “parameter efficiency” compared to quality-equivalent dense models. For instance, if we scale the dense model size to MT-NLG equivalent with 530B parameters, achieving similar quality with MoE based model might need a model with over 5 trillion parameters (assuming the 10x scaling still holds), which would require over 5K GPUs to just fit the model states for training. • Limited Inference Performance Due to the large model size and poor parameter efficiency mentioned above, fast inference of MoE based models is even more challenging. On one hand, the larger parameter size requires more GPUs to fit, and multi-gpu inference technology is not designed to work with MoE based models. On the other hand, as inference is often memory bandwidth bound, MoE based models, which can be 10x larger than their dense equivalent, could require 10x higher achievable memory bandwidth to achieve similar inference latency as the dense models. Despite the promising and non-trivial reduction in training cost, these above mentioned challenges severely limits the practical applicability of MoE.
Modified on	28/02/2023 12:58

Name	DeepSpeed-MoE~ Advancing Mixture-of-Experts Inference and Training to Power Next-Generation AI Scale
Number of Coding References	7
Number of Codes Coding	2
Coverage	10.74%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	As the training of giant dense models hits the boundary on the availability and capability of the hardware resources today, Mixture-of-Experts (MoE) models have become one of the most promising model architectures due to their significant training cost reduction compared to qualityequivalent dense models. Their training cost saving is demonstrated from encoder-decoder models (prior works) to a 5x saving for auto-aggressive language models (this work). However, due to the much larger model size and unique architecture, how to provide fast MoE model inference remains challenging and unsolved, limiting their practical usage. To tackle this, we present DeepSpeedMoE, an end-to-end MoE training and inference solution, including novel MoE architecture designs and model compression techniques that reduce MoE model size by up to 3.7x, and a highly optimized inference system that provides 7.3x better latency and cost compared to existing MoE inference solutions. DeepSpeed-MoE offers an unprecedented scale and efficiency to serve massive MoE models with up to 4.5x faster and 9x cheaper inference compared to quality-equivalent dense models. We hope our innovations and systems help open a promising path to new directions in the large model landscape, a shift from dense to sparse MoE models, where training and deploying higher-quality models with fewer resources becomes more widely possible.
Modified on	28/02/2023 12:57

Name	DeepSpeed-MoE~ Advancing Mixture-of-Experts Inference and Training to Power Next-Generation AI Scale
Number of Coding References	7
Number of Codes Coding	2
Coverage	10.74%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	DeepSpeed-MoE for NLG: Reducing the Training Cost of Language Models by 5 Times Transformer-based natural language generation (NLG) models offer convincing solutions to a broad range of language tasks. Given the tremendous compute and energy requirements for training NLG family of models, we explore the opportunities that MoE presents to reduce their training cost. We show that MoE-based NLG model can achieve 5x reduction in training cost to achieve the same model quality of a dense NLG model. Model Architecture We studied MoE architecture for the GPT-3 like NLG model (Brown et al., 2020). The follow- 2.0 2.2 2.4 2.6 2.8 3.0 3.2 350M dense 350M+MoE-128 1.3B dense 1.3B+MoE-128 6.7B dense 0 60B 120B 180B Tokens Figure 1: Token-wise validation loss curves for dense and MoE NLG models with different model sizes. ing models are selected: 350M/1.3B/6.7B (24/24/32 layers, 1024/2048/4096 hidden size, 16/16/32 attention heads). We use “350M+MoE-128” to denote a MoE model that uses 350M dense model as the base model and adds 128 experts on every other feedforward layer. We use a top-1 gating function to activate a single expert in the MoE layer for each token. Therefore, during both training and inference, our MoE model will have the same number of parameters to be activated for each token as their dense part (Figure 3 (left)). Training and Evaluation Settings We pre-trained both the dense and MoE models on 128 NVIDIA Ampere A100 GPUs (Azure ND A100 instances), using the same training data as described in (Microsoft & Nvidia, 2021). We use 300B tokens to train both dense and MoE models. In addition to the pre-training validation loss, we employ 6 zero-shot evaluation tasks to compare the final model quality: LAMBADA (Paperno et al., 2016), PIQA (Bisk et al., 2020), BoolQ (Wang et al., 2019), RACE-h (Lai et al., 2017), TriviaQA (Joshi et al., 2017), WebQs (Berant et al., 2013). Appendix B summarizes the hyperparameters for training the dense and MoE models. For dense models we followed the hyperparameters from the GPT-3 work. MoE models have two additional hyperparameters: the number of experts per MoE layer, and a coefficient when adding the MoE layer losses to the total training loss. MoE Leads to Better Quality for NLG Models Figure 1 shows that the validation loss of the MoE models is significantly better than their dense counter parts (e.g., 1.3B+MoE128 versus 1.3B dense). In addition, MoE models are on par with the validation loss of the dense models with 4-5x larger base (e.g., 1.3B+MoE-128 versus 6.7B dense). Furthermore, the model quality is also on par in terms of the zero-shot evaluation as shown in Table 1. Same Quality with 5x Less Training Cost Adding MoE to the NLG model significantly improves the model quality. In addition, these experts do not change the compute requirements of the model as each token is only processed by a single expert. A 1.3B+MoE-128 model requires roughly the same amount of training compute as 1.3B dense model, while offering much better model quality. Furthermore, the 1.3B+MoE-128 model can achieve the model quality of the 6.7B dense model at the training cost of 1.3B parameter dense model, resulting in a 5x training compute reduction:
Modified on	28/02/2023 12:58

Name	DeepSpeed-MoE~ Advancing Mixture-of-Experts Inference and Training to Power Next-Generation AI Scale
Number of Coding References	7
Number of Codes Coding	2
Coverage	10.74%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	DeepSpeed-MoE Inference: Serving MoE Models at Unprecedented Scale and Speed Optimizing inference latency and cost is crucial for MoE models to be useful in practice. During inference, the batch size is generally small, so the inference latency of an MoE model depends primarily on the time it takes to load the model parameters from the main memory, contrasting with the conventional belief that lesser compute should lead to faster inference. Therefore, the MoE inference performance depends on two main factors: the overall model size and the overall achievable memory bandwidth.
Modified on	28/02/2023 12:59

Name	DeepSpeed-MoE~ Advancing Mixture-of-Experts Inference and Training to Power Next-Generation AI Scale
Number of Coding References	7
Number of Codes Coding	2
Coverage	10.74%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Expert, Tensor and Data parallelism We use tensor parallelism, referred in Figure 5 as tensor-slicing (for non-expert parameters) and expert-slicing (for expert parameters), to split individual parameters across multiple GPUs to leverage the aggregate memory bandwidth across GPUs. However, tensor parallelism can only scale efficiently to a few GPUs due to communication overhead and fine-grained parallelism. To address this, we use expert parallelism in conjunction with tensor parallelism to scale experts parameters to hundreds of GPUs. Expert parallelism does not reduce computation granularity of individual operators, therefore allowing our system to leverage aggregate memory bandwidth across hundreds of GPUs. To scale the non-expert computation to the same number of GPUs, we use data parallelism at no communication overhead. Hierarchical All-to-all Hierarchical tree-based algorithms are often used with communication collectives like allreduce, broadcast, etc to reduce the number of communication hops. We implement a hierarchical all-to-all as a two-step process with a data-layout transformation, followed by an intra-node all-to-all, followed by a second data-layout transformation, and a final inter-node all-to-all. This reduces the communication hops from O (p) to O(G+ p/G), where G is the number of GPUs in a node and p is the total number of GPU devices. Figure 6 shows the design overview of this implementation. Despite the 2x increase in communication volume, this hierarchical implementation allows for better scaling for small batch sizes as communication at this message size is more latency-bound than bandwidth-bound. Parallelism Coordinated Communication All-to-all communication latency increases linearly with the number of GPUs. To avoid this linearly increase, we leverage the data redundancy created by tensor-slicing (Rajbhandari et al., 2020) to limit the GPUs that participate in all-to-all . Since each GPU in tensor parallelism contains the same data, the all-to-all communication can be limited within GPUs with the same tensor-parallelism rank. This reduces the total
Modified on	28/02/2023 13:00

Name	DeepSpeed-MoE~ Advancing Mixture-of-Experts Inference and Training to Power Next-Generation AI Scale
Number of Coding References	7
Number of Codes Coding	2
Coverage	10.74%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	In an effort to make MoE practical, accessible and applicable, in this paper, we address these challenges by offering three corresponding solutions: • We expand the scope of MoE based models to autoregressive NLG tasks, demonstrating training cost reduction of 5x to achieve same model quality for models like GPT-3 and MT-NLG (see Section 3). These results not only demonstrate clear opportunities to reduce the cost of training massive NLG models, but also opens up the possibilities to reach much higher next-generation model quality under the limitation of current generation hardware resource. • We improve parameter efficiency of MoE based models by developing a novel MoE architecture that we call Pyramid-Residual MoE (PR-MoE). PR-MoE is a hybrid dense and MoE model created using residual connections, while applying experts only where they are most effective. PR-MoE can reduce parameters by up to 3x with no change to model quality. In addition, we leverage staged knowledge distillation to create a distilled version of PR-MoE, which we call Mixture-of-Students (MoS), that further reduce model size by 12.5% while retaining 99.3% performance of the teacher (see Section 4). • We develop DeepSpeed-MoE inference system, a highly optimized MoE inference system which enables efficient scaling of inference workloads on hundreds of GPUs, providing up to 7.3x reduction in inference latency and cost when compared with existing MoE inference solutions (see Section 5). It offers ultra-fast inference latencies (under 25 ms) for trillion-parameter MoE models. DeepSpeed-MoE also offers up to 4.5x faster and 9x cheaper inference for MoE models compared to qualityequivalent dense models by combining both system and model optimizations. Together, our innovations and systems enable MoE to be a more effective and economic alternative comparing to dense models, achieving significantly lower training and inference cost while obtaining the same model quality. We hope DeepSpeed-MoE helps open a promising path to new directions in the large model landscape, a shift from dense to sparse MoE models, where training and deploying higherquality models with fewer resources becomes more widely possible. Software The generic DeepSpeed-MoE end-to-end framework for training and inference of MoE-based models is open-sourced as part of the DeepSpeed software, and the experiments presented in this paper were conducted on the Microsoft Azure AI platform. Please find the code, tutorials, and documents at DeepSpeed GitHub (https: //github.com/microsoft/DeepSpeed) and website (https://www.deepspeed.ai/).
Modified on	28/02/2023 12:58

Name	DeepSpeed-MoE~ Advancing Mixture-of-Experts Inference and Training to Power Next-Generation AI Scale
Number of Coding References	7
Number of Codes Coding	2
Coverage	10.74%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	While PR-MoE and MoS (previous section) help to reduce the MoE model size while preserving the model accuracy, we now present our multi-GPU MoE inference system that maximizes and leverages the aggregated memory bandwidth across hundreds of GPUs to speed up inference at an unprecedented scale. 5.1. Design of DeepSpeed-MoE Inference System MoE inference performance is an interesting paradox. From the best-case view, each input token of an MoE model only activates a single expert at each MoE layer, resulting in a critical data path that is equivalent to the base dense model size (e.g. 1.3 billion), orders-of-magnitude smaller than the actual model size (52 billion). From the worst-case view, the aggregate parameters needed to process a batch of tokens (e.g., a sentence or a paragraph of text) can be as large as the full model size since different tokens could activate different experts making it challenging to achieve short latency and high throughput. The design of DeepSpeed-MoE inference system (Figure 5) ensures that we steer the performance toward the best case. Here, we offer a brief summary of the main optimizations. For more details, please refer to (Rajbhandari et al., 2022).
Modified on	28/02/2023 12:59

Name	DeepSpeed-MoE~ Advancing Mixture-of-Experts Inference and Training to Power Next-Generation AI Scale Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Dependency Tracking for Risk Mitigation in Machine Learning (ML) Systems
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Dependency Tracking for Risk Mitigation in Machine Learning (ML) Systems Imported Notes (2)
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	DistRDF2ML - Scalable Distributed In-Memory Machine Learning Pipelines for RDF Knowledge Graphs
Number of Coding References	11
Number of Codes Coding	2
Coverage	8.04%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 23:01
Coded Text	Creating pipelines based on generic transformers (scikit-learn [21], Spark MLlib [15]) opens up the possibility for a wide range of software developers, data scientists, and knowledge engineers to develop machine learning models for their data. While the usual machine learning models implemented in Spark MLlib (BigDL [4], Analytics Zoo2) assume fixed length numeric feature vectors, a holistic framework for developing distributed machine learning pipelines is missing for handling large scale RDF knowledge graphs. Such a framework can empower the already existing Apache Spark clusters to operate natively on RDF knowledge graph data. SANSA (Semantic Analytics Stack) [14] uses the scalable processing capability of Apache Spark to allow distributed RDF processing. However, it did not offer a pipeline for ML solutions. Knowledge graphs do not provide a native representation that fits these requirements. Therefore, there are two possibilities: First, latent embeddings can be computed or learned (TransE [2], DistMult [27], RDF2Vec [22]). Alternatively, relevant features can be extracted and digitized.
Modified on	07/02/2022 23:04

Name	DistRDF2ML - Scalable Distributed In-Memory Machine Learning Pipelines for RDF Knowledge Graphs
Number of Coding References	11
Number of Codes Coding	2
Coverage	8.04%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 23:01
Coded Text	This paper presents DistRDF2ML, the generic, scalable, and distributed framework for creating in-memory data preprocessing pipelines for Spark-based machine learning on RDF knowledge graphs. This framework introduces software modules that transform large-scale RDF data into ML-ready fixed-length numeric feature vectors. The developed modules are optimized to the multimodal nature of knowledge graphs. DistRDF2ML provides aligned software design and usage principles as common data science stacks that offer an easy-to-use package for creating machine learning pipelines. The modules used in the pipeline, the hyper-parameters and the results are exported as a semantic structure that can be used to enrich the original knowledge graph. The semantic representation of metadata and machine learning results offers the advantage of increasing the machine learning pipelines' reusability, explainability, and reproducibility. The entire framework of DistRDF2ML is open source, integrated into the holistic SANSA stack, documented in scala-docs, and covered by unit tests. DistRDF2ML demonstrates its scalable design across different processing power configurations and (hyper-)parameter setups within various experiments.
Modified on	07/02/2022 23:01

Name	DistRDF2ML - Scalable Distributed In-Memory Machine Learning Pipelines for RDF Knowledge Graphs
Number of Coding References	11
Number of Codes Coding	2
Coverage	8.04%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	
Modified on	07/02/2022 23:06

Name	DistRDF2ML - Scalable Distributed In-Memory Machine Learning Pipelines for RDF Knowledge Graphs
Number of Coding References	11
Number of Codes Coding	2
Coverage	8.04%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	4.1 Pipeline The proposed DistRDF2ML framework offers the construction of end-to-end Apache Spark 3.x pipelines (see Figures 1, 2, 3). The principal pipeline contains the following modules: • Knowledge graph reader • SPARQL creation • SparqlFrame feature extractor • SmartVectorAssembler • Spark MLlib machine learning • Semantification of machine learning results and metadata • Result exporter
Modified on	07/02/2022 23:06

Name	DistRDF2ML - Scalable Distributed In-Memory Machine Learning Pipelines for RDF Knowledge Graphs
Number of Coding References	11
Number of Codes Coding	2
Coverage	8.04%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	Advantages ofdistributed processing: In the experiments, we ob- Figure 8: Dataset (-size) and number movies on cluster and local execution vs processing time serve that the development and use of Apache Spark as one fundamental backbone offer high scalability far beyond the capabilities local computers can achieve. The significant advantage, in addition, is that the same code can be used on any spark cluster instance and will make use out of the available resources. In Figure 7 we have shown the first-hand-on guideline to optimize the spark setup, which can improve performance.
Modified on	07/02/2022 23:07

Name	DistRDF2ML - Scalable Distributed In-Memory Machine Learning Pipelines for RDF Knowledge Graphs
Number of Coding References	11
Number of Codes Coding	2
Coverage	8.04%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	GeneralityofApproach: Within paragraph 6.7, DistRDF2ML shows its various application opportunities as enabler and glue between large scale RDF knowledge graphs and existing numeric featurebased Apache Spark MLlib machine learning approaches. The usage of DistRDF2ML within the examples of Clustering, Classification, and Regression shows how easily applicable and adjustable these transformers are. These transformers will majorly decrease the entry hurdle for creating baseline Apache Spark RDF ML pipelines. If the transformer does not suit the users' purpose, it can be used as starting point for further development cause of its fully open source nature.
Modified on	07/02/2022 23:07

Name	DistRDF2ML - Scalable Distributed In-Memory Machine Learning Pipelines for RDF Knowledge Graphs
Number of Coding References	11
Number of Codes Coding	2
Coverage	8.04%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	In conclusion, in this paper, we propose a framework that en- ables the intuitive creation ofRDF knowledge graph-based machine learning pipelines in multiple areas. Our framework, which is natively executable on Apache Spark clusters, enables many machine learning applications to scale with data size and computation performance. The contributions of this work are as follows: • An open source framework for distributed machine learning pipelines on RDF knowledge graphs • Generic modules for the creation of explainable and context preserving feature vectors • Various sample machine learning pipelines documented and testable within Databricks3 notebooks, executable sample classes, and unit tests • A software architecture which its semantic data machine learning pipelines is aligned with pipelines in established data analytics libraries • Reproducibility over semantic annotation of pipeline metadata and processed results
Modified on	07/02/2022 23:04

Name	DistRDF2ML - Scalable Distributed In-Memory Machine Learning Pipelines for RDF Knowledge Graphs
Number of Coding References	11
Number of Codes Coding	2
Coverage	8.04%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	Scalability ofapproach: The experiments show the scalable na- ture of Apache Spark and its modules. We stuck to Apache Spark design principles when creating the transformers. This programming principle paid off with excellent scalability behaviour among various (hyper-)parameters (see section 6).
Modified on	07/02/2022 23:07

Name	DistRDF2ML - Scalable Distributed In-Memory Machine Learning Pipelines for RDF Knowledge Graphs
Number of Coding References	11
Number of Codes Coding	2
Coverage	8.04%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	The DistRDF2ML framework proposes modules to preprocess RDF knowledge graph data for state-of-the-art artificial intelligence, data mining, and data analytics pipelines. The framework is fully available for the community as an open source GitHub release [25] and as an extension of the holistic SANSA stack to materialize big RDF data. DistRDF2ML enables various domains to empower their distributed Apache Spark clusters to process ML pipelines on RDF data. The performance and scalability of the framework have been evaluated on various (hyper-)parameter setups to present the advantages of the efficient data processing pipelines in data-intensive computing (see Section 6). The lack of existing extensions for processing RDF data within Spark MLlib pipelines was a significant motivation to develop this framework. The feature extraction over semi-automated query execution by Literal2Feature [17] and SparqlFrame modules also enable non-native semantic developers to construct explainable feature extraction pipelines. The feature and knowledge extraction based on queries and processed through the SmartVectorAssembler allow multi-modal content preserving representations of feature vectors. Within various partner projects, the framework shows its enabling nature to create with few lines of intuitive code ML pipelines for various domains.
Modified on	07/02/2022 23:06

Name	DistRDF2ML - Scalable Distributed In-Memory Machine Learning Pipelines for RDF Knowledge Graphs
Number of Coding References	11
Number of Codes Coding	2
Coverage	8.04%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44

Coded Text	These are exciting data representations for various applications and enable data integration through semantic web[1] standards via IRIs [19]. The data sizes of real-world linked open knowledge graphs are often infeasible to process in main memory of a single computer, as the data volume often exceeds the available main memory resources by far. A single computer cannot be arbitrarily scaled in performance regarding the number of processor cores and the amount of main memory. Additionally, specialized components with higher performance become more expensive as they are not in demand by the common consumer market, resulting in disproportionate costs. Cluster computation can solve this problem in terms of available main memory and required CPU processing power by combining the performance ofmultiple servers as nodes within a cluster. Many distributed computing frameworks have been developed in recent years and are in productive use, like Apache Spark [8, 15] and Apache Flink [6]. In the area of machine learning, pipelines, frameworks, and libraries that realize both modular pipeline creations and explainable AI have become very popular. These libraries and frameworks are open source and available to interested parties
Modified on	07/02/2022 23:03

Name	DistRDF2ML - Scalable Distributed In-Memory Machine Learning Pipelines for RDF Knowledge Graphs
Number of Coding References	11
Number of Codes Coding	2
Coverage	8.04%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	This paper presents DistRDF2ML, the generic, scalable, and distributed framework for creating in-memory data preprocessing pipelines for Spark-based machine learning on RDF knowledge graphs. This framework introduces software modules that transform large-scale RDF data into ML-ready fixed-length numeric feature vectors. The developed modules are optimized to the multimodal nature of knowledge graphs. DistRDF2ML provides aligned software design and usage principles as common data science stacks that offer an easy-to-use package for creating machine learning pipelines. The modules used in the pipeline, the hyper-parameters and the results are exported as a semantic structure that can be used to enrich the original knowledge graph. The semantic representation of metadata and machine learning results offers the advantage of increasing the machine learning pipelines' reusability, explainability, and reproducibility. The entire framework of DistRDF2ML is open source, integrated into the holistic SANSA stack, documented in scala-docs, and covered by unit tests. DistRDF2ML demonstrates its scalable design across different processing power configurations and (hyper-)parameter setups within various experiments.
Modified on	07/02/2022 23:02
Name	DLHub~ Simplifying publication, discovery, and use of machine learning models in science
Number of Coding References	13
Number of Codes Coding	4
Coverage	7.88%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	Concerns about reproducibility are having a profound effect on research <27>. While reproducibility initiatives have primarily focused on making data and experimental processes available to reproduce findings, there is a growing interest in making computational methods available as well <28; 29; 30>. Unlike sharing software products, there is little guidance for sharing ML models and their artifacts (e.g., weights, hyper-parameters, and training/test 3 sets). Without publishing these artifacts, it is almost impossible to verify or build upon published results. Thus, there is a growing need to develop standard ML model packages and metadata schema, and to provide rich model repositories and serving platforms that can be used to reproduce published results.
Modified on	28/02/2023 15:42

Name	DLHub~ Simplifying publication, discovery, and use of machine learning models in science
Number of Coding References	13
Number of Codes Coding	4
Coverage	7.88%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 22:50
Coded Text	
Modified on	07/02/2022 22:55

Name	DLHub~ Simplifying publication, discovery, and use of machine learning models in science
Number of Coding References	13
Number of Codes Coding	4
Coverage	7.88%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 22:50
Coded Text	
Modified on	07/02/2022 22:57

Name	DLHub~ Simplifying publication, discovery, and use of machine learning models in science
Number of Coding References	13
Number of Codes Coding	4
Coverage	7.88%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 22:50
Coded Text	DLHub offers a unique model serving infrastructure that is capable of serv- ing many different types of models on a range of distributed computing resources including clouds, clusters, and supercomputers. The serving infrastructure builds upon funcX <19>—a distributed Function-as-a-Service platform developed specifically to support remote and distributed execution of functions. DLHub implements a flexible pipeline that converts deposited models into servables—executable containers that implement a standard DLHub execution interface, irrespective of the model type, and includes the trained model, model components (e.g., training weights, hyperparameters), and dependencies (e.g., system or Python packages).
Modified on	07/02/2022 22:53

Name	DLHub~ Simplifying publication, discovery, and use of machine learning models in science
Number of Coding References	13
Number of Codes Coding	4
Coverage	7.88%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 22:50
Coded Text	In this paper, we present the Data and Learning Hub for science (DLHub) and outline initial experiences applying this learning system to science. While many learning systems focus on building and training ML models <14; 15; 3>, DLHub is a unique learning system that is designed to support the publication and serving of ML models in science. DLHub is implemented as a cloud-hosted service that allows researchers to deposit and share models of various types, in-
Modified on	07/02/2022 22:52

Name	DLHub~ Simplifying publication, discovery, and use of machine learning models in science
Number of Coding References	13
Number of Codes Coding	4
Coverage	7.88%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 22:50
Coded Text	There is a growing need for “learning systems” to support various phases in the ML lifecycle. While others have focused on supporting model development, training, and inference, few have focused on the unique challenges inherent in science, such as the need to publish and share models and to serve them on a range of available computing resources. In this paper, we present the Data and Learning Hub for science (DLHub), a learning system designed to support these use cases. Specifically, DLHub enables publication of models, with descriptive metadata, persistent identifiers, and flexible access control. It packages arbitrary models into portable servable containers, and enables low-latency, distributed serving of these models on heterogeneous compute resources. We show that DLHub supports low-latency model inference comparable to other model serving systems including TensorFlow Serving, SageMaker, and Clipper, and improved performance, by up to 95%, with batching and memoization enabled.
Modified on	07/02/2022 22:51

Name	DLHub~ Simplifying publication, discovery, and use of machine learning models in science
Number of Coding References	13
Number of Codes Coding	4
Coverage	7.88%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 22:49
Coded Text	However, scientific use of ML has specialized requirements, including the following. Publication, citation, and reuse: The scholarly process is built upon a common workflow of publication, peer review, and citation. Progress is dependent on being able to locate, verify, and extend prior research, and careers are built upon publications and citation. As scholarly objects, ML models should be subject to similar publication, review, and citation models. Lacking standard methods for doing so, (a) many models associated with published literature are not available <23>; and (b) researchers adopt a range of ad hoc methods (from customized websites to GitHub)
Modified on	07/02/2022 22:54

Name	DLHub~ Simplifying publication, discovery, and use of machine learning models in science
Number of Coding References	13
Number of Codes Coding	4
Coverage	7.88%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 22:49
Coded Text	Model in the loop: Scientific analyses often involve multiple steps, such as the staging of input data for pre-processing and normalization, extraction of pertinent features, execution of one or more ML models, application of uncertainty quantification methods, post-processing
Modified on	07/02/2022 22:54

Name	DLHub~ Simplifying publication, discovery, and use of machine learning models in science
Number of Coding References	13
Number of Codes Coding	4
Coverage	7.88%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 22:49
Coded Text	Reproducibility: Concerns about reproducibility are having a profound effect on research <27>. While reproducibility initiatives have primarily focused on making data and experimental processes available to reproduce findings, there is a growing interest in making computational methods available as well <28; 29; 30>.
Modified on	07/02/2022 22:54

Name	DLHub~ Simplifying publication, discovery, and use of machine learning models in science
Number of Coding References	13
Number of Codes Coding	4
Coverage	7.88%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 22:49
Coded Text	Research infrastructure: While industry and research share common re- quirements for scaling inference, the execution landscape differs. Researchers often want to use multiple (often heterogeneous) parallel and distributed computing resources to develop, optimize, train, and execute models. Examples include: laboratory computers, campus clusters, national cyberinfrastructure (e.g., XSEDE <31>, Open Science Grid <32>), supercomputers, and clouds. They often have their own resources that they would like to use for inference. Thus, learning systems need to support execution on different resources and enable migration between resources. Scalability: Large-scale parallel and distributed computing environments enable ML models to be executed at unprecedented scale. Researchers require learning systems that simplify training and inference on enormous scientific datasets and that can be parallelized to exploit large computing resources. Low latency: ML is increasingly being used in real-time scientific pipelines, for example to process and respond to events generated from sensor networks; classify and prioritize transient events from digital sky surveys for exploration; and to perform error detection on images obtained from X-ray light sources. There is a need in each case for low latency, near real-time ML inference for anomaly/error detection and for experiment steering purposes. As both the number of devices and data generation rates continue to grow, there is also a need to be able to execute many inference tasks in parallel, whether on centralized or "edge" computers. Research ecosystem: Researchers rely upon a large and growing ecosys- tem of research-specific software
Modified on	07/02/2022 22:54

Name	DLHub~ Simplifying publication, discovery, and use of machine learning models in science
Number of Coding References	13
Number of Codes Coding	4
Coverage	7.88%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 22:49
Coded Text	There is a growing need for "learning systems" to support various phases in the ML lifecycle. While others have focused on supporting model development, training, and inference, few have focused on the unique challenges inherent in science, such as the need to publish and share models and to serve them on a range of available computing resources. In this paper, we present the
Modified on	07/02/2022 22:52

Name	DLHub~ Simplifying publication, discovery, and use of machine learning models in science
Number of Coding References	13
Number of Codes Coding	4
Coverage	7.88%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	22/07/2022 11:44
Coded Text	Scalability We evaluate the scalability of DLHub by sending inference requests to con- nected funcX endpoints. We report only the scalability of the endpoints as the other components are hosted on AWS services, which are known to be highly scalable. We deployed funcX endpoints on Theta, Cooley, and PetrelKube with a varying number of containers for each of the six servables. We set 64 and 12 containers per node on Theta and Cooley, respectively. On PetrelKube, Kubernetes will automatically manage the container deployment to nodes. We sent 1000 requests of each servable to the endpoints directly and measured the completion time of all requests. Figure 6 shows the results on different platforms. Our results show that the endpoint can easily deploy hundreds of containers for each servable. However, we also see that scalability is dependent on the model. For example, when serving Matminer-featurize requests, throughput increases rapidly up to ~16 containers, after which more containers have dimin- 23 ishing benefits and throughput is saturated. As expected, servables that execute for shorter periods of time show a less significant benefit from additional containers, presumably because task dispatch latencies dominate execution time.
Modified on	22/07/2022 11:44

Name	DLHub~ Simplifying publication, discovery, and use of machine learning models in science
Number of Coding References	13
Number of Codes Coding	4
Coverage	7.88%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	22/07/2022 11:44
Coded Text	We used CIFAR-10 and Inception to compare the serving performance of TensorFlow Serving, SageMaker, Clipper, and DLHub when hosted on the PetrelKube Kubernetes cluster For TensorFlow Serving, we export the trained models and use the standard tensorflow model server. For SageMaker, we use the SageMaker service to create the models before exporting the model as a Docker container and deploy it as a Pod on PetrelKube. For Clipper, we use its Kubernetes container manager to deploy it on PetrelKube and register the CIFAR-10 and Inception models. For DLHub we use a funcX endpoint deployed on PetrelKube. To standardize our measurements we remove network overheads by submitting tasks directly to each platform and report the average time from 24 100 requests for each model and platform. TensorFlow Serving provides two model serving APIs: REST and gRPC. SageMaker also supports serving TensorFlow models through TensorFlow Serving or its native Flask framework. In our experiments, we explore all possible APIs and frameworks, i.e., TFServing-REST, TFServing-gRPC, SageMakerTFServing-REST, SageMaker-TFServing-gRPC, SageMaker-Flask. In addition, as Clipper supports caching, we evaluate it with and without caching. Figure 7 shows the invocation times and request times of CIFAR-10 and In- ception using each serving system. We see that the servables invoked through the TensorFlow Serving framework (i.e., TFServing-gRPC, TFServing-REST and SageMaker-TFServing) outperform those using other serving systems (SageMakerFlask and DLHub) in terms of both invocation time and request time. This is because the core tensorflow model server, implemented in C++, outperforms Python-based systems. gRPC leads to slightly better performance than REST due to the overhead of the HTTP protocol. Clipper performs similarly to other systems when caching is enabled. The clipper caching model maintains a cache at the query frontend (on a PetrelKube pod) and therefore performance is not significantly better than other systems as it requires that the request be transmitted to the query frontend. DLHub's performance is comparable to the other Python-based serving infrastructures.
Modified on	22/07/2022 11:48

Name	DOLL~ Distributed OnLine Learning Using Preemptible Cloud Instances
Number of Coding References	6
Number of Codes Coding	2
Coverage	14.69%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	04/02/2022 22:14
Coded Text	Most large-scale ML implementations scale to large amounts of data by utilizing multiple servers or virtual machines (VMs) that iteratively compute model updates on local data that are periodically synchronized. Due to the complexity of managing the resulting computing infrastructure, many companies run their ML jobs on external cloud providers' servers. However, cloud resources can be expensive, particularly for large ML jobs with long runtimes. A particularly popular method to limit the costs of train- ing ML jobs is to utilize preemptible cloud instances. These may be interrupted at the cloud provider's discretion, but they are significantly (up to 90%) cheaper than conventional on-demand instances. Most studies of these ML methods, however, assume the availability of large datasets at training time. In practice, training data may arrive at irregular intervals and models may be trained online as new data samples arrive, e.g., when monitoring data from IoT sensors. While some software frameworks like Apache Kafka can feed online data arrivals to ML algorithms, they provide little insight into the resulting costs of ML training. We extend prior work on provisioning preemptible instances to analyze available pools of data in order to run online ML on incoming datastreams, which presents new challenges due to the need to carefully handle data arrivals.
Modified on	16/02/2023 09:30

Name	DOLL~ Distributed OnLine Learning Using Preemptible Cloud Instances
Number of Coding References	6
Number of Codes Coding	2
Coverage	14.69%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	04/02/2022 22:14

Coded Text	Research Challenges and Our Contributions: When pools of data are readily available, the bottleneck to distributed ML training often lies in the time required for each VM to compute its model updates. In our scenario, however, the arrival rate of incoming data may also bottleneck data processing. An intuitive strategy would then be for each VM to process each data point as it arrives. However, since arrivals at di↵erent VMs may not be coordinated, synchronizing the model parameters at each VM between data arrivals may introduce additional delays, while asynchronous SGD methods can lead to slow convergence [1].
Modified on	16/02/2023 09:30

Name	DOLL~ Distributed OnLine Learning Using Preemptible Cloud Instances
Number of Coding References	6
Number of Codes Coding	2
Coverage	14.69%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	DOLL uses a batching and grouping process to limit the synchronization delay, which naturally realizes traditional mini-batch SGD so as to provide provable model convergence guarantees. Handling online data arrivals becomes particularly chal- lenging when we use preemptible instances to compute model updates. Existing methods utilizing preemptible instances
Modified on	16/02/2023 09:31

Name	DOLL~ Distributed OnLine Learning Using Preemptible Cloud Instances
Number of Coding References	6
Number of Codes Coding	2
Coverage	14.69%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44

Coded Text	for ML jobs largely focus on mitigating training interruptions [2] and their e↵ects on model convergence [3]. When used on datastreams, we face an additional challenge of interruptions pausing the data arrival process, which impedes the rate at which we can compute model updates and thus model convergence. Thus, one should ensure that preemptions do not happen "too often," e.g., by computing some updates on on-demand instances. Our work is the first to optimize the number of preemptible VMs used and demonstrate that we can meet ML convergence guarantees.
Modified on	16/02/2023 09:31

Name	DOLL~ Distributed OnLine Learning Using Preemptible Cloud Instances
Number of Coding References	6
Number of Codes Coding	2
Coverage	14.69%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	System Architecture System components: We consider a parameter server (PS) architecture [1,3] in which N distributed workers each receive a stream of data (see Figure 1). The goal is to train a model h (z,w), with input features z and trainable parameters w. Each worker is either an on-demand or preemptible cloud instance of the same type, and receives a stream (series) of data samples independent and identically distributed (i.i.d) across time and data streams. The arrival times of each data sample in the N data streams are modelled as N i.i.d renewal processes, i.e., each interarrival time is i.i.d, both within and between data streams. Training process: As is typical in ML training, we use a variant of stochastic gradient descent (SGD). The process proceeds iteratively for updates s =1, 2, .. .: each worker i continually accumulates data samples, sending a batch indicator to the PS every time it accumulates b samples since the last indicator. The PS periodically sends a model update trigger to all workers with the current model parameters ws. Each worker with a batch Sk then computes the gradient gk of the loss function. Workers wait to begin the computation until receiving the update trigger to ensure they are working from the correct global model. Data samples may continue to arrive during the gradient computations. The resulting K gradients are sent to the PS, which aggregates them to perform gradient descent. The next iteration commences once the PS sends another update trigger. We use ⇣ µX, λX = µX 1 , σ2 X ⌘ to denote the mean, recipro- cal mean, and variance of a random variable X. Handling preemptions: Workers may be preempted by the cloud provider at any time. While preempted, they cannot receive data samples or compute model updates; however, any samples received before the preemption begins are retained, through standard fault tolerance mechanisms [4]. If a preemption occurs between the time of sending a batch
Modified on	16/02/2023 09:32

Name	DOLL~ Distributed OnLine Learning Using Preemptible Cloud Instances
Number of Coding References	6
Number of Codes Coding	2
Coverage	14.69%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	We design, analyze, and optimize DOLL, which to the best of our knowledge is the first system that provides provable performance guarantees for Distributed OnLine Learning over preemptible instances.
Modified on	16/02/2023 09:31

Name	DOLL~ Distributed OnLine Learning Using Preemptible Cloud Instances Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Driftage~ a multi-agent system framework for concept drift detection
Number of Coding References	8
Number of Codes Coding	2
Coverage	8.61%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 20:46
Coded Text	
Modified on	07/02/2022 22:41

Name	Driftage~ a multi-agent system framework for concept drift detection
Number of Coding References	8
Number of Codes Coding	2
Coverage	8.61%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 20:46
Coded Text	One of the most famous CDD algorithms is ADWIN (adaptive sliding window algorithm) [52]. It efficiently keeps a variablelength window of recent items, whose contents can be compared to discern whether there has been any change in the data distribution. This window is further divided into 2 subwindows (W0, W1) used to determine whether a change has happened. ADWIN compares the average ofW0 andW1 to confirm that they correspond to the same distribution. Concept drift is detected if the distribution equality no longer holds. Upon detecting a drift, W0 is replaced by W1 and a newW1 is initialized. ADWIN uses a
Modified on	07/02/2022 22:41

Name	Driftage~ a multi-agent system framework for concept drift detection
Number of Coding References	8
Number of Codes Coding	2
Coverage	8.61%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 20:46
Coded Text	supervised [13, 14], semi-supervised [15], unsupervised [16, 17, 18], statistical [19, 20], or even evolutionary algorithms [21] to deal with these drifts, but none of them is perfect for all drift types. Some publications are arising with machine learning ensem- bles for CDD because of the nature of the data that these detectors need to adapt to [22–25]. There are several factors such as data seasonality or change of data drift type; these ensembles can choose the best estimator for each case, and each estimator can still act alone. Nevertheless, this approach necessitates retraining of base learners and strategies to select
Modified on	07/02/2022 22:39

Name	Driftage~ a multi-agent system framework for concept drift detection
Number of Coding References	8
Number of Codes Coding	2
Coverage	8.61%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 20:46
Coded Text	There are many types of drifts in the concept drift detection (CDD) area [7, 9, 10].
Modified on	07/02/2022 22:38

Name	Driftage~ a multi-agent system framework for concept drift detection
Number of Coding References	8
Number of Codes Coding	2
Coverage	8.61%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 20:46

Coded Text	This article proposes to create Driftage: a new framework using multi-agent systems to simplify the implementation of concept drift detectors considerably and divide concept drift detection responsibilities between agents, enhancing explainability of each part of drift detection. As a case study, we illustrate our strategy using a muscle activity monitor of electromyography. We show a reduction in the number of false-positive drifts detected, improving detection interpretability, and enabling concept drift detectors' interactivity with other knowledge bases.
Modified on	07/02/2022 22:38

Name	Driftage~ a multi-agent system framework for concept drift detection
Number of Coding References	8
Number of Codes Coding	2
Coverage	8.61%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 20:46
Coded Text	Driftage is a modular framework based on MAPE-K, chosen as the pattern to model this agent-based framework because CDD needs high adaptability and fits very well with MAS. Each agent type in Driftage has only 1 accountable agent on the MAPE-K architecture. Each agent can be implemented to follow the selected goal without affecting the others but can exchange information with others. Instead of an agent using the MAPE-K software pattern, an agent on the Driftage framework can be implemented following 1 of the 4 types: Monitor, Analyser, Planner, or Executor. Each type can generate multiple autonomous agents. There are 2 main flows on this framework: (i) Monitor–Analyser: for capture and fast prediction ofconcept drifts on data; (ii) Planner–Executor: to analyse whether concept drift detected should be alerted. These 2 flows can intercommunicate by means ofaKB, where drifts are stored, and we make all history about drift analysis persistent. Each agent communicates through an XMPP server on the framework because the implementation extends Spade [49], which is a library for MAS using Python. The XMPP protocol solves some problems with MAS, already providing authentication and communication channels for the agents. XMPP servers also work for load balancing and guarantee message exchanges. We have implemented Driftage using Python because data engineers widely use it, and it enables the programmer to answer the system's requirements.
Modified on	07/02/2022 22:40

Name	Driftage~ a multi-agent system framework for concept drift detection
Number of Coding References	8
Number of Codes Coding	2
Coverage	8.61%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 20:46
Coded Text	One approach to designing adaptive software is using the MAPE-K (Monitor-Analyse-Plan-Execute over a shared Knowledge) software pattern for self-aware systems [27–30]. MAPE-K is organized into 4 components: (i) The ”Monitor” is responsible for environmental monitoring, basically capturing data from sensors or what else the software knows about the environment and stores on the knowledge base (KB); (ii) The ”Analyser” will enrich knowledge using the collected data from the environment and reporting to the KB the result of its analysis; (iii) The ”Planner” understands the analysis made by analysers and makes decisions on it while saving this information into the KB; and (iv) The ”Executor” gets decisions from the KB and knows how to execute them. The most
Modified on	07/02/2022 22:40

Name	Driftage~ a multi-agent system framework for concept drift detection
Number of Coding References	8
Number of Codes Coding	2
Coverage	8.61%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 20:46
Coded Text	The amount of data and behavior changes in society happens at a swift pace in this interconnected world. Consequently, machine learning algorithms lose accuracy because they do not know these new patterns. This change in the data pattern is known as concept drift. There exist many approaches for dealing with these drifts. Usually, these methods are costly to implement because they require (i) knowledge of drift detection algorithms, (ii) software engineering strategies, and (iii) continuous maintenance concerning new drifts.
Modified on	07/02/2022 22:37
Name	Edge Computing Solutions for Distributed Machine Learning
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Edge Computing Solutions for Distributed Machine Learning
Number of Coding References	3
Number of Codes Coding	1
Coverage	7.15%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	and caching) and send it to the cloud (or fog) for storage purposes or to perform complex learning tasks that cannot be performed at this level or in that device. Here it is possible to aggregate local models learned from many devices (model aggregation). • Fog layer. This is an intermediary layer that can benefit from the computation that is closer to the cloud and more powerful than that provided by the edge layer. Data can be stored and exploited for collective learning. In particular, after the local learning on private data at the device level, a collective training phase can take place at this level in which labels are assigned to shared and unlabeled data by means of a consensus-based algorithm. • Cloud layer. This layer acts as the backbone of the network and provides persistence data storage, and powerful and very large processing resources [44] that are not available in the other layers. If needed, it can be leveraged for aggregating global models (global learning).
Modified on	16/02/2023 11:54

Name	Edge Computing Solutions for Distributed Machine Learning
Number of Coding References	3
Number of Codes Coding	1
Coverage	7.15%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	Executing machine learning algorithms in distributed environments requires (a) to separate the tasks that compose an algorithm, (b) to coordinate their execution on the different computing nodes in accordance with any dependencies that exist between them, and (c) to manage failures that can occur on computing nodes and communication links that can be unreliable. For this reason, the choice of an appropriate distributed algorithm to solve a given problem depends both on the characteristics of the problem and on the features and configuration of the system on which the algorithm will run, the type of communication and synchronization among processes that can be performed. All these problems of traditional distributed systems are amplified when we consider IoT systems characterized by limited computing capacities, problems of energy consumption and latency, different technologies and software stacks. The approach we discuss here aims to adapt distributed versions of machine learning algorithms to IoT environments. As shown in Figure 1, in the edge-cloud continuum we can identify four different layers [43]: • Device layer. This is the layer at which IoT devices generate or collect data (data collection). This data can be generally stored in the device's storage system, both in a persistent or temporary way (data storage). Before storing or using it to perform analytics on the device, data can be filtered according to application requirements (data filtering). Then, this local data can be used to train a learning model (local learning), which at higher levels may be used in federated learning tasks. • Edge layer. This layer is where gateways such as routers, base stations or micro data centers serve to bring computing closer to IoT devices. The goal of this proximity is to gather the sensed data of IoT devices (data aggregation), preprocess and possibly cache it (data filtering
Modified on	16/02/2023 11:54

Name	Edge Computing Solutions for Distributed Machine Learning
Number of Coding References	3
Number of Codes Coding	1
Coverage	7.15%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	Not all of the levels described above are necessary (for ex- ample the fog is optional), but if reduced to two extreme levels device-cloud the architecture turns into a traditional cloudbased solution. As an example of a distributed algorithm on the edge-cloud continuum architecture we can consider a baggingbased ensemble learning algorithm for image classification. In this example, edge devices acquire data from the surrounding environment through the use of a camera. This data can be stored persistently in the device memory, or can be processed to reduce its complexity (e.g., image compression). A weak learner (i.e., a basic machine learning model such as a decision tree) can be trained on the edge device using locally acquired images only. In addition, the different local models can be hierarchically aggregated in higher levels. For example, an edge server can aggregate only the models produced by the devices associated with it. In this way we obtain a partial model that can be combined by the cloud (or fog) layer with the local models of other edge servers to generate a more accurate global model. Overall, the implementation of decentralized algorithms on this four-layers architecture must take into account the following aspects: 1) Data location - Computation is moved as close to the data as possible to minimize data movement among nodes. Since the various levels are made up of heterogeneous software components having different computing capabilities, if a task cannot be performed (or is inefficient) on a certain node of a given level, support is requested to the higher level (and so on). For example, tasks that cannot be performed on devices are offloaded to edge nodes, and so on to fog nodes up to the cloud. 2) Geographical distribution - It is necessary to consider the physical distribution of the hardware components that will execute the distributed algorithm. The hardware components may be in different places far from each other. Therefore, localization of devices may influence communication overhead and algorithm performance. 3) Algorithm features and configurations - Each algorithm can be adapted differently than the others and there may not be a unique way of migrating algorithms (code) in these environments. There may be patterns that allow a user to define and execute classes of machine learning algorithms in IoT environments, but there are many variables to consider. Moreover, deploying distributed algorithms in hybrid cloud/edge architectures is an extremely complex problem, as there are many combinations of configuration parameters [45]. 4) Task scheduling and persistence of data - A data-aware scheduler is needed to efficiently perform the tasks that compose distributed learning algorithms. The scheduler must ensure a load balance between all the computing nodes (some nodes can have different computing power), pay more attention to highly critical tasks, and, if necessary, perform replication of the tasks to improve both execution time and fault tolerance [46]. Often a centralized or distributed master node must be identified to coordinate all these scheduling activities. Moreover, consider that temporary data may be stored on nonpersistent components and therefore may be lost. 5) Application constraints - Functional and not functional constraints (quality of service or QoS) of the application that will use a machine learning algorithm must be respected. Often it is essential to meet crucial application and system requirements such as low latency, energy saving, network traffic reduction, privacy preservation, and high scalability. For those reasons machine learning algorithms must also be configurable in order to meet the requirements. In fact, the same algorithm can be performed differently on the architecture with different degrees of freedom (e.g., a less precise model on the edge nodes, a more accurate model on the cloud). 6) Coding and testing - Let consider that each layer could be implemented by different hardware and could be programmed by different software tools/libraries. This aspect makes it difficult to write and thoroughly test an algorithm. Furthermore, real tests using a large number of hardware nodes could be very expensive and inflexible, and benchmarking and setting up real experiments could be very challenging. Simulation tools result to be powerful and flexible for reproducing and testing IoT systems and networks [47]–[49].
Modified on	16/02/2023 11:54

Name	Edge Computing Solutions for Distributed Machine Learning Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Edge Computing Solutions for Distributed Machine Learning Imported Notes (2)
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Elastic Machine Learning Algorithms in Amazon SageMaker
Number of Coding References	9
Number of Codes Coding	2
Coverage	9.44%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	04/02/2022 22:14
Coded Text	(i) Support for incremental training and model freshness: It is highly uncommon to encounter large static datasets. In most cases, data keeps being generated constantly, which is often addressed with an unwelcome trade-off between training cost and accuracy. Training on a large subset of the data produces accurate models but can become extremely costly and slow, while training on new, small updates of the data (e.g., the last day) is cheaper but might not lead to very accurate results. Therefore, industrial ML platforms have to support incremental model training to regularly and costefficiently update existing models and to quickly provide accurate and 'fresh' models. (ii) Predictability oftraining costs: for large amounts of data, customers need to be able to roughly estimate in advance how much a training job would cost and how long it would run for. It is difficult to estimate the cost ahead of time for many scalable systems, which do not support incremental learning with linear update times or have irregular performance drops for high-dimensional models [5]. (iii) Elasticity and support for pausing and resuming train- ing jobs: large-scale ML scenarios often result in imbalanced workloads, where data scientists spend several days without running a single job (while they are collecting data or writing code) and then they launch several large concurrent training jobs on hundreds of machines. They also may want to pause and resume such jobs, e.g., for hyperparameter tuning or if
Modified on	28/02/2023 11:45

Name	Elastic Machine Learning Algorithms in Amazon SageMaker
Number of Coding References	9
Number of Codes Coding	2
Coverage	9.44%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	04/02/2022 22:14
Coded Text	their compute bandwidth has irregular limitations. A cloud ML platform should reduce the operational complexity of such scenarios. (iv) Furthermore, an industrial-scale ML platform must be able to handle ephemeral data, as some data streams like network traffic or video streams are never meant to be stored, which makes ingesting and learning on such data challenging. (v) Finally, anML platform must automate hyperparameter optimization and model tuning as much as possible. These tasks are already tedious even for small use cases, let alone, for large scale machine learning scenarios. Automating this part ofthe platform is crucial for driving down training costs and supporting non-ML expert users.
Modified on	28/02/2023 11:45

Name	Elastic Machine Learning Algorithms in Amazon SageMaker
Number of Coding References	9
Number of Codes Coding	2
Coverage	9.44%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	03/02/2022 10:07
Coded Text	• We discuss challenges and requirements for building an industrial-scale elastic ML platform (Section 1). • We describe the computational model (Section 2) and selected algorithms of SageMaker, which support incremental, resumable and elastic learning, as well as automatic hyperparameter optimization (Sections 3 & 4). • We compare SageMaker to several scalable, JVM-based algorithm implementations (Section 6), which we significantly outperform with regard to computation time and cost.
Modified on	28/02/2023 11:47

Name	Elastic Machine Learning Algorithms in Amazon SageMaker
Number of Coding References	9
Number of Codes Coding	2
Coverage	9.44%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	03/02/2022 10:07
Coded Text	Amazon SageMaker, a system for scalable, elastic model training on large streams of data, available as part of Amazon's cloud offerings.1 We choose a streaming model to meet requirements like linear update time, pause/resume capabilities and elasticity. Operating in such a constrained setting is mathematically complex. For example, consider the task of computing the median of a stream of numbers. The best possible approximation for finding quantiles (e.g., the median) in streams was only recently discovered [18], and the resulting algorithm is significantly more complex than offline approaches [14]
Modified on	28/02/2023 11:47

Name	Elastic Machine Learning Algorithms in Amazon SageMaker
Number of Coding References	9
Number of Codes Coding	2
Coverage	9.44%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	03/02/2022 10:07
Coded Text	Model Abstraction&Expressive State.We provide a model abstraction for the outputs of training algorithms to enforce common behavior among the different ML models. A model must implement two functions: (i) the score function evaluates a model using a metric on a dataset, and is used for debugging and hyperparameter optimization (HPO). (ii) The function evaluate receives a batch ofdata and computes the output of the model. Note that this output is model specific, and can be a single number representing a predicted class, a vector in the case of dimensionality reduction or a complete embedding matrix.
Modified on	28/02/2023 11:49

Name	Elastic Machine Learning Algorithms in Amazon SageMaker
Number of Coding References	9
Number of Codes Coding	2
Coverage	9.44%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	03/02/2022 10:07
Coded Text	Model Tuning andHyperparameter Optimisation. Typical hyperparameter optimisation workloads comprise of several training jobs that are run in parallel with different hyperparameters to determine the best configuration, which can quickly become tedious and costly. The pause-resume capabilities of our streaming model enable a couple of beneficial techniques for HPO workloads.
Modified on	28/02/2023 11:49

Name	Elastic Machine Learning Algorithms in Amazon SageMaker
Number of Coding References	9
Number of Codes Coding	2
Coverage	9.44%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	03/02/2022 10:07
Coded Text	Nevertheless, training ML models on large, continuously evolving datasets is still a difficult and costly undertaking for many companies and institutions. We discuss such challenges and derive requirements for an industrial-scale ML platform. Next, we describe the computational model behind Amazon SageMaker which is designed to meet such challenges. SageMaker is an ML platform provided as part of Amazon Web Services (AWS), and supports incremental training, resumable and elastic learning as well as automatic hyperparameter optimization.We detail how to adapt several popular ML algorithms to its computational model. Finally, we present an experimental evaluation on large datasets,
Modified on	28/02/2023 11:44

Name	Elastic Machine Learning Algorithms in Amazon SageMaker
Number of Coding References	9
Number of Codes Coding	2
Coverage	9.44%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	03/02/2022 10:07
Coded Text	Our system is therefore designed to take advantage of multiple EC2 instance types, to support modern hardware like GPUs, and to scale out model training across many machines (Figure 1). Execution onModern Hardware. In order to operate across CPUs and GPUs seamlessly, most algorithms in SageMaker use the MXNet [10] library as an interface to the underlying hardware. MXNet represents ML algorithms via a computational graph of tensor operators, optimizes this graph (e.g, to re-use allocated memory), assigns the operators to devices, and efficiently executes the computation in parallel. Distributed Learning and State Management via a Parameter Server. Distribution is achieved via a parameter server which maintains the shared state of all the machines
Modified on	28/02/2023 11:48

Name	Elastic Machine Learning Algorithms in Amazon SageMaker
Number of Coding References	9
Number of Codes Coding	2
Coverage	9.44%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	03/02/2022 10:07
Coded Text	participating in training. The parameter server is designed for fast update speeds via asynchronous communication and allows us to loosen the consistency of parameter updates. Note that the tradeoff space between update consistency and convergence speed is well explored for ML workloads [21, 22, 32, 44]. We implement the shared state abstraction as well as the server-side aggregation functions required for our computational model (outlined in Section 2) via MXNet's parameter server called KVStore [10].
Modified on	28/02/2023 11:49

Name	Elastic Machine Learning Algorithms in Amazon SageMaker Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Emerging and Changing Tasks in the Development Process for Machine Learning Systems
Number of Coding References	10
Number of Codes Coding	3
Coverage	9.28%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 20:28
Coded Text	
Modified on	07/02/2022 20:43

Name	Emerging and Changing Tasks in the Development Process for Machine Learning Systems
Number of Coding References	10
Number of Codes Coding	3
Coverage	9.28%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 20:28

Coded Text	In this paper, we aim to answer the following research question: What tasks are emerging and changing in the software development process, when developing ML-based systems? We present the results of an interview study with 16 participants from industry and academia in Europe (mostly Sweden), Asia, and North America. The interviews focused on methods and processes applied by the interviewees. Collected data was analyzed regarding challenges and emerging tasks. The results show that we are just starting to understand the implications that the use of machine learning has on software engineering. We identified several tasks that change or emerge when machine learning is used. These tasks can be associated with software engineering phases, such as requirements engineering, estimation & contracting, development iterations, and knowledge management
Modified on	07/02/2022 20:39

Name	Emerging and Changing Tasks in the Development Process for Machine Learning Systems
Number of Coding References	10
Number of Codes Coding	3
Coverage	9.28%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 20:28
Coded Text	Lessons Learned for Practitioners The following lessons learned can be summarized for practitioners. Involve ML-experts early. One lesson learned is that ML-experts need to be involved early on in the software development process. There are even potentials for synergy, when bringing together data scientists and requirements engineers, which work on building up a domain understanding. Enable bi-directional knowledge exchange. Data scientists and software engineers might need to work even closer together. An information exchange from data scientists to software engineers is often already performed. However, information needs also be exchanged the other way around. This way data scientists can benefit from domain knowledge build up by requirements engineers. Furthermore exchanging information can foster cooperation between data scientists and software engineers, e.g. when creating user documentations. Develop strategies forcustomerexpectation managementand train- ing. When setting up a new project with ML-components, software companies should from the start identify customers and future users and develop strategies on how to manage the customers' expectations throughout the project. Furthermore, projects should include a plan for user training or documentation that specifically considers the impact of the ML-component(s). Develop a portfolio ofrisk-management strategies. Software engi- neers/architects and data scientists in a team should aim to develop a portfolio of risk-management strategies together in order to account in the software system for uncertainties that come with the use ofmachine learning components. This requires expertise ofboth roles as the choice and implementation of the risk-management strategy affects the software design, while an assessment of the risk can often only be done with the help of the data scientists.
Modified on	07/02/2022 20:43
Name	Emerging and Changing Tasks in the Development Process for Machine Learning Systems
Number of Coding References	10
Number of Codes Coding	3
Coverage	9.28%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 20:28
Coded Text	We performed an interview study with 16 participants, focusing on emerging and changing task. The results uncover a set of 25 tasks associated to different software development phases, such as requirements engineering or deployment. We are just starting to understand the implications of using machinelearning components on the software development process. This study allows some first insights into how widespread the required process changes are.
Modified on	07/02/2022 20:38

Name	Emerging and Changing Tasks in the Development Process for Machine Learning Systems
Number of Coding References	10
Number of Codes Coding	3
Coverage	9.28%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 20:28
Coded Text	
Modified on	07/02/2022 20:42

Name	Emerging and Changing Tasks in the Development Process for Machine Learning Systems
Number of Coding References	10
Number of Codes Coding	3
Coverage	9.28%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 20:28
Coded Text	
Modified on	07/02/2022 20:42

Name	Emerging and Changing Tasks in the Development Process for Machine Learning Systems
Number of Coding References	10
Number of Codes Coding	3
Coverage	9.28%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 20:28
Coded Text	
Modified on	07/02/2022 20:42

Name	Emerging and Changing Tasks in the Development Process for Machine Learning Systems
Number of Coding References	10
Number of Codes Coding	3
Coverage	9.28%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 20:28
Coded Text	Integrating machine learning components in software systems is a task more and more companies are confronted with. However, there is not much knowledge today on how the software development process needs to change, when such components are integrated into a software system
Modified on	07/02/2022 20:37

Name	Emerging and Changing Tasks in the Development Process for Machine Learning Systems
Number of Coding References	10
Number of Codes Coding	3
Coverage	9.28%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 20:28
Coded Text	Recent research started to focus on the role of the data scientist [6] and their workflows [1][5]. Other researchers focus on general changes in software development practices [17][18] and arising challenges [15][7]. However, little is known about how the software development process changes when machine learning components are integrated.
Modified on	07/02/2022 20:38
Name	Emerging and Changing Tasks in the Development Process for Machine Learning Systems
Number of Coding References	10
Number of Codes Coding	3
Coverage	9.28%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56
Coded Text	Studies taking an empirical software engineering perspective on machine learning are still relatively few. Here, we focus on studies that investigate the process of building machine learning-based software systems as well as studies investigating challenges when building ML-based software systems. We used simple key-word search combined with snow-balling to identify relevant papers. Kim et al. [6] investigated the different roles a Data scientist can have in software engineering. They performed a series of interviews with data scientists to understand their roles. As a result Kim et al. [6] identified five basic types of data scientist roles, which are summarized in Table 1. Process perspectives on data science have been around for a couple of years, often in grey literature. For example, Sapp [14] describes a process based on three steps: modeling, validation, and execution. These steps involve considering computer resources, numbers of features, computation time and criteria such as scalability, reliability, and efficiency. Amershi et al. [1] summarize a nine-stage workflow process that they observed integrated in “agile-like” software engineering processes within Microsoft (see Figure 1). For that, they interviewed
Modified on	07/02/2022 20:40

Name	End-to-end Machine Learning using Kubeflow
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	End-to-end Machine Learning using Kubeflow Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Evolving Software to be ML-Driven Utilizing Real-World A~B Testing~ Experiences, Insights, Challenges
Number of Coding References	18
Number of Codes Coding	2
Coverage	7.80%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 20:28
Coded Text	Avoid Complex Black-box Models Recently, deep learning models have been picking up mo- mentum and shown to be effective in numerous applications, such as facial recognition [6], text analysis [3], and voice recognition [7]. Our experience with ML-driven Windows features, which involve tabular telemetry data as inputs (such as day of week or seconds of interactive usage) rather than images, texts, or audio, indicate that unless the scenario is known to benefit from deep learning models (like the ones mentioned), traditional ML models are preferred. There are three reasons.
Modified on	07/02/2022 20:34

Name	Evolving Software to be ML-Driven Utilizing Real-World A~B Testing~ Experiences, Insights, Challenges
Number of Coding References	18
Number of Codes Coding	2
Coverage	7.80%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 20:28
Coded Text	C. Make ML Models Updatable, Separate From Code Updates While there are some practitioners who believe in evergreen ML models, e.g. ML burned onto silicon [1], our experiences are that updatable ML models separate from other engineering codes are preferable. In our ML-driven software features, the
Modified on	07/02/2022 20:33

Name	Evolving Software to be ML-Driven Utilizing Real-World A~B Testing~ Experiences, Insights, Challenges
Number of Coding References	18
Number of Codes Coding	2
Coverage	7.80%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 20:28
Coded Text	However, user feedback indicated that the rules and heuristics could be improved: an anecdote being a user actively working on their PC outside of active hours, stepping away momentarily, and then returning to find their PC in the middle of a reboot. The engineering team leveraged WISE to enhanced WU behaviors with ML. When attempting a potentially disruptive reboot post-installation, a functionality was added to predict whether the PC will be active using ML, and then possibly adjusting behaviors (e.g. defers the reboot). This was referred internally as Smart Busy Check (SBC). In addition to reducing user disruptions, the engineering team also believed that leveraging ML could improve upgrade velocity. Since users often choose to defer (i.e. not complete updates) when notified of impending reboots, the ML model can also improve upgrade velocity by avoiding potentially disruptive situations and reducing reboot deferrals.
Modified on	07/02/2022 20:32

Name	Evolving Software to be ML-Driven Utilizing Real-World A~B Testing~ Experiences, Insights, Challenges
Number of Coding References	18
Number of Codes Coding	2
Coverage	7.80%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 20:28
Coded Text	In this section, we provide background of ML infrastructure and A/B testing capabilities on the Windows operating system. A. Windows Intelligent Services Engine The ML-driven software features discussed in this paper are built/evolved using the Windows Intelligent Services Engine (WISE): a Windows internal platform for ML-driven software features on Windows PCs [31]. WISE is an end-toend solution, with components that link together cloud-side ML training (via Azure Machine Learning) and client-side inferences (e.g. WinML). It abstracts away the complexities of building, deploying, A/B testing, and maintaining ML models from Windows engineering teams. Once deployed, the ML model can be continuously evolved/improved, in a data-driven manner, without the need to update application codes. The system (patent pending) has four integrated parts: cloud-compute ML training/retraining pipeline, scalable and secure deployment infrastructure, configurable client-side data manager and prediction orchestrator, as well as a mechanism to execute A/B testing.
Modified on	07/02/2022 20:30

Name	Evolving Software to be ML-Driven Utilizing Real-World A~B Testing~ Experiences, Insights, Challenges
Number of Coding References	18
Number of Codes Coding	2
Coverage	7.80%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 20:28
Coded Text	Make the ML Model an Addition Since evolving software to be ML-driven usually involves starting with the existing functionality that has rules and conditions, we found that adding the ML model on top of existing system is an effective incremental approach. It reduces both engineering risks as well as organizational risks. For our software features, replacing all the rules and condi- tions by ML models would have been much larger scoped and more error-prone, which entails risks not only from adding codes but also from removing codes. For example, for WU, removing the ”Active Hours” functionality would have entailed changing UI elements, shown in Figure 1, as well as behaviors. By adding the ML model as another check, the risks were lowered. This enabled engineering team to successfully build and deploy with high quality, and then start the work to retire/migrate other rules and conditions.
Modified on	07/02/2022 20:34

Name	Evolving Software to be ML-Driven Utilizing Real-World A~B Testing~ Experiences, Insights, Challenges
Number of Coding References	18
Number of Codes Coding	2
Coverage	7.80%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 20:28
Coded Text	ML-driven software is heralded as the next major advancement in software engineering; existing software today can benefit from being evolved to be ML-driven. In this paper, we contribute practical knowledge about evolving software to be ML-driven, utilizing real-world A/B testing. We draw on experiences evolving two software features from the Windows operating system to be ML-driven, with more than ten realworld A/B tests on millions of PCs over more than two years. We discuss practical reasons for using A/B testing to engineer ML-driven software, insights for success, as well as on-going realworld challenges.
Modified on	07/02/2022 20:29

Name	Evolving Software to be ML-Driven Utilizing Real-World A~B Testing~ Experiences, Insights, Challenges
Number of Coding References	18
Number of Codes Coding	2
Coverage	7.80%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 20:28
Coded Text	testing platform for the Windows operating system [17]. WExp leverages several existing components/services at Microsoft, while building new components and using existing ones in innovative ways. For example, the delivery of assets (e.g. codes and ML models) to devices leverages the Windows Update service [32]. Data are recorded and transmitted using Windows Telemetry services and adheres to user settings and restrictions [33]. Randomization and statistical comparisons of data are performed using the ExP system [16]. Finally, WExp has a set of policies and processes that help to ensure safety and best-practices. For example, all A/B tests intended to reach the general population must first be ran on internal and/or Windows Insiders devices (a self-selected group of users who run pre-release versions of Windows operating system and provide feedback [34]) to ensure safety and quality.
Modified on	07/02/2022 20:30

Name	Evolving Software to be ML-Driven Utilizing Real-World A~B Testing~ Experiences, Insights, Challenges
Number of Coding References	18
Number of Codes Coding	2
Coverage	7.80%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 20:28
Coded Text	The Windows Experimentation Platform WISE evaluates the causal impact of ML models with the Windows Experimentation Platform (WExp): a client-side A/B
Modified on	07/02/2022 20:30

Name	Evolving Software to be ML-Driven Utilizing Real-World A~B Testing~ Experiences, Insights, Challenges
Number of Coding References	18
Number of Codes Coding	2
Coverage	7.80%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 20:28
Coded Text	Use Overall Success Metrics in Addition to Scenario Success Metrics In addition to scenario success metrics, overall success metrics are needed in A/B testing because ML can cause
Modified on	07/02/2022 20:34

Name	Evolving Software to be ML-Driven Utilizing Real-World A~B Testing~ Experiences, Insights, Challenges
Number of Coding References	18
Number of Codes Coding	2
Coverage	7.80%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 20:28
Coded Text	C. Make ML Models Updatable, Separate From Code Updates While there are some practitioners who believe in evergreen ML models, e.g. ML burned onto silicon [1], our experiences are that updatable ML models separate from other engineering codes are preferable. In our ML-driven software features, the
Modified on	07/02/2022 20:33

Name	Evolving Software to be ML-Driven Utilizing Real-World A~B Testing~ Experiences, Insights, Challenges
Number of Coding References	18
Number of Codes Coding	2
Coverage	7.80%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 20:28
Coded Text	Different Definitions of Equality for Different Applications Machine learning can make software less intrusive, more informative, and more reliable; however, using ML to change behaviors of an application can have unintended consequences of unfairly impacting its users [45]–[47]. Building fair machine learning systems is a difficult socio-technical effort that requires us to carefully define ”fairness” in each scenario. Defining fairness is difficult in general but can be especially difficult when trying to ensure a complicated ML system is ”fair” to all users.
Modified on	07/02/2022 20:34

Name	Evolving Software to be ML-Driven Utilizing Real-World A~B Testing~ Experiences, Insights, Challenges
Number of Coding References	18
Number of Codes Coding	2
Coverage	7.80%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 20:28
Coded Text	Evolving software to be ML-driven (perhaps more so than software built from the ground up to be ML-driven) entails working well with other software components as well as with other parts of the ML platform. As we will discuss in more detail in the next section, an effective approach to evolve existing software to be ML-driven is to add the ML model to the existing rules. This implies that the ML functionality has integrated appropriately with other parts of the software. For example, in ML-driven WU, when the ML model defers a reboot, integration with a retry logic is needed to ensure that critical updates are installed in a timely manner by retrying reboots as soon as possible.
Modified on	07/02/2022 20:32

Name	Evolving Software to be ML-Driven Utilizing Real-World A~B Testing~ Experiences, Insights, Challenges
Number of Coding References	18
Number of Codes Coding	2
Coverage	7.80%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 20:28
Coded Text	Federated Learning with Local Privacy Federated learning (FL) with local privacy (LP) is a promis- ing approach of distributing the costs of training ML models and strengthening the privacy protection of user data [3]. Using methods like Secure Multiparty Communications and Local Differential Privacy, researchers have shown that edge devices can collaboratively train an ML model without users' data leaving their devices. With increasing computing capabilities of edge devices and rising concerns over user data privacy, FL with LP has been receiving growing interests and has been investigated in many applications [23].
Modified on	07/02/2022 20:34

Name	Evolving Software to be ML-Driven Utilizing Real-World A~B Testing~ Experiences, Insights, Challenges
Number of Coding References	18
Number of Codes Coding	2
Coverage	7.80%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 20:28
Coded Text	For real-world software (especially the Windows operat- ing system), it is known that many unforeseen usage situations/contexts are possible and that a complete in-house testing is impractical [38]. Therefore, the use of real-world A/B tests helps to ensure quality. This is particularly true for ML-driven software, since the motivations/justifications for using ML are complex and not well covered by existing rules. Real-word A/B testing is potentially even more important since it can be hard to foresee the behaviors of ML-driven software in complex situations.
Modified on	07/02/2022 20:32

Name	Evolving Software to be ML-Driven Utilizing Real-World A~B Testing~ Experiences, Insights, Challenges
Number of Coding References	18
Number of Codes Coding	2
Coverage	7.80%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 20:28
Coded Text	In this paper, we contribute practical knowledge about evolving software to be ML-driven, utilizing real-world A/B testing. We draw on experiences evolving two software features from the Windows operating system to be ML-driven, with more than ten realworld A/B tests on millions of PCs over more than two years. We discuss practical reasons for using A/B testing to engineer ML-driven software, insights for success, as well as on-going realworld challenges.
Modified on	07/02/2022 20:28

Name	Evolving Software to be ML-Driven Utilizing Real-World A~B Testing~ Experiences, Insights, Challenges
Number of Coding References	18
Number of Codes Coding	2
Coverage	7.80%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 20:28
Coded Text	Make the ML Model an Addition Since evolving software to be ML-driven usually involves starting with the existing functionality that has rules and conditions, we found that adding the ML model on top of existing system is an effective incremental approach. It reduces both engineering risks as well as organizational risks. For our software features, replacing all the rules and condi- tions by ML models would have been much larger scoped and more error-prone, which entails risks not only from adding codes but also from removing codes. For example, for WU, removing the ”Active Hours” functionality would have entailed changing UI elements, shown in Figure 1, as well as behaviors. By adding the ML model as another check, the risks were lowered. This enabled engineering team to successfully build and deploy with high quality, and then start the work to retire/migrate other rules and conditions.
Modified on	07/02/2022 20:33

Name	Evolving Software to be ML-Driven Utilizing Real-World A~B Testing~ Experiences, Insights, Challenges
Number of Coding References	18
Number of Codes Coding	2
Coverage	7.80%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 20:28
Coded Text	ML models, by themselves, do not produce user-facing be- haviors in our case studies (e.g. initiating reboots or displaying surveys); to do so, other codes have to call the ML model and then take its outputs to perform actions. Those codes can have bugs. Consequently, as with other types of software, A/B testing helps to ensure quality. For both cases of our MLdriven software features, A/B testing helped to uncover quality issues, which were then addressed.
Modified on	07/02/2022 20:32

Name	Evolving Software to be ML-Driven Utilizing Real-World A~B Testing~ Experiences, Insights, Challenges
Number of Coding References	18
Number of Codes Coding	2
Coverage	7.80%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 20:28
Coded Text	sample ratio mismatches (SRMs). An SRM in A/B testing is where the sample size is statistically significantly different between different groups. For example, in a 50/50 A/B test of 100 devices, 0 and 50 devices reporting a metric would constitute an SRM for the metric.
Modified on	07/02/2022 20:33

Name	Failure Prediction using Transfer Learning in Large-scale Continuous Integration Environments
Number of Coding References	2
Number of Codes Coding	1
Coverage	3.93%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	10/02/2022 11:26
Coded Text	
Modified on	16/02/2023 08:34

Name	Failure Prediction using Transfer Learning in Large-scale Continuous Integration Environments
Number of Coding References	2
Number of Codes Coding	1
Coverage	3.93%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	10/02/2022 11:26

Coded Text	Based on domain similarity, transfer learning algorithms can be categorized into two main classes, e.g., homogeneous transfer learning and heterogeneous transfer learning. The homogeneous transfer learning algorithms consider that the source and target domain have the same feature space, but the data distribution between these two domains is different. Homogeneous transfer learning algorithms try to minimize that difference. On the other hand, heterogeneous transfer learning algorithms assume that the source and target domain have different feature spaces. For our case, the source and target domain have the same feature space; thus, we consider the homogeneous transfer learning algorithms for test suite failure prediction.
Modified on	16/02/2023 08:31

Name	Failure Prediction using Transfer Learning in Large-scale Continuous Integration Environments Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Federated Clouds for Efficient Multitasking in Distributed Artificial Intelligence Applications
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Federated Clouds for Efficient Multitasking in Distributed Artificial Intelligence Applications Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Finding data compatibility bugs with JSON subschema checking
Number of Coding References	7
Number of Codes Coding	2
Coverage	2.52%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	
Modified on	07/02/2022 19:46

Name	Finding data compatibility bugs with JSON subschema checking
Number of Coding References	7
Number of Codes Coding	2
Coverage	2.52%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	Incorrect ML Pipelines in Lale. As a second real-world usage scenario, we apply jsonsubschema to type-check machine learning pipelines implemented in Lale [21]. Lale uses JSON schemas to describe both ML operators and ML datasets. An ML pipeline is a graph where nodes are operators and edges are dataflow. Lale uses jsonsubschema to check whether the root operators of an ML pipeline are compatible with an input dataset, as well as whether on all edges the intermediate dataset from the predecessor is compatible with the successor.
Modified on	07/02/2022 19:46

Name	Finding data compatibility bugs with JSON subschema checking
Number of Coding References	7
Number of Codes Coding	2
Coverage	2.52%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	06/01/2022 16:31
Coded Text	
Modified on	20/06/2022 13:44

Name	Finding data compatibility bugs with JSON subschema checking
Number of Coding References	7
Number of Codes Coding	2
Coverage	2.52%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	06/01/2022 16:31
Coded Text	Figure 6 shows an example. Lines 1ś2 configure an ML pipeline with two operators, RFE (recursive feature elimination) and NMF (nonnegative matrix factorization). The pipe combinator (») introduces a dataflow edge between these two operators. Line 5 tries to train the pipeline on a given dataset. Internally, Lale uses jsonsubschema to check whether the output schema of RFE is a subschema of the input schema of NMF. In this example, that check quickly yields an error message, since RFE returns an array of arrays of numbers, but NMF expects an array of arrays of non-negative numbers. In this example, the subschema check saved time compared to the approach of first training RFE, which can take minutes, and only then triggering an exception in NMF due to negative numbers. We ran Lale's regression test suite with instrumentation to log all subschema checks and counted 2,818 unique schema pairs being checked. In two years of production use of jsonsubschema, the checks have revealed many bugs, 38 of which are summarized in the lower part of Table 3. All bugs have been fixed in response to finding them with jsonsubschema. For example, in Lale-1 (Figure 7), a subschema check reveals that the output ofa classifier could either be numeric or string labels, but no mixing of the two kinds. The approach detects 43 bugs, most of which are already fixed.
Modified on	20/06/2022 13:42

Name	Finding data compatibility bugs with JSON subschema checking
Number of Coding References	7
Number of Codes Coding	2
Coverage	2.52%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	06/01/2022 16:31
Coded Text	In summary, this paper makes the following contributions: • Formulating the problem of detecting data compatibility bugs as JSON subschema checking (Section 2). • A canonicalizer and simplifier that converts a given schema into a schema that is simpler to check yet permits the same set of documents (Sections 3.1 and 3.2). • A subschema checker for canonicalized JSON schemas that uses separate subschema checking rules for each basic JSON type (Section 3.3). • Empirical evidence that the approach outperforms the state of the art, and that it reveals real-world data compatibility bugs in different domains (Section 5).
Modified on	07/02/2022 19:45

Name	Finding data compatibility bugs with JSON subschema checking
Number of Coding References	7
Number of Codes Coding	2
Coverage	2.52%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	06/01/2022 16:31
Coded Text	Incorrect ML Pipelines in Lale. As a second real-world usage scenario, we apply jsonsubschema to type-check machine learning pipelines implemented in Lale [21]. Lale uses JSON schemas to describe both ML operators and ML datasets. An ML pipeline is a graph where nodes are operators and edges are dataflow. Lale uses jsonsubschema to check whether the root operators of an ML pipeline are compatible with an input dataset, as well as whether on all edges the intermediate dataset from the predecessor is compatible with the successor.
Modified on	20/06/2022 13:41

Name	Finding data compatibility bugs with JSON subschema checking
Number of Coding References	7
Number of Codes Coding	2
Coverage	2.52%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	06/01/2022 16:31
Coded Text	paper presents a novel way of detecting a class of data compatibility bugs via JSON subschema checking. Subschema checks find bugs before concrete JSON data is available and across all possible data specified by a schema. For example, one can check if evolving a schema would break API clients or if two components of a machine learning pipeline have incompatible expectations about data. Deciding whether one JSON schema is a subschema of another is non-trivial because the JSON Schema specification language is rich. Our key insight to address this challenge is to first reduce the richness of schemas by canonicalizing and simplifying them, and to then reason about the subschema question on simpler schema fragments using type -specific checkers. We apply our subschema checker to thousands of real-world schemas from different domains. In all experiments, the approach is correct whenever it gives an answer (100% precision and correctness), which is the case for most schema pairs (93.5% recall), clearly outperforming the state-of-the-art tool.
Modified on	20/06/2022 13:32

Name	Flow-Based Programming for Machine Learning
Number of Coding References	8
Number of Codes Coding	1
Coverage	9.69%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	Design ofModular Components An ML flow consists of a set of connected components. Hence, we need to design the foundational constituent or components first to realise flow-based ML programming. A component mostly does one data processing step in ML model creation and internally abstracts a specific API for delivering that functionality. However, modelling every single Spark ML API as a different component would defeat the very purpose of abstraction. With such a design, the components would have a one-to-one correspondence with the underlying APIs and programming syntax of the ML framework, making it harder for less-skilled programmers to comprehend. Hence, we introduce our first design choice by grouping several APIs as one component such that the component represents a data processing operation at a high level and is understandable to end-users. Moreover, the abstracted APIs are invoked in an ordered fashion within the component to deliver the functionality. The different parameters accepted by the APIs used inside a component are made available on the front-end as component property which the user can configure to fine-tune the operation of the component. The essential parameters of the APIs are initialised with acceptable default settings unless overridden by the user of the component. The second design choice is making the components as loosely coupled to each other possible and the achieving tight functional cohesion between the APIs used within a single Future Internet 2022, 14, 58 6 of 23 component. This leaves space for a future extension where we can introduce new Spark ML APIs as components without interfering with the existing pool of components. To achieve this, we clearly define the input and output interface of every component and define positional hierarchy rules for each one of them. These rules help to decide whether a component can be used in a specific position in a flow or not. The flow-validation step (discussed in Section 5.2) checks this to ensure that the order in which the components are connected or rather the sequence in which the APIs are invoked would not lead to compile-time errors. The third design choice introduces additional abstraction by hiding away parts of the ML application which are necessary for it to compile and run, written by developers when coding from scratch, nevertheless, which do not directly correspond to the data processing logic of the application. An example would be the code responsible for initialising the Spark session or closing it within which the remaining data processing APIs are invoked. Another example can be configuring the Spark session like setting the application name, configuring the running environment mode (local or cluster), specifying the driver memory size and providing the binding address among many others. We handle these aspects at the back-end to enable the end-user of such a tool to focus solely on the business logic or data processing logic of the application. The code-generator, running at the backend (discussed in Section 5.3), is responsible for adding such crucial parts and initialising required settings with sensible defaults to the final code to make it compilable. Nevertheless, the default settings can be overridden by the user from the front-end. For example, the “Start” component (discussed in Section 5.2) is a special component used by the user to mark the start of the flow which can be configured to fine-tune and explicitly override different property values of the Spark session as discussed above. 5.2. Flow Specification and Flow-Checking As a second step, we take the components (discussed in Section 5.1) and make them available to the end-user via a programming tool. The programming tool must have an interactive graphical user interface consisting of a palette, a drawing area called canvas, a property pane and a message pane. The palette contains all available modular components which can be composed in a flow confirming to some standard flow composition rules. The actual flow composition takes place on the canvas when the user drags a component from the palette and places it on the canvas. The component, when present on the canvas and when selected, should display all its configurable properties in the property pane. The user can override default settings and provide custom settings for the operation of a component via this pane. The flow, while being composed on the canvas, is captured, converted into a directed acyclic graph (DAG) and checked for the correct order of the connected components. The flow-checking should be done whenever there is a state change. A state change occurs whenever something changes on the canvas. For example, a new component is dragged onto the canvas from the palette or the connection between two already present components change, among other such change possibilities. Flow-checking checks the user flow for any potential irreconcilability with the compositional rules. Such a flow when passed to the back-end

	for code generation will generate target code which does not produce any compile-time errors. The components are composable in the form of a flow if they adhere to the following compositional rules: 1. A flow is a DAG consisting of a series of connected components. 2. It starts with a particular component called the “Start” component and ends with a specific component called the “Save Model” component. 3. Every component has its input and output interface adequately defined, and a component can be allowed to be used in a specific position in a flow if it is compatible with the output of its immediate predecessor and if it is permitted at that stage of processing. For example, the application of the ML algorithm is only possible after feature extraction. Hence, the component corresponding to the ML algorithm and evaluation must be connected after the feature extraction component.
Modified on	18/02/2023 15:56
Name	Flow-Based Programming for Machine Learning
Number of Coding References	8
Number of Codes Coding	1
Coverage	9.69%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	Flow-Based Programming (FBP) is a programming paradigm invented by J. Paul Rodker Morrison in the late 1960s [18]. It is an approach to develop applications where program steps communicate with each other by transmitting streams of data. The data flow is unidirectional. At one point in time, only one process can work on data. Each process/component is independent of others, and hence many flows of operation can be generated with different combinations of the components. The components are responsible only for the input data that they consume. Therefore, the input/output data formats are part of the specifications of the components in FBP. The components act as software black boxes and are as loosely coupled as possible in the FBP application which provides the flexibility to add new features without affecting the existing network of FBP components. 2.4. Model-Driven Software Development MDSD abstracts away the domain-specific implementation from the design of the software systems [7]. The levels of abstraction in a model-driven approach help to communicate the design, scope, and intent of the software system to a broader audience, which increases the quality of the system overall. The models in MDSD are the abstract representation of real-world things that need to be understood before building a system. These
Modified on	18/02/2023 15:30

Name	Flow-Based Programming for Machine Learning
Number of Coding References	8
Number of Codes Coding	1
Coverage	9.69%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	Machine Learning (ML) has gained prominence and has tremendous applications in fields like medicine, biology, geography and astrophysics, to name a few. Arguably, in such areas, it is used by domain experts, who are not necessarily skilled-programmers. Thus, it presents a steep learning curve for such domain experts in programming ML applications. To overcome this and foster widespread adoption of ML techniques, we propose to equip them with domain-specific graphical tools. Such tools, based on the principles of flow-based programming paradigm, would support the graphical composition of ML applications at a higher level of abstraction and auto-generation of target code. Accordingly, (i) we have modelled ML algorithms as composable components; (ii) described an approach to parse a flow created by connecting several such composable components and use an APIbased code generation technique to generate the ML application. To demonstrate the feasibility of our conceptual approach, we have modelled the APIs of Apache Spark ML as composable components and validated it in three use-cases. The use-cases are designed to capture the ease of program specification at a higher abstraction level, easy parametrisation of ML APIs, auto-generation of the ML application and auto-validation of the generated model for better prediction accuracy.
Modified on	18/02/2023 15:28
Name	Flow-Based Programming for Machine Learning
Number of Coding References	8
Number of Codes Coding	1
Coverage	9.69%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	models are transformed into platform-specific implementation through domain modelling languages. The MDSD can be compared to the transformation of a high-level programming language to machine code. MDSD often involves transforming the model into text, which is popularly known as code generation. There are different kinds of code-generation techniques like templates and filtering, templates and meta-model, code weaving and API-based code generation among others [7]. API-based code generators are the most simple and the most popular. These simply provide an API with which the elements of the target platform or language can be generated. They are dependent on the abstract syntax of the target language and are always tied to that language. To generate target code in a new language, we need new APIs working on the abstract syntax of the new target language. 3. Related Work Literature hardly indicated any significant research work done to support graphical ML programming at a higher level of abstraction and simultaneously explaining programming concepts necessary for such. Nevertheless, there are a number of relevant works in the literature as well as products in the market which support high-level ML programming like WEKA [19], Azure Machine Learning Studio [20], KNIME [21], Orange [22], BigML [23], mljar [24], RapidMiner [25], Streamanalytix [26], Lemonade [27] and Streamsets [28] among others. Out of these, only Streamanalytix, Lemonade and Streamsets specifically deal with Spark ML.
Modified on	18/02/2023 15:30

Name	Flow-Based Programming for Machine Learning
Number of Coding References	8
Number of Codes Coding	1
Coverage	9.69%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	Previously, we had attempted to support programming of Spark applications via graphical flow-based programming paradigm [39–42]. The previous work culminated in the doctoral dissertation of the last author. This work is an extension of the previous work. The main difference lies in the code-generation technique. Previously, we had used the API-based code generation technique to generate only the basic skeleton of the Spark application. A library called 'SparFlo' [42] was developed, which contained a generic method implementation of various Spark APIs. Codeweaving was used to invoke these generic method implementations inside the basic skeleton of the target Spark program. This ensured that the SparFlo library, when supported by any graphical programming tool, would easily support Spark programming. Nevertheless, any changes or updates in Spark libraries would cause the release of a new version of the SparFlo library containing the latest generic method implementations of the Spark APIs. Hence, in this attempt, we have relied only on the API-based code generation technique, which eliminates our conceptual approach to develop a pre-packaged implementation of all Spark APIs. It also decouples from a specific Spark version as nowwe can independently parse a Spark version to generate relevant target source code.
Modified on	18/02/2023 15:57

Name	Flow-Based Programming for Machine Learning
Number of Coding References	8
Number of Codes Coding	1
Coverage	9.69%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	There are a plethora of ML libraries available, including TensorFlow [9,10], Py- Torch [11], FlinkML [12], SparkML and scikit-learn [13], among others. TensorFlow has become one of the most prominent libraries for both ML as well as DL. It provides flexible APIs for different programming languages with support for easy creation of models by abstracting low-level details. PyTorch is another open-source ML library developed by Facebook. It is based on Torch [14], an open-source ML library used for scientific computation. This library provides several algorithms for DL applications like natural language processing and computer vision, among others via Python APIs. Similarly, Scikit-learn is an open-source ML framework based on SciPy [15], which includes lots of packages for scientific computing in Python. The framework has excellent support for traditional ML applications like classification, clustering and dimensionality reduction. Open-source computer vision (OpenCV) is a library providing ML algorithms and mainly used in the field of computer vision [16]. OpenCV is implemented in C++. However, it provides APIs in other languages like Java, Python, Haskell and many more. Apache Flink, the popular distributed stream processing platform, provides ML APIs in the form of a library called FlinkML. The application designed using these APIs will run inside the Flink execution environment. Another prominent open-source library is Weka. These are some of the most widely used libraries and listing all the available all the ML libraries is beyond the scope of this paper. We, therefore, invite interested readers to refer to [17] for more comprehensive information about ML and DL libraries.
Modified on	18/02/2023 15:30

Name	Flow-Based Programming for Machine Learning
Number of Coding References	8
Number of Codes Coding	1
Coverage	9.69%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	Tools Deep Learning Studio Microsoft Azure ML Streamanalytix StreamSets Rapidminer Lemonade Our Solution Graphical Interface Target Framework Flow-based GUI Keras Flow-based GUI Graphical wizard-based Flow-based GUI Graphical wizard-based Flow-based GUI Spark ML (Python, R) Spark ML, H2O, PMML Spark ML (Python & Scala) Unknown Spark ML (PySpark) Code-Snippet Not Required as Input ~ ~ (~ for auto ML) ~ ~ ~ ~ Flow-based GUI Spark ML (Java) ~ Code Generation for ML Program ~ ~ ~ ~ ~ ~ ~ Include Data Pre-Processing ~ ~ ~ ~ ~ ~ ~ Ensemble Method Supported ~ ~ (auto ML), ~ (for Saprk application) ~ ~ ~ ~ ~
Modified on	18/02/2023 15:58

Name	Flow-Based Programming for Machine Learning
Number of Coding References	8
Number of Codes Coding	1
Coverage	9.69%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	We take the Java APIs of Spark ML operating on DataFrame [3], a popular ML library of Apache Spark [4–6], model them as composable components. Every component abstracts one or more underlying APIs such that they represent one unit of processing step in an ML application. The different parameters accepted by the underlying APIs are made available on the front-end while using a specific component to support easy parametrisation. 2. Development of a conceptual approach to parse an ML flow created by connecting several such components from step 1. The parsing ensures that the components are connected in an acceptable positional hierarchy such that it would generate target code which is compilable. The parsed user-flow is used to generate target ML code using principles of Model-Driven Software Development (MDSD). Model to text transformation is used, especially API based code generation techniques [7], to transform the graphical model to target code. 3. The conceptual approach is validated by designing three ML use-cases involving prediction using decision trees, anomaly detection with k-Means clustering, and collaborative filtering techniques to develop a music recommender application. The use-cases demonstrate how such flows can be created by connecting different components from step 1 at a higher level of abstraction, parameters to various components can be configured with ease, automatic parsing of the user flow to give feedback to the user if a component has been used in a wrong position in a flow and finally automatic generation of ML application without the end-user having to write any code. The user can split the initial dataset into training and testing datasets, specify a range for different model parameters for the system to iteratively generate models and test them till a model is produced with higher prediction accuracy.
Modified on	18/02/2023 15:29

Name	Flow-Based Programming for Machine Learning Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Going deeper than deep learning for massive data analytics under physical constraints
Number of Coding References	11
Number of Codes Coding	2
Coverage	18.11%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	Deep Neural Networks (DNNs) are a set of powerful yet computationally complex learning mechanisms that are projected to dominate various artificial intelligence and massive data analytic domains. Physical viability, such as timing, memory, or energy efficiency, are standing challenges in realizing the true potential of DNNs.
Modified on	11/02/2022 14:43

Name	Going deeper than deep learning for massive data analytics under physical constraints
Number of Coding References	11
Number of Codes Coding	2
Coverage	18.11%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	
Modified on	11/02/2022 14:44

Name	Going deeper than deep learning for massive data analytics under physical constraints
Number of Coding References	11
Number of Codes Coding	2
Coverage	18.11%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	3.2 Execution Phase Once the DNN model is trained to meet the desired infer- ence accuracy, there are two main steps to predict the class label for each incoming test data (Figure 3). First, each test sample is projected based on the learned dictionary matrix D. Second, the corresponding coefficient vector C is fed into the trained DNN model to obtain its class label. The execution phase only requires a forward propagation for each transformed data sample. The same computational model is directly applicable to the execution phase by excluding the backward propagation.
Modified on	11/02/2022 14:44

Name	Going deeper than deep learning for massive data analytics under physical constraints
Number of Coding References	11
Number of Codes Coding	2
Coverage	18.11%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	4. EVALUATIONS Platform Setup. We use Nvidia Tegra K1 development kit as our hardware platform [7]. The Nvidia TK1 is an embedded processor designed for realizing different computer vision, robotics, security, automotive, and mobile sensing applications. It includes 192 CUDA cores and a 4-Plus-1 quad-coreARMCortex A15CPUwitha2GBmemory. We leverage all the available CPU cores on the specified platform to perform data projection using standard Message Passing Interface (MPI), while the DNN training and execution have been conducted using the CUDA cores. We adopt stochastic gradient descent with momentum [8] for back propagation and Tangent-Hyperbolic as the non-linear activation function for hidden layers. Application Data. We evaluate DeLight using (i) Imaging, (ii) Smart-sensing, and (iii) Audio datasets that are key enablers in deployment of different autonomous learning tasks. Table 2 specifies our dataset size for each application.
Modified on	11/02/2022 14:44

Name	Going deeper than deep learning for massive data analytics under physical constraints
Number of Coding References	11
Number of Codes Coding	2
Coverage	18.11%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Bag of DNN Models. To further boost the inference accuracy, DeLight uses the achieved performance gain to
Modified on	11/02/2022 14:44

Name	Going deeper than deep learning for massive data analytics under physical constraints
Number of Coding References	11
Number of Codes Coding	2
Coverage	18.11%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	models by maximizing the number of parameters that can be trained within the limits of a given computational or timing budget. (ii) It enables on-chip processing of sensor data acquired on portable embedded platforms such as autonomous vehicles, smart phones, and wearables while evading the requirement to offload personal content to the clouds. (iii) It empowers deployment of several DNN configurations within the confine of the underlying resource provisioning to empirically identify the best model.
Modified on	11/02/2022 14:44

Name	Going deeper than deep learning for massive data analytics under physical constraints
Number of Coding References	11
Number of Codes Coding	2
Coverage	18.11%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Physical Profiling. To update the DNN parameters, each batch of training data is processed in two steps: (i) Forward Propagation (FP), and (ii) Backward Propagation (BP). Table 1 details the computation and communication cost of each step. We use TFP i and TBP i to denote the forward and backward propagation runtimes associated with model i. DeLight characterizes the physical coefficients listed in Table 1 by executing a set of micro-benchmarks that emulate basic operations involved in forward and backward pass. The characterization is a one-time process that incurs a fixed negligible overhead. A similar approach can be used to model energy consumption.
Modified on	11/02/2022 14:44

Name	Going deeper than deep learning for massive data analytics under physical constraints
Number of Coding References	11
Number of Codes Coding	2
Coverage	18.11%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Platform Aware Data Projection. DeLight's data projection is a pre-processing step. It works by projecting the input data Am×n to an ensemble of lower-dimensional embeddings by seeking the best suited dictionary matrix Dm×l and a corresponding coefficient matrix Cl×n s.t., the required number of neurons per layer of a DNN topology for delivering a target inference accuracy, is minimized. In brief, DeLight optimization is as follows:
Modified on	11/02/2022 14:44

Name	Going deeper than deep learning for massive data analytics under physical constraints
Number of Coding References	11
Number of Codes Coding	2
Coverage	18.11%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	There are several advantages in devising resource efficient deep learning methodologies: (i) It benefits scaling of DNN
Modified on	11/02/2022 14:43

Name	Going deeper than deep learning for massive data analytics under physical constraints
Number of Coding References	11
Number of Codes Coding	2
Coverage	18.11%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03

Coded Text	We propose DeLight, a set of novel methodologies which aim to bring physical constraints as design parameters in the training and execution of DNN architectures. We use physical profiling to bound the network size in accordance to the pertinent platform's characteristics. An automated customization methodology is proposed to adaptively conform the DNN configurations to meet the characterization of the underlying hardware while minimally affecting the inference accuracy. The key to our approach is a new content- and resource-aware transformation of data to a lower-dimensional embedding by which learning the correlation between data samples requires significantly smaller number of neurons. We leverage the performance gain achieved as a result of the data transformation to enable the training of multiple DNN architectures that can be aggregated to further boost the inference accuracy. An accompanying API is also developed, which can be used for rapid prototyping of an arbitrary DNN application customized to the platform. Proof-of concept evaluations for deployment of different imaging, audio, and smart-sensing applications demonstrate up to 100-fold performance improvement compared to the state-of-the-art DNN solutions.
Modified on	11/02/2022 14:43

Name	Going deeper than deep learning for massive data analytics under physical constraints
Number of Coding References	11
Number of Codes Coding	2
Coverage	18.11%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	We propose DeLight, an end-to-end learning framework that enables simultaneous training and execution of DNN models customized to the pertinent resource provisioning [4]. The resource efficiency of DNN training and execution is explicitly governed by the number of neurons per layer of a DNN architecture. Traditionally, the input layer size of a DNN model is dictated by the feature space size of the incoming data measurements. DeLight proposes the use of a new context- and resource-aware data projection as the primary step to reduce the dimensionality of the input layer of DNNs customized to platform resources and constraints. Our approach subsequently shrinks the dimensionality of the overall DNN model, resulting in significant cost reduction while delivering the same inference accuracy. The proposed data projection is a pre-processing
Modified on	11/02/2022 14:44

Name	Governance of artificial intelligence
Number of Coding References	5
Number of Codes Coding	2
Coverage	30.77%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 11:05
Coded Text	Steps forward for AI governance The conceptual framing of AI is a crucial determinant of how the problems introduced by AI are understood, whether they are included in policies and the degree of priority afforded to the problem in public policy formulation (Perry & Uuk, 2019), but the issue of framing has yet to be extensively discussed as a key component of AI governance. As AI is still developing with the potential to grow more salient and diverse, the complexity of its challenges suggests that decision-making in AI systems needs to be carefully conceptualised according to their context of application, and these framing processes should be subject to public debate (Cunneen, Mullins, & Murphy, 2019). For instance, how policymakers frame the relative importance of different ethical principles arising from a particular AI application, conflicts between ethical principles, and the justification of their importance have critical implications on the resulting trade-offs from designing AI systems and the public's compliance with different ethical guidelines (Piano, 2020). The initial framing of AI is also critical to avoid amplifying perceived risks and fears surrounding new technologies and instead promote a more balanced discourse on what AI is, the goals and norms that AI should reinforce in society, and the design requirements to maximise its benefits while minimising its risks (Cath et al. 2018; Cunneen et al., 2019) To address the governance challenges posed by the uncertainty and complexity of AI developments, there are increasing calls for the adoption of innovative governance approaches such as adaptive governance and hybrid or 'de-centred' governance (Dafoe, 2018; Linkov et al., 2018b; Pagallo, Casanovas, & Madelin, 2019; Tan & Taeihagh, 2021b). Characteristic of adaptive and hybrid governance is the diminished role of the government in controlling the distribution of resources in society. This is defined in hybrid governance as a combination of state and non-state actors, or blurring the public/private distinction by different degrees where regulation exists as a combination of industry standards and 'public regulatory oversight' (Guihot et al., 2017; Hemphill, 2016). Hybrid governance can exist in the forms of co-regulation, enforced self-regulation, and metaregulation (Hemphill, 2016), all of which emphasise the increasing role played by nonstate actors and the need for 'ongoing assessment of the balance of power' between private and public actors (Leiser & Murray, 2016). Similarly, adaptive governance emphasises the need to shift away from 'command and control' measures (Gasser & Almeida, 2017) towards more flexible approaches characterised by the iterative adjustment and improvement of regulations and policies as new information is gathered (Li, Taeihagh, De Jong, & Klinke, 2021; Linkov et al., 2018; Tan & Taeihagh, 2021b). Adaptive approaches are purported to be advantageous for proactively identifying and addressing the risks introduced from ML systems that are expected to change over time, as well as raising the public's awareness of AI and engaging with the public to identify new issues that have not yet entered the government's agenda (Cihon, Maas, & Kemp, 2020; Linkov et al., 2018; Pagallo et al., 2019). Flexibility is critical to enable diverse groups of stakeholders to build consensus around the norms and trade-offs in designing AI systems, as well as for global AI governance to be applicable across different geographical, cultural, and legal contexts and aligned with existing standards of democracy and human rights (Gasser & Almeida, 2017; Wirtz, Weyerer, & Geyer, 2019). Examples of adaptive governance include laws that require regular risk assessments of the regulated activity, POLICY AND SOCIETY 147 soft law approaches that involve collaboration with the affected stakeholders to develop guidelines, and legal experimentation and regulatory sandboxes to test innovative frameworks for liability and accountability for AI that will be adapted in iterative phases (Cath et al. 2018; Hemphill, 2020; Linkov et al., 2018; Philipsen, Stamhuis, & De Jong, 2021). New governance frameworks can also be adapted from the approaches taken to regulate previous emerging technologies (Gasser & Almeida, 2017). Studies have analysed hybrid governance to regulate the Internet and emerging digital technologies. For instance, Leiser and Murray (2016) highlight the need to account for the increasing role of non-state actors, particularly private actors such as technology companies that control the exchange of information across the globe, transnational private actors that are developing design principles, as well as the role of civil society groups in ensuring accountability of the former two. Lessons could be drawn from the experiences of governing previous emerging technologies such as the Internet, nanotechnology, aviation safety and space law (Butcher & Beridze, 2019; Snir, 2014) to inform the governance

of AI and other new emerging technologies. In addition, a key research agenda for future studies on AI governance would be to analyse the distinctive features of AI technology that warrants different approaches from previous technologies. An emerging body of literature has proposed governing AI systems through their
design, where social, legal, and ethical rules can be enforced through code to regulate the behaviour of AI systems (Leenes & Lucivero, 2014). For instance, provisions in data protection laws can be translated into technical specifications for robots equipped with cameras to automatically blur faces to prevent categorisation of individuals based on sensitive characteristics such as ethnicity, as well as to remove data after it has exceeded a specified storage timeframe or to restrict third party access to particular categories of data (Leenes et al., 2017). However, several implementation challenges must also be tackled to govern AI systems through code effectively. Many legal and ethical rules cannot be translated into explicit code, which includes 'subtle exceptions' that often require value judgements and consideration of contextual factors for interpretation, and the resulting machine interpretation also depends on how it was initially designed (Leenes & Lucivero, 2014; Mulligan & Bamberger, 2018). Other risks that governments need to manage from such an approach are potential manipulations of encoded rules to 'subvert regulatory aims', which is easily masked by the opacity of ML processes (Mulligan & Bamberger, 2018). Common to recent studies in their proposed frameworks for AI governance is the
emphasis on building broad societal consensus around AI ethical principles and ensuring accountability, but there is a need for studies examining how these frameworks can be implemented in practice. Gasser and Almeida's (2017) three-tiered framework comprises a technical layer involving the AI system processes and data structures, a layer for the ethical design of AI, and the third layer encompassing AI's societal implications and the role of regulation and legislation. Rahwan (2018) proposes extending the 'human-in-theloop' approach to a broader 'society-in-the-loop' approach where society is first responsible for finding consensus on the values that should shape AI and the distribution of benefits and costs among different stakeholders. Other proposals centre on the need for greater centralisation and cross-cultural cooperation to improve coordination among national approaches (Cihon et al., 2020; Óhéigeartaigh, Whittlestone, Liu, Zeng, & Liu, 2020). Cihon et al. (2020) propose a framework to centralise the current fragmented state

of international AI governance, outlining methods to monitor and coordinate national approaches and examining the disadvantages of centralisation relative to decentralisation, such as regulatory lags and limited adaptability to rapidly evolving risks. Among these approaches, there are increasing calls to produce more concrete specifications on implementing these governance frameworks in practice and identifying the parties in government that are responsible for leading different aspects of AI governance (Wirtz et al., 2020).

5. Overview of the special issue articles

The articles in the special issue tackle the issues of governing AI and robotics through case studies and comparative analyses addressing gaps in the literature pertaining to examining the risks and benefits of deploying AI in different applications and sectors, identifying dominant ideals envisioned in the meta-discourse on AI governance and their implications. Furthermore, exploration of new legal/regulatory/governance approaches taken by various countries/governments to govern AI and examination of approaches which can enhance implementation of AI and cross-country comparisons of AI governance approaches is conducted. Authors have explored the limitations/effectiveness of different AI governance approaches through case studies and derived key lessons to facilitate policy learning. Below a brief summary of the articles in the special issue on the Governance of AI and Robotics is presented.

5.1 Framing governance for a contested emerging technology: insights from AI policy (Ulnicane, Knight, Leach, Stahl, & Wanjiku, 2021) Whether AI develops in socially beneficial or problematic ways largely depends on public policies, governance arrangements and regulation. In recent years many national governments, international organisations, think tanks, consultancies, and civil society organisations have launched their AI strategies outlining benefits and risks as well as initial governance and regulatory frameworks. There are many questions regarding the regulation and governance of emerging disruptive technologies (Taeihagh et al., 2021). This article address the following question regarding the governance of AI: What governance and regulatory models are emerging to facilitate benefits and avoid risks of AI? What role do different actors – governments, international organisations, business, academia, and civil society – play in the emerging governance of AI? Do radical technological innovations in AI require radical or rather incremental innovations in governance? Are emerging governance and technology frameworks in different countries, regions and organisations converging or diverging and why? To study these questions, Ulnicane et al. (2021) draw on a dataset of more than 60 AI
strategies from 'national governments, international organisations, consultancies, think tanks and civil society organisations' worldwide. The interdisciplinary research framework of the article draws on concepts and approaches from Science, Technology and Innovation Studies, Policy Analysis and Political Science. It consists of two main pillars. First, to study risks, uncertainties, and unintended consequences of AI in policy documents, the concepts of policy framing and positive and negative expectations are used. Second, to study emerging governance and regulation frameworks outlined in the policy

documents, concepts of governance and governance models are used. AI policy documents are analysed to establish how they frame AI. The emerging governance of AI is analysed according to diverse governance models such as market, participatory, flexible, and deregulated (Peters, 2001). This article contributes to the studies of emerging disruptive technologies by analysing how the framing of risks and uncertainties of AI leads to the development of specific governance and regulatory arrangements mapping similarities and differences across countries, regions, and organisations.

5.2 Steering the governance of artificial intelligence: national strategies in perspective (Radu, 2021) The latest wave of AI developments is built around deep learning, speech and image recognition, and data analytics tools deployed daily, such as banking, e-learning, medical diagnosis – and more recently, smart vehicles (Radu, 2021). The article empirically investigates the governance responses to AI developments worldwide. The article examines recent AI national strategies worldwide to uncover the modalities through which regulation is articulated at the state level. The article highlights the key elements of the dominant regulatory discourses and compares AI national projects. Using content analysis and comparative methods, Radu (2021) examines the strategies of numerous countries to determine the articulation of AI governance and the co-production of political and socio-technological constructs. Building on the empirical evidence, the article highlights how the collective representation of a technical project is predefined politically and is encapsulated into a vision that is integrated into regulating the relevant sectors of society (Flichy, 2008). As such, the article contributes to the policy discourse around AI by conceptualisation the debates and critically examining the governance of AI and its complex effects in the future.

5.3 The Governance of Artificial Agency (Gahnberg, 2021)

Gahnberg argues for the need for new regulatory strategies to conceptualise the challenges of governing artificial agency. In the article, he argues that the notion of 'intelligence' is a vague metric of the success of the system and, the key characteristic of an AI system is its overall ability to act as an autonomous agent within a specific environment (Gahnberg, 2021). In the article, he posits that while different AI applications range from use in filtering Spam to killer robots, they share a common characteristic of being delegated authority to perform tasks regarding a specific objective(s). The article underlines the social and legal constraints for delegating this authority to
an artificial agent, including legal and ethical provisions that may prohibit tasking the agent to pursue certain objectives (e.g. using lethal force). Gahnberg (2021) further highlights the role of non-state actors in the development of governance mechanisms, such as in the case of development of standards and best practices developed by the technical community for fail-safe mechanisms or initiatives such as the Algorithmic Justice League that aims to mitigate the risks of algorithmic bias through creating inclusive training data. By examining the challenges of governing artificial agency, the article provides insights into the debates about AI governance.

5.4 Governing the adoption of robotics and autonomous systems in long-term care in Singapore (Tan & Taeihagh, 2021)

Autonomous systems and robotics have been dubbed as a viable technological solution to address the ever-increasing demand for long-term care globally, which is exacerbated by ageing populations (Robinson, MacDonald, & Broadbent, 2014; Tan et al., 2021). However, adopting new technologies in long-term care, like all other emerging technologies, involves unintended consequences and poses risks (Taeihagh et al., 2021). For instance, there are issues about safety, privacy, cybersecurity, liability, influence to the existing health workforce, patient's autonomy, patient's dignity, moral agency, social justice, and trust that need to be addressed by the government in the process of adoption of autonomous systems (Sharkey & Sharkey, 2012; Stahl & Coeckelbergh, 2016; Tan et al., 2021). Addressing these risks and ethical issues warrant the assessment of the government's policy capacity (Wu, Ramesh, & Howlett, 2015) and the various governance strategies that government chooses to deploy (no response, prevention-oriented, precaution-oriented, control-oriented, toleration-oriented and adaptation oriented) in the implementation of these autonomous systems (Li, Taeihagh, & De Jong, 2018, Li et al., 2021; Taeihagh & Lim, 2019). The article is a theoretically-informed empirical study that examines the adoption of autonomous systems in long-term care as a policy measure to arrest the issue of rising social care demand due to the ageing population by examining Singapore as a case study. The article reviews various existing and potential applications of autonomous systems in social care and long-term care for older people in Singapore. It then discusses the technological risks and ethical implications of deploying these systems to users, society, and the economy at large. Finally, it assesses the extent to which governance strategies influence the policy response adopted by the Singapore government, specifically in balancing the need for innovation and addressing uncertainties arising from the deployment of emerging technologies in healthcare. Insights gathered are important for policy learning, especially in understanding the governance process of novel and emerging technologies as possible solutions to address the demand-supply mismatch in the health sector. The analysis will showcase the interactive dynamics of policy capacity, governance strategies and policy response as three major ingredients in the governance of emerging technologies and calibrate their respective proportions to facilitate effective implementations of novel technologies in healthcare (Tan & Taeihagh, 2021).

5.5. Exploring governance dilemmas of disruptive technologies: the case of care robots in Australia and New Zealand (Dickinson, Smith, Carey, & Carey, 2021)

While there is a range of experiments ongoing worldwide applying AI and robotics in various care settings (Tan et al., 2021), with high expectations for positive outcomes, Dickinson et al. (2021) are concerned with the unexpected consequences and risks, and their article explores the use of robots in care

	services in the Australian and New Zealand. They point out that while there is a burgeoning literature on robots, much of it is around technical efficacy, acceptability, or the legal ramifications. They point out the lack of Downloaded from https://academic.oup.com/policyandsociety/article/40/2/137/6509315 by Oslo University user on 27 February 2023 POLICY AND SOCIETY 151 attention to the implementation of robots in care settings and their implications for policymaking. Informed by semi-structured interviews with policymakers, care providers, suppliers of robots and technology experts, Dickinson et al. (2021) elicit a series of 'dilemmas' of governance faced in this space and identify three dilemmas relating to independence and surveillance, the re-shaping of human interactions, the dynamics of caregiving and receiving care illustrating some of the tensions involved in the governance of robotics in care services and draw attention to the issues that governments need to address for the effective adoption of these technologies. 5.6 Law and tech collide: foreseeability, reasonableness, and advanced driver assistance systems (Leiman, 2021) There have been a number of fatalities involving automated vehicles since May 2016, which have been highly publicised. Simultaneously, millions of serious injuries and fatalities have occurred due to collisions by human-operated vehicles. In jurisdictions where compensations depend on the establishment of fault, many of the injured or their dependents will not recover any compensations (Leiman, 2021). In many Australian jurisdictions, the standard of care required presents significant challenges when applied to partly, highly, and fully automated vehicles (ibid). Leiman (2021) explores how the existing regulatory framework in Australia considers perceptions of risk in establishing legal liability in the case of motor vehicles and whether this approach can be applied to automated vehicles. The article examines whether the law itself may affect the perceptions of risk in view of the data generated by the automated vehicles and considers the efficacy of no-fault or hybrid schemes for legal liability in existence in Australia and compares them with the alternative legislation passed by the Parliament in the UK that imposes responsibility on the insurers. Leiman also discusses proposals concerning the role of government in assuring safety and fault in the case of Automated vehicles. 5.7 Co-regulating algorithmic disclosure for digital platforms. Di Porto and Zuppetta (2021) Di Porto and Zuppetta (2021) explore the increasing ability of IT-mediated platforms to adopt algorithmic decision making and whether disclosure self-regulation, as it currently stands at the EU level, is an appropriate strategy to address the risks posed by the increasing adoption of algorithmic decision making taken by private entities. While general data protection regulation (GDPR) requires platforms to provide information about the treatment of personal data by AI systems and self-assess the risks of data breaches, there is no reference made to the different abilities of the receiving entities to understand the meaning and consequences of algorithmic decisions. The article argues that the disclosure self-regulation should be rethought considering the new AI developments and points out that since final consumers and SMEs are unaware of the technical underpinnings of these platforms and the value of personal data, the regulators should step in and address this issue. Furthermore, as platforms have increasingly deployed AI and Big data, they have gained the ability to manipulate the information they produce, thus weakening the validity of consumers and small businesses choices (Di Downloaded from https://academic.oup.com/policyandsociety/article/40/2/137/6509315 by Oslo University user on 27 February 2023 152 A. TAEIHAGH Porto & Zuppetta, 2021). The authors advocate that the regulators should tackle information asymmetry, low bargaining power and wrong information and endorse an enforced co-regulatory approach that allows participation of platforms and consumers, and development of personalised discloses based on the needs of the consumers to empower them
Modified on	28/02/2023 15:07

Name	Governance of artificial intelligence
Number of Coding References	5
Number of Codes Coding	2
Coverage	30.77%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 11:04
Coded Text	Governing AI 4.1 Why AI governance is important Understanding and managing the risks posed by AI is crucial to realise the benefits of the technology. Increased efficiency and quality in the delivery of goods and services, greater autonomy and mobility for the elderly and disabled, and improved safety from using AI in safety-critical operations such as in healthcare, transport and emergency response are the many socio-economic benefits arising from AI that can propel smart and sustainable development (Agarwal, Gurjar, Agarwal, & Birla, 2015; Lim & Taeihagh, 2018; Yigitcanlar et al., 2018). Thus, as AI systems develop and increase in complexity, their risks and interconnectivity with other smart devices and systems will also increase, necessitating the creation of both specific governance mechanisms, such as for healthcare, transport and autonomous weapons, as well as a broader global governance framework for AI (Butcher & Beridze, 2019). 4.2 Challenges to AI governance The high degree of uncertainty and complexity of the AI landscape imposes many challenges for governments in designing and implementing effective policies to govern AI. Many challenges posed by AI stem from the nature of the problem, which are highly unpredictable, intractable and nonlinear, making it difficult for governments to formulate concrete objectives in their policies (Gasser & Almeida, 2017; Perry & Uuk, 2019). ML systems' inherent opacity and unpredictability pose technical challenges for governments in ensuring the accountability of AI. Firstly, the opacity of complex ML algorithms remains a major barrier to AI governance as it limits the extent of transparency, explainability and accountability that can be achieved in AI systems (Lim & Taeihagh, 2019; Mittelstadt et al., 2016). Even with mandated levels of transparency and explainability of algorithms, it is impossible for the experts themselves to interpret how certain algorithmic outputs are derived from its inputs and designing the algorithm to be more explainable reduces their complexity, which has been shown to undermine accuracy and performance (Felzmann, Villaronga, Lutz, & Tamò-Larrieux, 2019; Piano, 2020). This issue is highlighted as a severe limitation of the EU General Data Protection Regulation in increasing algorithmic transparency to tackle discrimination (Goodman & Flaxman, 2017). Algorithms are often kept intentionally opaque by their developers to prevent cyber-attacks and to safeguard trade secrets, which is legally justified by intellectual property rights, and the complexity of extensive datasets used by ML algorithms makes it nearly impossible to identify and remove all variables that are correlated with sensitive categories of personal data (Carabantes, 2020; Goodman & Flaxman 2016; Kroll et al., 2016). As most individuals lack sufficient technical literacy or the willingness to pay for accessing such expertise to help them to interpret these explanations, requirements by the likes of GDPR for mandated explanations are unlikely to inform and/or empower them (Carabantes, 2020; Felzmann et al., 2019). Secondly, ML decisions are unpredictable as they are derived from the data and can differ significantly with slight changes in inputs. The lack of human controllability over the behaviour of AI systems suggests the difficulty of assigning liability and accountability for harms resulting from software defects, as manufacturers and programmers often cannot predict the inputs and design rules that could yield unsafe or discriminatory outcomes (Kim et al. 2017; Kroll et al., 2016). Data governance is a key issue surrounding the debate on AI governance, as multiple organisational and technological challenges exist that impede effective control over data and attribution of responsibility for data-driven decisions made by AI systems (Janssen et al., 2020). Data fragmentation and lack of interoperability between systems limits an organisation's control over data flows throughout its entire life cycle and shared roles between different parties in data sharing clouds the chain of accountability and causation between AI-driven decisions/events and the parties involved in facilitating that decision (Janssen et al., 2020). Existing governance and regulatory frameworks are also ill-equipped to manage the societal problems introduced by AI due to the insufficient information required for understanding the technology and regulatory lags behind AI developments. Major technology companies and AI developers such as Google, Facebook, Microsoft, and Apple possess huge informational and resource advantages over governments in regulating AI, which significantly supersedes governments' traditional role in distributing and controlling resources in society (Guihot et al., 2017). The information asymmetries between technology companies and regulators compound the difficulty for the latter in

	understanding and applying new or existing legislation to AI applications (Taeihagh et al., 2021). Regulators are struggling to meet these informational gaps and are lagging behind due to rapid developments in the technology, which in turn leads to the formulation of laws that are 'too general' or 'vague' and lack specificity to effectively regulate the technology (Guihot et al., 2017; Larsson, 2020). In particular, lawmakers can be deterred from outlining specific rules and duties for algorithm programmers to allow for future experimentation and modifications to code to improve the software, but doing so provides room for programmers to evade responsibility and accountability for the system's resulting behaviour in society (Kroll et al., 2016). These challenges demonstrate how the four governing resources traditionally used by governments for regulation are insufficient to manage the risks arising from AI and the need for governments to find new Downloaded from https://academic.oup.com/policyandsociety/article/40/2/137/6509315 by Oslo University user on 27 February 2023 POLICY AND SOCIETY 145 ways of acquiring information and devising effective policies that can adapt to the evolving AI landscape (Guihot et al., 2017; Wirtz et al., 2020). Amidst the issues with 'hard' regulatory frameworks, industry bodies and govern- ments have increasingly adopted self-regulatory or 'soft law' approaches to govern AI design, but they remain limited in their effectiveness. Soft law approaches refer to 'nonbinding norms and techniques' that create 'substantive expectations that are not directly enforceable'. Industry bodies have released voluntary standards, guidelines, and codes of conduct (IEEE 2019), and governments alike have formed expert committees to devise AI strategies and released multiple AI ethics guidelines and governance frameworks (PDPC et al. 2020; AI; Hleg, 2019; Jobin et al., 2019). Guidelines can be amended and adapt more rapidly than traditional regulation and thus are advantageous in keeping pace with technological developments (Hemphill, 2020; Larsson, 2020). This approach has been adopted in response to previous emerging technologies and can promote ethical, fair, and non-discriminatory practices in AI design, but many issues exist regarding their efficacy. Firstly, the voluntary nature of selfregulatory initiatives cannot assure that the outlined principles will always be adhered to, particularly as they are often not subject to uniform enforcement standards (Butcher & Beridze, 2019; Hemphill, 2016). Secondly, governments will face the challenge of ensuring consistent application of these guidelines in designing the same AI technology across different sectors if the principles differ across multiple guidelines and are not well-coordinated with regulations (Cath et al. 2018; Guihot et al., 2017). In addition, self-regulation alone could be insufficient and even undesirable for AI governance due to their inability to ensure inclusivity and representation of diverse stakeholders. The deep involvement of industry stakeholders in developing ethical principles and regulations for AI raises concerns that corporate interests dominate AI regulations, which has been an ongoing critique of self-regulatory initiatives to govern emerging technologies in general (Hemphill, 2016). For instance, major technology companies have exerted significant influence over the framing of AI issues and the formulation of AI policy, such as through lobbying efforts and their inclusion in AI expert groups formed by governments (Cath et al. 2018; Jobin et al., 2019). Studies have highlighted the risks of regulatory capture by AI industries due to the significant informational advantages of AI developers that make their latter's technological expertise 'particularly valuable' for regulators (Guihot et al., 2017). Furthermore, the 'inscrutable' and highly opaque nature of ML algorithms could be used by corporations to legitimise deep industry involvement in AI regulations and the choices made behind these regulations are often made away from public scrutiny. The unbridled influence of corporations, as well as key political figures in the AI landscape, could exacerbate power imbalances and social inequalities, as the ideologies and interests of an elite few could manifest themselves through the design of AI and the decisions that they make in society (Jobin et al., 2019). To ensure greater inclusivity and diversity in AI governance, more research is required to examine the key actors, their roles, the dominant ideas, and values promoted in AI policies, whether there is a global convergence in these values across different countries and the degree to which these values reflect society's interests or are politically motivated (Jobin et al., 2019; Mulligan & Bamberger, 2018).
Modified on	28/02/2023 15:06

Name	Governance of artificial intelligence
Number of Coding References	5
Number of Codes Coding	2
Coverage	30.77%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 11:04
Coded Text	The rapid developments in Artificial Intelligence (AI) and the intensification in the adoption of AI in domains such as autonomous vehicles, lethal weapon systems, robotics and alike pose serious challenges to governments as they must manage the scale and speed of socio-technical transitions occurring. While there is considerable literature emerging on various aspects of AI, governance of AI is a significantly underdeveloped area. The new applications of AI offer opportunities for increasing economic efficiency and quality of life, but they also generate unexpected and unintended consequences and pose new forms of risks that need to be addressed. To enhance the benefits from AI while minimising the adverse risks, governments worldwide need to understand better the scope and depth of the risks posed and develop regulatory and governance processes and structures to address these challenges. This introductory article unpacks AI and describes why the Governance of AI should be gaining far more attention given the myriad of challenges it presents. It then summarises the special issue articles and highlights their key contributions. This special issue introduces the multifaceted challenges of governance of AI, including emerging governance approaches to AI, policy capacity building, exploring legal and regulatory challenges of AI and Robotics, and outstanding issues and gaps that need attention. The special issue showcases the state-of-the-art in the governance of AI, aiming to enable researchers and practitioners to appreciate the challenges and complexities of AI governance and highlight future avenues for exploration. 1. Introduction Artificial intelligence (AI) is rapidly changing how transactions and social interactions are organised in society today. AI systems and the algorithms supporting their operations play an increasingly important role in making value-laden decisions for society, ranging from clinical decision support systems that make medical diagnoses, policing systems that predict the likelihood of criminal activities and filtering algorithms that categorise and provide personalised content for users (Helbing, 2019; Mittelstadt, Allo, Taddeo, Wachter, & Floridi, 2016). The ability to mimic or rival human intelligence in complex problem-solving sets AI apart from other technologies, as many cognitive tasks CONTACT Araz Taeihagh spparaz@nus.edu.sg; araz.taeihagh@new.oxon.org Lee Kuan Yew School of Public Policy, National University of Singapore, 469B Bukit Timah Road, Li Ka Shing Building, Level 2, #02-10 259771 Singapore © 2021 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group. This is an Open Access article distributed under the terms of the Creative Commons Attribution-NonCommercial License (http:// creativecommons.org/licenses/by-nc/4.0/), which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited. KEYWORDS Governance; artificial intelligence; AI; robotics; public policy Downloaded from https://academic.oup.com/policyandsociety/article/40/2/137/6509315 by
Modified on	28/02/2023 15:05

Name	Governance of artificial intelligence
Number of Coding References	5
Number of Codes Coding	2
Coverage	30.77%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 11:04
Coded Text	Understanding the risks of AI Many scholars highlight the safety issues that can arise from deploying AI in various domains. A major challenge faced by most AI applications to date stems from their lack of generalizability to different contexts, in which they can face unexpected situations widely referred to as 'corner cases' that the system had not been trained to handle (Bostrom & Ludkowsky 2014; Lim & Taeihagh, 2019; Pei, Cao, Yang, & Jana, 2017). For instance, fatal crashes have already resulted from trials of Tesla's partially autonomous vehicles due to the system's misinterpretation of unique environmental conditions that it had not previously experienced during testing. While various means of detecting these corner cases in advance have been devised, such as simulating data on many possible driving situations for autonomous vehicles, not all scenarios can be covered or even envisioned by the human designers (Bolte, Bar, Lipinski, & Fingscheidt, 2019; Pei et al., 2017). Due to the complexity and adaptive nature of ML processes, it is difficult for humans to articulate or understand why and how a decision was made, which hinders the identification of corner case behaviours in advance (Mittelstadt et al., 2016). As ML decisions are highly data-driven and unpredictable, the system can exhibit vastly different behaviours in response to almost identical inputs that make it difficult to specify 'correct' behaviours and verify their safety in advance (Koopman & Wagner, 2016). In particular, scholars point out potential safety hazards that can also arise from the interaction between AI systems and their users due to the problem of automation bias, where humans afford more credibility to automated decisions due to the latter's seemingly objective nature and, thus, grow complacent and display less cautious behaviour while using AI systems (Osoba & Welser, 2017; Taeihagh & Lim, 2019). Thus, human-machine interfaces significantly shape the degree of safety, particularly in social settings that involve frequent interactions with users such as robots for personal care, autonomous vehicles, and service providers. The decision-making autonomy of AI significantly reduces human control over their decisions, creating new challenges for ascribing responsibility and legal liability for the harms imposed by AI on others. Existing legal frameworks for the ascribing of responsibility and liability for machine operation treat machines as tools that are controlled by their human operator based on the assumption that humans have a certain degree of control over the machine's specification (Matthias 2004; Leenes & Lucivero, 2014). However, as AI relies largely on ML processes that learn and adapt their own rules, humans are no longer in control and, thus, cannot be expected to always bear responsibility for AI's behaviour. Under strict product liability, manufacturers and software designers could be subject to liability for manufacturing defects and design defects, but the unpredictability of ML decisions implies that many erroneous decisions made by AI are beyond the control of and cannot be anticipated by these parties (Butcher & Beridze, 2019; Kim et al. 2017; Lim & Taeihagh, 2019). This raises critical questions regarding the extent to which different parties in the AI supply chain will be held liable in different accident scenarios and the degree of autonomy that is sufficient to 'limit' the responsibility of these parties for such unanticipated accidents (Osoba & Welser, 2017; Wirtz, Weyerer, & Sturm, 2020). It is also widely recognised that excessive liability risks can hinder long-run innovation and improvements to the technology, which highlights a major issue regarding how governments can structure new liability frameworks that balance the benefits of promoting innovation with the moral imperative of protecting society from the risks of emerging technologies (Leenes et al., 2017). Given the value-laden nature of the decisions automated by algorithms in various aspects of society, AI systems can potentially exhibit behaviours that conflict with societal values and norms, prompting concerns regarding the ethical issues that can arise from AI's rapid adoption. One of the most intensively discussed issues across industry and academia is the potential for algorithmic decisions to be biased and discriminatory. As ML algorithms can learn from data gathered from society to make decisions, they could not only conflict with the original ethical rules they were programmed with but also reproduce the inequality and discriminatory patterns of society that is contained in such data (Goodman & Flaxman, 2017; Osoba & Welser, 2017; Piano, 2020). If sensitive personal characteristics such as gender or race in the data are used to classify individuals, and some characteristics are found to negatively correlate with the outcome that the algorithm is designed to optimise, the individuals categorised with these traits will be penalised over others with different group characteristics (Liu 2018). This could yield disparate outcomes in terms of

	risk exposure and access to social and economic benefits. Bias can also be introduced through the human designer in constructing the algorithm, and even if sensitive attributes are removed from the data, there are techniques for ML algorithms to use 'probabilistically inferred' variables as a proxy for sensitive attributes, which is much harder to regulate (Kroll et al., 2016; Osoba & Welser, 2017). The risk of bias and discrimination stemming from the optimisation process in AI algorithms reflects a dominant concern surrounding discussions of fairness in AI governance – the trade-off between equity and efficiency in algorithmic decision-making – (Sætra, 2020) and how a balance can be struck to produce socially desirable outcomes catering to the different groups' ethical preferences remains subject to debate.
Modified on	28/02/2023 15:06

Name	Governance of artificial intelligence
Number of Coding References	5
Number of Codes Coding	2
Coverage	30.77%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 11:04
Coded Text	While there is considerable literature emerging on various aspects of AI, governance of AI is an emerging but significantly underdeveloped area. To enhance the benefits of AI while minimising the adverse risks they pose, governments worldwide need to understand better the scope and depth of the risks posed. There is a need to reassess the efficacy of traditional governance approaches such as the use of regulations, taxes, and subsidies, which may be insufficient due to the lack of information and constant changes (Guihot, Matthew, & Suzor, 2017), and the speed and scale of adoption of AI threatens to outpace the regulatory responses to address the concerns raised (Taeihagh, Ramesh, & Howlett, 2021). As such, governments face mounting pressures to design and establish new regulatory and governance structures to deal with these challenges effectively. The increasing recognition of AI governance across government, the public (Chen, Kuo, & Lee, 2020; Zhang & Dafoe, 2019, 2020) and industry is evident from the emergence of new governance frameworks in the meta-discourse on AI such as adaptive and hybrid governance (Leiser & Murray 2016; Linkov et al., 2018; Tan & Taeihagh, 2021b), and selfregulatory initiatives such standards and voluntary codes of conduct to guide AI design (Guihot et al., 2017; IEEE 2019). The first half of 2018 saw the release of new AI strategies from over a dozen countries, significant boosts in pledged financial support by governments for AI, and the heightened involvement of industry bodies in AI regulatory development (Cath, 2018), raising further questions regarding what ideas and interests Downloaded from https://academic.oup.com/policyandsociety/article/40/2/137/6509315 by Oslo University user on 27 February 2023 POLICY AND SOCIETY 139 should shape AI governance to ensure inclusion and diverse representation of all members of society (Hemphill, 2016; Jobin, Ienca, & Vayena, 2019). This special issue introduces the multifaceted challenges of governance of Artificial Intelligence, including emerging governance approaches to AI, policy capacity building, and exploring legal and regulatory challenges of AI and Robotics. This introduction unpacks AI and describes why the Governance of AI should be gaining far more attention given the myriad of challenges it presents. The introduction then summarises of the special issue articles are presented, and their key contributions are highlighted. Thanks to the diverse set of articles comprising this special issue; it highlights the state-of-the-art in the governance of AI and discusses the outstanding issues and gaps that need attention, aiming to enable researchers and practitioners to appreciate the challenges that AI brings better and understand the complexities of governance of AI and future avenues for exploration. 2. AI – background and recent trends Conceptions of AI date back to earlier efforts in developing artificial neural networks to replicate human intelligence, which can be referred to as the ability to interpret and learn from the information. Originally designed to understand neuron activity in the human brain, more sophisticated neural networks were developed in the late 20th century with the aid of advancements in processing power to solve problems such as image and speech recognition (Izenman 2008). These efforts led to the introduction of the concept of AI as computer programs (or machines) that can perform predefined tasks at much higher speeds and accuracy. In the most recent wave of AI developments facilitated by advancements in big data analytics, AI capabilities have expanded to include computer programs that can learn from vast amounts of data and make decisions without human guidance, commonly referred to as Machine-learning (ML) algorithms (Izenman 2008). Unlike earlier algorithms that rely on pre-programmed rules to execute repetitive tasks, ML algorithms are designed with rules about how to learn from data that involves ‘inferential reasoning’, ‘perception’, ‘classification’, and ‘optimisation’ to replicate human decisionmaking (Bathaee, 2018; Linkov et al., 2018). The learning process involves feeding these algorithms with large data sets, from which they seek and test complex mathematical correlations between candidate variables to maximise predictions of a specified outcome (Kleinberg et al. 2018; Brauneis & Goodman, 2018). As these algorithms adapt their decision-making rules with more experience, ML-driven decisions are primarily dependent on the data rather than on pre-programmed rules and, thus, typically cannot be predicted well in advance (Mittelstadt et al., 2016). Among AI experts and researchers, there is a broad consensus that AI still ‘falls short’ of human cognitive abilities, and most AI applications that have been successful to date stem from ‘narrow AI’ or ‘weak AI’, which refer to AI

	applications that can perform tasks in specific and restricted domains, such as chess, image, and speech recognition (Bostrom & Ludkowsky 2014; Lele, 2019b). Narrow AI is expected to automate and replace many mid-skill professions due to their ability to execute routine, cognitive tasks at much higher speeds and accuracy than their human counterparts (Lele, 2019bb; Linkov et al., 2018). In future, it is expected that this form of AI will eventually achieve 'General AI' or 'artificial general intelligence', a level of intelligence Downloaded from https://
Modified on	28/02/2023 15:06
Name	Governance of artificial intelligence Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	
Name	High Performance Data Engineering Everywhere
Number of Coding References	14
Number of Codes Coding	4
Coverage	12.10%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	
Modified on	07/02/2022 16:41

Name	High Performance Data Engineering Everywhere
Number of Coding References	14
Number of Codes Coding	4
Coverage	12.10%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	
Modified on	07/02/2022 16:41

Name	High Performance Data Engineering Everywhere
Number of Coding References	14
Number of Codes Coding	4
Coverage	12.10%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	C. Cylon, Spark vs. Dask Figure 9 (a) shows a strong scaling wall-clock time com- parison between Cylon, Spark and Dask. The same strong scaling setup for Inner-Joins was used in this comparison. When comparing with Dask and Spark, Cylon performs better than them on the wall-clock time. For this 200 million line join, it scales better than both of the other frameworks. It should be noted that Dask failed to complete for the world sizes 1 and 2, even when doubling the resources. It continued to fail even with the factory LocalCluster settings, with higher memory. Cylon shows better strong scaling, reaching a higher in- dividual speedup. As shown in Table II for a single worker (serial) Inner-joins, Cylon Hash, Cylon Sort, and Spark took 141s, 164s and 587s respectively. For Union, Cylon and Spark took 34s and 75s respectively. Thus not only does Cylon show better scaling, it achieves a superior wall-clock speed up because its serial case wall-clock time is an improvement on Spark. Figure 9 (b) shows the results for the Union (Distinct) operation. Unfortunately Dask (as of its latest release) does not have a direct API for distributed Union operation. As a result the comparison is limited to Spark and Cylon. As the graph depicts, Cylon performs better than Spark, with more than 2x better performance at each experiment.
Modified on	07/02/2022 16:42

Name	High Performance Data Engineering Everywhere
Number of Coding References	14
Number of Codes Coding	4
Coverage	12.10%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	Cylon1 is a data engineering toolkit designed to work with AI/ML systems and integrate with data processing systems. This vision is highlighted in Figure 1 where Cylon is shown to support common data structures and systems. It can be deployed either as a library or a framework. Big Data systems like Apache Spark, Apache Flink and Twister2 [2] can use Cylon to boost the performance in the ETL pipeline. For AI/ML systems like PyTorch [12], Tensorflow [13] and MXNet [14], it acts as a library to enhance ETL performance. Additionally, Cylon is being expanded to perform as a generic framework for supporting ETL and efficient distributed modeling of AI/ML workloads. Cylon currently provides a set of distributed data-parallel operators to extract, transform and load structured relational data. These operators are exposed as APIs in multiple programming languages (e.g., C++, Python, Java) that are commonly used in Machine Learning and Artificial Intelligence platforms, enabling tight integration with them. When an operator is invoked in any of these platforms, that invocation is delegated to the ”Core Cylon” framework, which implements the actual logic to perform the operation in a distributed setting.
Modified on	07/02/2022 16:40

Name	High Performance Data Engineering Everywhere
Number of Coding References	14
Number of Codes Coding	4
Coverage	12.10%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	One important question is whether those existing Big Data frameworks utilize the full potential of the computing power and parallelism available to process data. Both Big Data and AI/ML applications spend a goodly amount of time preprocessing data. Minimizing the pre-processing time clearly increases the throughput of these applications. Productivity is another important aspect of such frameworks. Most available data analytics tools are implemented using a rapid programming language such as Java, Python or R. This allows data engineers to develop applications without diverging into the details of complex distributed data processing algorithms. Still, we rarely see these two aspects (high performance and productivity) meet each other in the existing Big Data frameworks [5]. We have also seen the world increasingly moving towards user-friendly frameworks such as NumPy [6], Python Pandas [7] or Dask [8]. Big Data frameworks have been trying to match this by providing similar APIs (for example, PySpark, Dask-Distributed). But this comes at the cost of performance owing to the overheads that arise from switching between runtimes.
Modified on	07/02/2022 16:39

Name	High Performance Data Engineering Everywhere
Number of Coding References	14
Number of Codes Coding	4
Coverage	12.10%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	One of the main goals of Cylon is to be a versatile library which facilitates data processing as a function (DPAF) and thus provide efficient data engineering across different systems. When working over multiple systems, data representation and conversion is a key factor affecting performance and interoperability. Cylon internally uses Apache Arrow data structure, which is supported by many other frameworks such as Apache Spark, TensorFlow, and PyTorch. Apache Arrow can be converted into other popular data structures such as NumPy and Pandas efficiently. In addition our core data structures can work with zero copy across languages. For example, when Cylon creates a table in CPP, it is available to the Python or Java interface without need for data copying. Cylon C++ kernels efficiently support data loading and data processing. These functions can be used either in distributed or local setting. Most of the deep learning libraries like PyTorch, Tensorflow and MXNet are designed on top of such high performance kernels. Cylon APIs are made available to the user in a similar manner. Such designs lead to lower frictions in system integration. With these design principles, we envision the following scenarios where Cylon could work with other systems to create rich applications.
Modified on	07/02/2022 16:41

Name	High Performance Data Engineering Everywhere
Number of Coding References	14
Number of Codes Coding	4
Coverage	12.10%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	We believe that a data processing framework focused on high performance and productivity would provide a more robust and efficient data engineering pipeline. In this paper we introduce Cylon: a high-performance, MPI (Message Passing Interface)-based distributed memory data parallel library for processing structured data. Cylon implements a set of relational operators to process data. While ”Core Cylon” is implemented using system level C/C++, multiple language interfaces (Python and Java (R in future)) are provided to seamlessly integrate with existing applications, enabling both data and AI/ML engineers to invoke data processing operators in a familiar programming language. Large-scale ETL operations most commonly involve map- ping data to distributed relations and applying queries on them. There are distributed table APIs implemented on top of Big Data frameworks such as Apache Spark [9] and Apache Flink [10] which are predominantly based on Java program-
Modified on	07/02/2022 16:40

Name	High Performance Data Engineering Everywhere
Number of Coding References	14
Number of Codes Coding	4
Coverage	12.10%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	We discuss Cylon's architecture in detail, and reveal how it can be imported as a library to existing applications or operate as a standalone framework. Initial experiments show that Cylon enhances popular tools such as Apache Spark and Dask with major performance improvements for key operations and better component linkages. Finally, we show how its design enables Cylon to be used cross-platform with minimum overhead, which includes popular AI tools such as PyTorch, Tensorflow, and Jupyter notebooks.
Modified on	07/02/2022 16:39

Name	High Performance Data Engineering Everywhere
Number of Coding References	14
Number of Codes Coding	4
Coverage	12.10%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	31/01/2022 14:44
Coded Text	Large-scale data processing/engineering has gone through remarkable transformations over the past few decades. Developing fast and efficient Extract, Transform and Load frameworks on commodity cloud hardware has taken center stage in handling the information explosion and Big Data. Subsequently, we have seen a wide adoption of Big Data frameworks from Apache Hadoop [1], Twister2 [2], and Apache Spark [3] to Apache Flink [4] in both enterprise and research communities. Today, Artificial Intelligence (AI) and Machine Learning (ML) have further broadened the scope of data engineering,
Modified on	07/02/2022 16:39

Name	High Performance Data Engineering Everywhere
Number of Coding References	14
Number of Codes Coding	4
Coverage	12.10%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	31/01/2022 14:44
Coded Text	The amazing advances being made in the fields of machine and deep learning are a highlight of the Big Data era for both enterprise and research communities. Modern applications require resources beyond a single node's ability to provide. However this is just a small part of the issues facing the overall data processing environment, which must also support a raft of data engineering for pre- and post-data processing, communication, and system integration. An important requirement of data analytics tools is to be able to easily integrate with existing frameworks in a multitude of languages, thereby increasing user productivity and efficiency. All this demands an efficient and highly distributed integrated approach for data processing, yet many of today's popular data analytics tools are
Modified on	07/02/2022 16:38

Name	High Performance Data Engineering Everywhere
Number of Coding References	14
Number of Codes Coding	4
Coverage	12.10%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	03/02/2022 10:07
Coded Text	C. Cylon, Spark vs. Dask Figure 9 (a) shows a strong scaling wall-clock time com- parison between Cylon, Spark and Dask. The same strong scaling setup for Inner-Joins was used in this comparison. When comparing with Dask and Spark, Cylon performs better than them on the wall-clock time. For this 200 million line join, it scales better than both of the other frameworks. It should be noted that Dask failed to complete for the world sizes 1 and 2, even when doubling the resources. It continued to fail even with the factory LocalCluster settings, with higher memory. Cylon shows better strong scaling, reaching a higher in-dividual speedup. As shown in Table II for a single worker (serial) Inner-joins, Cylon Hash, Cylon Sort, and Spark took 141s, 164s and 587s respectively. For Union, Cylon and Spark took 34s and 75s respectively. Thus not only does Cylon show better scaling, it achieves a superior wall-clock speed up because its serial case wall-clock time is an improvement on Spark. Figure 9 (b) shows the results for the Union (Distinct) operation. Unfortunately Dask (as of its latest release) does not have a direct API for distributed Union operation. As a result the comparison is limited to Spark and Cylon. As the graph depicts, Cylon performs better than Spark, with more than 2x better performance at each experiment.
Modified on	07/02/2022 16:42

Name	High Performance Data Engineering Everywhere
Number of Coding References	14
Number of Codes Coding	4
Coverage	12.10%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	03/02/2022 10:07
Coded Text	We believe that a data processing framework focused on high performance and productivity would provide a more robust and efficient data engineering pipeline. In this paper we introduce Cylon: a high-performance, MPI (Message Passing Interface)-based distributed memory data parallel library for processing structured data. Cylon implements a set of relational operators to process data. While ”Core Cylon” is implemented using system level C/C++, multiple language interfaces (Python and Java (R in future)) are provided to seamlessly integrate with existing applications, enabling both data and AI/ML engineers to invoke data processing operators in a familiar programming language. Large-scale ETL operations most commonly involve map- ping data to distributed relations and applying queries on them. There are distributed table APIs implemented on top of Big Data frameworks such as Apache Spark [9] and Apache Flink [10] which are predominantly based on Java program-
Modified on	07/02/2022 16:43
Name	High Performance Data Engineering Everywhere
Number of Coding References	14
Number of Codes Coding	4
Coverage	12.10%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	C. Cylon, Spark vs. Dask Figure 9 (a) shows a strong scaling wall-clock time com- parison between Cylon, Spark and Dask. The same strong scaling setup for Inner-Joins was used in this comparison. When comparing with Dask and Spark, Cylon performs better than them on the wall-clock time. For this 200 million line join, it scales better than both of the other frameworks. It should be noted that Dask failed to complete for the world sizes 1 and 2, even when doubling the resources. It continued to fail even with the factory LocalCluster settings, with higher memory. Cylon shows better strong scaling, reaching a higher in- dividual speedup. As shown in Table II for a single worker (serial) Inner-joins, Cylon Hash, Cylon Sort, and Spark took 141s, 164s and 587s respectively. For Union, Cylon and Spark took 34s and 75s respectively. Thus not only does Cylon show better scaling, it achieves a superior wall-clock speed up because its serial case wall-clock time is an improvement on Spark. Figure 9 (b) shows the results for the Union (Distinct) operation. Unfortunately Dask (as of its latest release) does not have a direct API for distributed Union operation. As a result the comparison is limited to Spark and Cylon. As the graph depicts, Cylon performs better than Spark, with more than 2x better performance at each experiment.
Modified on	07/02/2022 16:42

Name	High Performance Data Engineering Everywhere
Number of Coding References	14
Number of Codes Coding	4
Coverage	12.10%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	We believe that a data processing framework focused on high performance and productivity would provide a more robust and efficient data engineering pipeline. In this paper we introduce Cylon: a high-performance, MPI (Message Passing Interface)-based distributed memory data parallel library for processing structured data. Cylon implements a set of relational operators to process data. While ”Core Cylon” is implemented using system level C/C++, multiple language interfaces (Python and Java (R in future)) are provided to seamlessly integrate with existing applications, enabling both data and AI/ML engineers to invoke data processing operators in a familiar programming language. Large-scale ETL operations most commonly involve map- ping data to distributed relations and applying queries on them. There are distributed table APIs implemented on top of Big Data frameworks such as Apache Spark [9] and Apache Flink [10] which are predominantly based on Java program-
Modified on	07/02/2022 16:43
Name	How does Machine Learning Change Software Development Practices~
Number of Coding References	12
Number of Codes Coding	2
Coverage	6.72%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	
Modified on	07/02/2022 16:36

Name	How does Machine Learning Change Software Development Practices~
Number of Coding References	12
Number of Codes Coding	2
Coverage	6.72%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	4.1 RQ1. Differences in Software Engineering Practices 4.1.1 Software Requirements Nearly every interviewee made a strong statement about differences between the requirements of ML systems versus the requirements of non-ML systems. In essence, requirements of ML systems are generally data-driven—closely coupled with existing large-scale data of a particular application context.
Modified on	07/02/2022 16:35

Name	How does Machine Learning Change Software Development Practices~
Number of Coding References	12
Number of Codes Coding	2
Coverage	6.72%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	4.1.2 Software Design Interviewees repeatedly mentioned that the design of ML systems and non-ML software systems differently place emphasis in a few ways. First, the high-level architectural design for ML systems is relatively fixed [7 S4]. As P3 summarized, the architecture of MLsystems typically consists of data collection, data cleaning, feature engineering, data modeling, execution, and deployment. In contrast, the architectural design for non-ML software systems is a more creative process,
Modified on	07/02/2022 16:35

Name	How does Machine Learning Change Software Development Practices~
Number of Coding References	12
Number of Codes Coding	2
Coverage	6.72%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	4.1.3 Software Construction andTools Interviewees reported several differences between ML systems and non-ML software systems in terms of coding practice. First, the coding workload of ML systems is low compared to non -ML software systems [S8]. Instead of coding for implementing particular functionalities in non-ML software systems, coding in ML systems generally includes data processing (e.g., transformation, cleansing, and encoding), feature analysis (e.g., visualization and statistical testing), and data modeling (e.g., hyperparameters selection and model training). P14 pointed at the availability of useful frameworks and libraries
Modified on	07/02/2022 16:35

Name	How does Machine Learning Change Software Development Practices~
Number of Coding References	12
Number of Codes Coding	2
Coverage	6.72%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	4.1.4 Software Testing andQuality Although software quality is important in both ML and non-ML systems, the practice of testing appears to differ significantly. One significant difference exists in the reproducibility of test results. In contrast to non-ML software systems, the testing results of ML systems is hard to reproduce because of a number of sources of randomness [S13]. As P8 explained:
Modified on	07/02/2022 16:35

Name	How does Machine Learning Change Software Development Practices~
Number of Coding References	12
Number of Codes Coding	2
Coverage	6.72%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	4.1.5 Software Maintenance andConfiguration Management Interviewees suggested that less effort may be required in the maintenance for ML systems than traditional software
Modified on	07/02/2022 16:36

Name	How does Machine Learning Change Software Development Practices~
Number of Coding References	12
Number of Codes Coding	2
Coverage	6.72%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	4.1.6 Software Engineering Process andManagement ML and non-ML systems differ in the processes that are followed in their development. As P2 suggested, during the step of context understanding, it is important to communicate with other stakeholders about what machine learning is and is not capable of. P6 mentioned that some stakeholders might misunderstand what machine learning is capable of: They usually overestimate the effect of machine learning technologies. Machine learning is not a silver bullet; the effect highly depends on the data they have [3 3 S21].
Modified on	07/02/2022 16:36

Name	How does Machine Learning Change Software Development Practices~
Number of Coding References	12
Number of Codes Coding	2
Coverage	6.72%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	Adding an ability for a system to learn inherently adds uncertainty into the system. Given the rising popularity of incorporating machine learning into systems, we wondered how the addition alters software development practices. We performed a mixture of qualitative and quantitative studies with 14 interviewees and 342 survey respondents from 26 countries across four continents to elicit significant differences between the development of machine learning systems and the development of non-machinelearning systems. Our study uncovers significant differences in various aspects of software engineering (e.g., requirements, design, testing, and process) and work characteristics (e.g., skill variety, problem solving and task identity). Based on our findings, we highlight future research directions and provide recommendations for practitioners
Modified on	07/02/2022 16:33

Name	How does Machine Learning Change Software Development Practices~
Number of Coding References	12
Number of Codes Coding	2
Coverage	6.72%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	RQ1. How does the incorporation of ML into a system impact software development practices? Is developing ML systems different from developing non-ML systems? How does it differ? If developing ML systems is indeed different from non-ML software development, past software engineering research may need to be expanded to better address the unique challenges of developing ML systems; previous tools and practices may become inapplicable to the development of ML systems; software engineering educators may need to teach different skills for the development of ML systems. Our study found several statistically significant differences in software engineering practices between ML and non-ML development: ~ Requirements: Collecting requirements in the development of ML systems involves more preliminary experiments, and creates a need for the predictable degradation in the performance. ~ Design: Detailed design of ML systems is more timeconsuming and tends to be conducted in an intensively iterative way. ~ Testing and Quality: Collecting a testing dataset requires more effort for ML development; Good performance3 during testing cannot guarantee the performance of ML systems in production. ~ Process and Management: The availability of data limits the capability of ML systems; Data processing is more important to the success of the whole process.
Modified on	07/02/2022 16:34

Name	How does Machine Learning Change Software Development Practices~
Number of Coding References	12
Number of Codes Coding	2
Coverage	6.72%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	To tackle the ML specific issues, recent studies have put effort into building tools for testing [26], [30], [36], [39] and debugging [16], [27], [28] of machine learning code, and creating frameworks and environments to support development of ML systems [3], [6]. Despite these efforts, software practitioners still struggle to operationalize and standardize the software development practices of systems using ML.2 Operationalization and standardization of software development practices are essential for cost-effective development of high-quality and reliable ML systems. How does machine learning change software development practices? To systematically explore the impact, we performed a mixture of qualitative and quantitative studies to investigate the differences in software development that arise from machine learning.
Modified on	07/02/2022 16:34
Name	How does Machine Learning Change Software Development Practices~
Number of Coding References	12
Number of Codes Coding	2
Coverage	6.72%
Folder Location	Codes\\Snowballing
Parent Name	Codes\\Snowballing\\Snowballing
Created on	21/06/2022 14:00
Coded Text	~ Design. Both studies agree that maintaining modularization in ML development is difficult. Their study [4] summarized the reasons as the low extensibility of an ML model and the non-obvious interaction between ML models. In contrast, we found ML development places comparable emphasis on low coupling in components as non-ML development [S5]. Both studies agree that the rapid iterations exist in the detailed design of ML systems [3 S7]. ~ Construction and Tools. Both studies agree that code reuse in ML development is challenging due to varying application context and input data. Despite the challenge of code reuse in ML development, we found that code reuse happens in ML development as frequently as in non-ML development [S9]. ~ Process and Management. Both studies agree that management of ML development is challenging due to the involvement of data, and that the availability of data usually limits the capability of ML systems [33 S21]. ~ Configuration Management. Both studies agree that data versioning is required in ML development. Despite the necessity of data versioning, we found that current configuration management activities in ML development still focuses on code versioning [S20]. In addition to data versioning, the earlier study [4] suggested to keep track of how data is gathered and processed. As discussed above, our study confirms some of the findings reported in Amershi et al.'s work. Being different from Amershi et al.'s work, since our study followed the SWEBOK and considered work characteristics from applied psychology domain in our interviews, we recognized the differences between ML and non-ML development in other aspects, e.g., requirement gathering, job complexity, problem solving process, and task identity. Moreover, we collected perceptions from broader population groups, e.g., involving open source practitioners and professionals from various software industries.
Modified on	22/06/2022 09:39

Name	How does Machine Learning Change Software Development Practices~
Number of Coding References	12
Number of Codes Coding	2
Coverage	6.72%
Folder Location	Codes\\Snowballing
Parent Name	Codes\\Snowballing\\Snowballing
Created on	21/06/2022 14:00
Coded Text	Some prior work provides prescriptive practices for the development of machine learning systems (e.g., [29]). Some discuss the realistic challenges and best practices in the industry, e.g., machine learning model management at Amazon [33], and best practices of machine learning engineering at Google [40]. Some investigated machine learning related questions on Stack Overflow [38]. These works are based on the experience of the authors and largely do not contextualize machine learning development as a special type of software engineering. In contrast, our findings are based on empirical observations that explicitly focus on the differences between ML and non-ML development. Like our work, several researchers have conducted empirical studies of software engineering for data science. Some focus on how data scientists work inside a company via interviews to identify pain points from a general tooling perspective [13], and explore challenges and barriers for adopting visual analytic tools [20]. Other focus on characterizing professional roles and practices regarding data science: Harris et al. [17] surveyed more than 250 data science practitioners to categorize data science practitioners and identify their skill sets. Kim et al. interviewed sixteen data scientists at Microsoft to identify five working styles [21], and supplement Harris et al.'s survey with tool usage, challenges, best practices and time spent on different activities [22]. In contrast to this prior work, our paper studies broad differences between ML and non-ML development. Most similar to our study is an empirical investigation of integrating AI capabilities into software and services and best practices from Microsoft teams [4]. From their proposed ML workflow, they identified 11 challenges, including 1) data availability, collection, cleaning, and management, 2) education and training, 3) hardware resources, 4) endto-end pipeline support, 5) collaboration and working culture, 6) specification, 7) integrating AI into larger systems, 8) guidance and mentoring, 9) AI tools, 10) scale, and 11) model evolution, evaluation, and deployment. The identified challenges emerge across different software development practices. Our findings differ from theirs in a number of areas:
Modified on	22/06/2022 09:39

Name	How Teams Communicate about the Quality of ML Models~ A Case Study at an International Technology Company
Number of Coding References	15
Number of Codes Coding	2
Coverage	6.88%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	Our interviews and survey focused on observing what quality means to team members working on ML models and how the quality of ML models is communicated within big teams holding different roles. Through our observations, we identified challenges that team members face in perceiving the quality ofmodels and how they tackled. Some teams overcome the challenges in communicating ML models by having a middleman, usually a PM or SE, to communicate model quality aspects between model developers and other team members who are non-ML experts (e.g., UX designers, legal, sales, etc.). This causes some information to be lost in translation. As some of our participants reported, involving ML developers in meetings with other team members has proven to be more efficient and fruitful to the discussion and overall success of a product. In this section, we start by synthesizing and discussing the main challenges in communicating the quality ofML models to a wide audience. After that, we discuss best practices in communicating quality through five lenses (who and with whom, what, form, and goal). Throughout the discussion, we use the word stakeholders to refer to internal employees who are part of the same software organization but are from different teams, such as UX, legal, sales, etc.
Modified on	07/02/2022 16:29

Name	How Teams Communicate about the Quality of ML Models~ A Case Study at an International Technology Company
Number of Coding References	15
Number of Codes Coding	2
Coverage	6.88%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	.3.3 Workflows. Sharing the process that led to the solution is also crucial in promoting confidence in the quality of the solution. Some details about ML models seem to be important to the audience, such as the reason for choosing a specific class ofmodels or how the parameters were tuned and so on. For instance, a data scientist mentioned what to discuss around ML models: “Problem being solved and why we picked the ML model? Your thought process behind picking the model”. A software engineer also suggested a more engaging discussion in which the results and the journey to the solution matter to non-ML developers “show how you’ve arrived at the results and how tweaking certain parameters can impact the model output
Modified on	07/02/2022 16:30

Name	How Teams Communicate about the Quality of ML Models~ A Case Study at an International Technology Company
Number of Coding References	15
Number of Codes Coding	2
Coverage	6.88%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	5.1.2 Struggle in problem formulation. Another emerging pattern in our observations from the survey and interviews is due to the lack of knowledge regarding the capabilities and potentials of ML. The problem is divided into two groups: One occurs at the beginning of the ML development workflow due to issues in problem formulation as a result of not knowing what to expect from ML, and the other occurs during validation . For example, a data scientist clearly described the challenge he faces regarding his audience expectations: “Many non-technical people assume that ML can do anything and that it will tell them what they need to know. But ML is only good when you have 1) good questions, and 2) good data (with good labels!).
Modified on	07/02/2022 16:29

Name	How Teams Communicate about the Quality of ML Models~ A Case Study at an International Technology Company
Number of Coding References	15
Number of Codes Coding	2
Coverage	6.88%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45

Coded Text	5.1.3 The need for education before communication . Not only is the lack ofML knowledge challenging at the start of the workflow during problem formulation, but it also requires extra effort from the model developer to educate and raise awareness in their audience. As one data scientist explained, “We usually need to put in extra work to achieve the baseline level ofknowledge before we can discuss the actual mode/feature”. Another data scientist mentioned how this mismatch in the level of knowledge forces her to hide some details to avoid misunderstanding. In her words, the challenge is “[h]ow to convert technical terminology into daily words. Sometimes, in order to explain clearly, I have to ignore some exceptions/details to avoid confusion”.
Modified on	07/02/2022 16:29

Name	How Teams Communicate about the Quality of ML Models~ A Case Study at an International Technology Company
Number of Coding References	15
Number of Codes Coding	2
Coverage	6.88%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	5.1.4 Conflicting documentation and standards. . One of the interesting patterns in the results is how a lack of standards and documentation that are universal across team members is a hurdle in the communication process because no common language is available. One data scientist explained, “The biggest challenge I’ve run into on our team is folks having a common language to use when discussing models in terms offunctionality.” He then followed up with how it is less of a challenge due to his team process “Fortunately our APIconsumes the ONNXmodel format which has helped improve these conversations.”. Another data scientist mentioned, “There is no standard checklist for checking the model’s accuracy. When you go and talk to different data scientists, even seniors they have their own mind map, so going through the model with different people is different”.
Modified on	07/02/2022 16:29

Name	How Teams Communicate about the Quality of ML Models~ A Case Study at an International Technology Company
Number of Coding References	15
Number of Codes Coding	2
Coverage	6.88%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	5.1.5 Failure to see ‘The Big Picture’. Another challenge around communicating models arises from the fact that an ML model is part of an ecosystem in which conversations are bidirectional. The challenge arises out of the fact that some data scientists are not aware of the deployment and engineering practices. For example one PM mentioned: “Modelers don’t always know what it takes to take a model to production”. An SE confirmed this by stating: “Scoping work can be difficult; it’s not always easy to know how many models we’ll need to try and how long we’ll need to iterate”.
Modified on	07/02/2022 16:30

Name	How Teams Communicate about the Quality of ML Models~ A Case Study at an International Technology Company
Number of Coding References	15
Number of Codes Coding	2
Coverage	6.88%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	5.1.6 Struggle to explain and understand common model metrics in context. Variations in metrics and their subjectivity raises a challenge in discussions within teams. For example one a data scientist mentioned, “Most people do not understand that it’s difficult to compare metrics across models - sometimes an AUC of0.9 is good enough, sometimes it is not”. Another senior data scientist mentioned something along those lines: “Sometimes a slightly low accuracy does not ruin the UX, it could be ignored or maybe it does not have impact. It is hard to convey that to our stakeholders”. Another issue related to communicating meaningful metrics is how they are being presented and discussed, because some data scientists choose to bias their presentation toward those metrics that work. As a data scientist expressed: “Not everyone knows what metrics really mean. Some people just blindly follow the metrics.
Modified on	07/02/2022 16:30

Name	How Teams Communicate about the Quality of ML Models~ A Case Study at an International Technology Company
Number of Coding References	15
Number of Codes Coding	2
Coverage	6.88%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	5.1.7 Struggle to explain and interpret model behavior. There is often a difficulty in “explaining” a model’s behavior, (i.e., When the model works well, why? When the model is not working as
Modified on	07/02/2022 16:30

Name	How Teams Communicate about the Quality of ML Models~ A Case Study at an International Technology Company
Number of Coding References	15
Number of Codes Coding	2
Coverage	6.88%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	5.1.8 Intimidation by the perceived complexity, or is it too much trust? An obvious pattern in our interviews with data scientists is the issue of trust in the sense that a model consumer often trusts that the model builder knows their job; therefore, there is not as much discussion around the model as there might need to be. As one UX designer mentioned, “I have no formal education related to ML models. So basically, I just rely on the adhoc understanding ofthe model developer”. A PM also mentioned that, “stakeholders don’t care about model metrics they just want to know ifthe model is good or not, Usually I just share one number (aggregate measure like accuracy) and no one ever asked for details. They just want to see data related to the requirements”.
Modified on	07/02/2022 16:30

Name	How Teams Communicate about the Quality of ML Models~ A Case Study at an International Technology Company
Number of Coding References	15
Number of Codes Coding	2
Coverage	6.88%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	5.4.1 Presentation tools. Looking at the results of Table 3, we see some practices and tools that are currently used to present models’ quality to elicit launching decisions or any of the other goals shown in Table 4. PowerPoint and OneNote were popular in communicating models, which presents a potential design opportunity.
Modified on	07/02/2022 16:30

Name	How Teams Communicate about the Quality of ML Models~ A Case Study at an International Technology Company
Number of Coding References	15
Number of Codes Coding	2
Coverage	6.88%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	5.4.3 Quality shines with context and visuals. In presenting models, having the right form of presentation that relates the model to a context is a helpful exercise. As one PM mentioned regarding his approach when data scientists are presenting their models to him: “How explicable is this model? it helps to have a nice story around the model, what’s the story and how will it perform in the wild? the person has to [tell] a story.”.
Modified on	07/02/2022 16:31

Name	How Teams Communicate about the Quality of ML Models~ A Case Study at an International Technology Company
Number of Coding References	15
Number of Codes Coding	2
Coverage	6.88%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	Challenges in Communicating ML Models In this section, we discuss the main challenges and best practices around communicating ML models within teams. In the surveys and interviews, participants were asked two questions about challenges: One was aimed at ML experts who develop the models, asking them directly about challenges they face when discussing ML models with non-experts. The other question was directed toward non-ML developers working on ML projects and the challenges they face within their teams when those ML models are discussed. Through another affinity diagram exercise, we iterated over the reported open-ended questions, we identified emerging patterns. We share the following challenges and best practices to inspire future work and design direction for new solutions. 5.1.1 Mismatch between the discussion ofuser-need-driven and model-driven performance. One of the common problems that has been observed in a variety of ways is that when discussing ML features in a system, data scientists tend to be more model driven as opposed to user-need driven (e.g. they discuss the performance of a model in terms of accuracy or recall, but not in terms of the overall task). This mostly affects those ML model developers who are building customer facing features ofmodels. For example, a software engineer mentioned the challenge he faces when discussing the model as follows: “Our team focuses on building tools for 3rd party developers, so our work is generally focused on models as collections/generally. It’s often hard to discuss the models when we’re focused on how they perform on a standard task (ImageNet, CoCo) but customers care about their specific problem space”. Another participant who is a UX designer working on customer facing ML based projects explains the situation with the following metaphor:“Trying to navigate the ’cart before the horse’ discussion because it can be a sensitive topic. Cart before the horse = conducting research, creating and training models before having a sense ofwhat real customer problem you’re trying to solve. Can lead to sometimes feeling like we’re shoe-horning ML into products where they *might* not be needed”. Another UX designer who has been working on AI based projects for more than 5 years indicated that the fact that that priorities for each practitioner in AI based teams is different and often causes challenges in communicating efficiently and described as a “build model
Modified on	07/02/2022 16:29

Name	How Teams Communicate about the Quality of ML Models~ A Case Study at an International Technology Company
Number of Coding References	15
Number of Codes Coding	2
Coverage	6.88%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	Our results were elicited from observing the communication process through our interviews and surveys. Our questions were inspired by Lasswell's model of communication [25] to provide a comprehensive analysis of the communication process. During our interviews, participants shared artifacts they used to discuss MLmodels (e.g., emails sent to other team members, presentations, projecting their work, etc.), which helped us in the elicitation process. However, a longer ethnographic shadowing would be beneficial in adding to the challenges and best practices we have identified. Our recruited interview and survey participants were all employees of a software company with large multidisciplinary teams. Most of our participants had some exposure to ML because the company is actively advertising events and courses to learn ML. Therefore, some of our observations might not generalize to less experienced teams. Furthermore, the total number of data scientists and software engineers was higher than the numbers of any other user group reflected in the software company population. We tried to randomize the sample by randomizing the questions regarding communication to different roles.
Modified on	07/02/2022 16:31

Name	How Teams Communicate about the Quality of ML Models~ A Case Study at an International Technology Company
Number of Coding References	15
Number of Codes Coding	2
Coverage	6.88%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	Prior studies have explored how ML is affecting development team roles beyond data scientists, including user experience designers, program managers, developers and operations engineers. However, there has been little investigation ofhowteam members in different roles on the team communicate about ML, in particular about the quality ofmodels. We use the general term quality to look beyond technical issues ofmodel evaluation, such as accuracy and overfitting, to any issue affecting whether a model is suitable for use, including ethical, engineering, operations, and legal considerations. What challenges do teams face in discussing the quality ofML models? What work practices mitigate those challenges? To
Modified on	07/02/2022 16:27

Name	How Teams Communicate about the Quality of ML Models~ A Case Study at an International Technology Company
Number of Coding References	15
Number of Codes Coding	2
Coverage	6.88%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	This emergence of ML also means that software teams increasingly need to communicate about ML models and their quality. Here, we use the general term quality to encompass not only technical issues about ML model evaluation such as accuracy, but also any issue affecting whether a model is suitable for use, including ethical, engineering, operations, and legal considerations. Because ML is involved in many aspects of software development—from UX, to engineering and operations, to management—this communication spans many roles on the team [7, 37]. Prior research explored how data scientists ensure high confidence in their analysis results [24, 36, 37, 57]. However, concerns about ML quality are not limited to issues related to the role of data scientists. Other roles on the team, including UX designers, program managers (PMs), developers, operations engineers, and product managers, also need to communicate about ML models and their quality, to support coordination and decision making. What are the challenges software teams face when communicating about ML models and their quality? What practices and tools can be introduced to mitigate these challenges?
Modified on	07/02/2022 16:27

Name	HyperNOMAD~ Hyperparameter Optimization of Deep Neural Networks Using Mesh Adaptive Direct Search
Number of Coding References	11
Number of Codes Coding	2
Coverage	7.66%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	03/02/2022 10:04
Coded Text	
Modified on	07/02/2022 16:20

Name	HyperNOMAD~ Hyperparameter Optimization of Deep Neural Networks Using Mesh Adaptive Direct Search
Number of Coding References	11
Number of Codes Coding	2
Coverage	7.66%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	03/02/2022 10:04
Coded Text	
Modified on	07/02/2022 16:21

Name	HyperNOMAD~ Hyperparameter Optimization of Deep Neural Networks Using Mesh Adaptive Direct Search
Number of Coding References	11
Number of Codes Coding	2
Coverage	7.66%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	03/02/2022 10:04

Coded Text	4.1 Overview of NOMAD TheNOMADsoftware [37]isa C++ implementation oftheMADS algorithm [7, 9], which is a direct search method that generates, at each iteration k, a set ofpoints on the meshMk = {x + diag(δk)z : x ∈ Vk, z ∈ Zn },where Vk contains the points that were previously evaluated (including the current iterate xk)and δk ∈ Rn is the mesh size vector. Each iteration of MADS is divided into two steps: the search and the poll.The search phase is optional and can contain different strategies to explore a wider space in order to generate a finite number of possible mesh candidates. This step can be based on surrogate functions, Latin hypercube sampling, and so forth [4, 11]. The poll, on the other hand, is strictly defined since the convergence theory ofMADS relies entirely on this phase. In the poll step, the algorithm generates directions around the current iterate xk to search for candidates locally in a region centered around xk and of radius, in each dimension, of Δk ∈ Rn, which is called the poll size vector.The setof candidates in this step defines the poll set Pk. IfMADS finds a better point than the incumbent, then the iteration is declared a success, and the mesh and poll sizes are increased. However, if the iteration fails, then both parameters are reduced so thatδk ≤ Δk is maintained. This relation ensures that the set ofsearch directions becomes dense in the unit sphere asymptotically. In addition, NOMAD can handle categorical variables by adding a step in the basic MADS algorithm. A variable is categorical when it can take a finite number of nominal or numerical values that express a qualitative property that assign the variable to a class (or category). The algorithm relies on an ad hoc neighborhood structure, provided in practice by the user as a list ofneighbors for any given point. The poll step ofMADS is augmented with the socalled extended poll that links the current iterate xk with the independent search spaces where the neighbors can be found. The first neighbor that improves the objective function is chosen and the optimization carries on in the corresponding search space.
Modified on	07/02/2022 16:23

Name	HyperNOMAD~ Hyperparameter Optimization of Deep Neural Networks Using Mesh Adaptive Direct Search
Number of Coding References	11
Number of Codes Coding	2
Coverage	7.66%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	03/02/2022 10:04
Coded Text	Bayesian optimization (BO) can be seen as a subclass of DFO methods and, as such, can be used to solve the HPO problem. The BO methods use information collected during previous assessments to diagnose the search space and predict which areas to explore first. Among them, Gaussian processes (GPs) are models that seek to explain the collected observations that supposedly come from a stochastic function. GPs are a generalization ofmulti-variate Gaussian distributions, defined by a mean and a covariance function. GPs are popular models for optimizing the hyperparameters ofneural networks [56, 60]. However, the disadvantage ofGPs is that they do not fit well to categorical features, and their performance depends on the choice of the kernel function that defines them. Tree-structured Parzen Estimator (TPE) is also a Bayesian method that can be used as a model instead of a GP. After a certain number of evaluations
Modified on	07/02/2022 16:21

Name	HyperNOMAD~ Hyperparameter Optimization of Deep Neural Networks Using Mesh Adaptive Direct Search
Number of Coding References	11
Number of Codes Coding	2
Coverage	7.66%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	03/02/2022 10:04

Coded Text	Genetic algorithms are evolutionary heuristics that are also used for the HPO problem. Inspired by biology, a genetic algorithm generates an initial population, i.e., a set ofconfigurations. Then, it combines the best parents to create a new generation of children. It also introduces random mutations to ensure a certain diversity in the population. These heuristics are therefore adaptive, thus exploring the space more wisely even ifsome randomness remains in the process. These algorithms are often used to optimize hyperparameters [26, 57, 63]. In [45], a method based on particle swarm optimization is able to provide networks with higher performance than those defined by experts in less time than what would have required a grid search or a completely random search. Another approach using the evolutionary algorithm CMA-ES [46] was proposed with satisfactory results.
Modified on	07/02/2022 16:21

Name	HyperNOMAD~ Hyperparameter Optimization of Deep Neural Networks Using Mesh Adaptive Direct Search
Number of Coding References	11
Number of Codes Coding	2
Coverage	7.66%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	03/02/2022 10:04
Coded Text	One of the first scientific approaches used to tackle the HPO problem of neural networks is grid search. This method consists of discretizing the hypercube defined by the range of each
Modified on	07/02/2022 16:21

Name	HyperNOMAD~ Hyperparameter Optimization of Deep Neural Networks Using Mesh Adaptive Direct Search
Number of Coding References	11
Number of Codes Coding	2
Coverage	7.66%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	03/02/2022 10:04
Coded Text	The HyperNOMAD package is available onGitHub.1 It contains a series ofPython modules that act as a blackbox, which takes a set ofhyperparameters described in Section 3 as inputs and constructs the corresponding network that is trained and tested before returning the test accuracy as the output. This blackbox uses the PyTorch package [48] for its simplicity. HyperNOMAD also contains an interface that runs the optimization ofthe blackbox using the NOMAD software [37] described in the rest of this section. The basic usage of HyperNOMAD is described in Appendix A.
Modified on	07/02/2022 16:22

Name	HyperNOMAD~ Hyperparameter Optimization of Deep Neural Networks Using Mesh Adaptive Direct Search
Number of Coding References	11
Number of Codes Coding	2
Coverage	7.66%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	03/02/2022 10:04
Coded Text	The hyperparameters that define a deep neural network can be separated into two categories: the ones that define the architecture ofthe network and the ones that affect the optimization process of the training phase. Tuning the hyperparameters ofthe first category alone has led to a separate field of research called Neural Architecture Search (NAS) [25] that allowed achievement of state-ofthe-art performance [53, 65] on some benchmark problems, although at a massive computational cost of800 GPUs for a fewweeks. Typically, one would perform an NAS first and then start tuning the other hyperparameters with the optimized architecture. However, Zela et al. [64] argue that this separation is not optimal since the two aspects are not entirely independent from one another.
Modified on	07/02/2022 16:21

Name	HyperNOMAD~ Hyperparameter Optimization of Deep Neural Networks Using Mesh Adaptive Direct Search
Number of Coding References	11
Number of Codes Coding	2
Coverage	7.66%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	03/02/2022 10:04
Coded Text	The performance ofdeep neural networks is highly sensitive to the choice ofthe hyperparameters that define the structure of the network and the learning process. When facing a new application, tuning a deep neural network is a tedious and time-consuming process that is often described as a “dark art.” This explains the necessity of automating the calibration of these hyperparameters. Derivative-free optimization is a field that develops methods designed to optimize time-consuming functions without relying on derivatives. This work introduces the HyperNOMAD package, an extension of the NOMAD software that applies the MADS algorithm [7] to simultaneously tune the hyperparameters responsible for both the architecture and the learning process of a deep neural network (DNN).
Modified on	07/02/2022 16:19

Name	HyperNOMAD~ Hyperparameter Optimization of Deep Neural Networks Using Mesh Adaptive Direct Search
Number of Coding References	11
Number of Codes Coding	2
Coverage	7.66%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	03/02/2022 10:04
Coded Text	The selected neighborhood structure in HyperNOMAD relies on blocks of categorical variables with their associated variables. The following subsections describe this structure. 4.2.1 Blocks ofHyperparameters. HyperNOMAD splits the hyperparameters (HPs) defined in Section 3.1 into different blocks: one for the convolution layers, the fully connected layers, and the optimizer and one for each of the other HPs. A block is an implemented structure that stores a list ofvalues, each one starting with a header and followed by the associated variables, when applicable, that are gathered into groups. For example, consider a CNN with two convolutional layers, each one defined with the number of output channels, the kernel size, the stride, the padding, and whether a pooling is applied or not as stated in Table 2. Then consider the values (16, 5, 1, 1, 0) and (7, 3, 1, 1, 1). Each set ofvalues corresponds to a group ofvariables that describes one convolutional layer and both groups are part of the convolution block. The header of the convolution block is the categorical variable that represents the number ofconvolutional layers (n1) that the CNN contains as showninFigure 5 (top).
Modified on	07/02/2022 16:23

Name	HyperNOMAD~ Hyperparameter Optimization of Deep Neural Networks Using Mesh Adaptive Direct Search
Number of Coding References	11
Number of Codes Coding	2
Coverage	7.66%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	03/02/2022 10:18
Coded Text	However, the performance of a neural network is strongly linked to its structure and to the values of the parameters of the optimization algorithm used to minimize the error between the predictions of the network and the data during its training. The choices of the neural network hyperparameters can greatly affect its ability to learn from the training data and to generalize with new data. The algorithmic hyperparameters of the optimizer must be chosen a priori and cannot be modified during optimization. Hence, to obtain a neural network, it is necessary to fix several hyperparameters of various types: real, integer, and categorical. A variable is categorical when it describes a class, or category, without a relation of order between these categories. The search for an optimal configuration is a very slow process that, along with the training, takes up the majority of the time when developing a network for a new application. It is a relatively new problem that is often solved randomly or empirically. Derivative-free optimization (DFO) [8, 21] is the field that aims to solve optimization problems where derivatives are unavailable, although they might exist. This is the case, for example, when the objective and/or constraint functions are non-differentiable, noisy, or expensive to evaluate. In addition, the evaluation in some points may fail, especially if the values of the objective and/or constraints are the outputs of a simulation or an experience. Blackbox optimization (BBO) is a subfield of DFO where the derivatives do not exist and the problem is modeled as a blackbox. This term refers to the fact that the computing process behind the output values is unknown. The general DFO problem is described as min $x \in \Omega$ $f(x)$, where f is the objective function to minimize over the domain Ω. There are two main classes ofDFO methods: model-based and direct search methods. The first uses the value of the objective and/or the constraints at some already evaluated points to build a model able to guide the optimization by relying on the predictions of the model. For example, this class includes methods based on trust regions [21, Chapter 10] or interpolation models [52]. This differentiates them from direct search methods [31] that adopt a more straightforward strategy to optimize the blackbox. At each iteration, direct search methods generate a set of trial points that are compared to the “best solution” available. For example, the GPS algorithm [59]definesamesh on the search space and determines the next point to evaluate by choosing a search direction. DFO algorithms usually include a proof ofconvergence that ensures a good-quality solution under certain hypotheses on the objective function. BBO algorithms extend beyond this scope by including heuristics such as evolutionary algorithms, sampling methods, and so on. In [5, 10], the authors explain how a hyperparameter optimization (HPO) problem can be seen as a blackbox optimization problem. Indeed, the HPO problem is equivalent to a blackbox that takes the hyperparameters of a given algorithm and returns some measure of performance defined in advance such as the time to solution, the value of the best point found, or the number of solved problems. In the case of neural networks, the blackbox can return the accuracy on the test dataset as a measure of performance. With this formulation, DFO techniques can be applied to solve the original HPO problem.
Modified on	07/02/2022 16:20

Name	Inspector gadget~ a data programming-based labeling system for industrial images
Number of Coding References	12
Number of Codes Coding	3
Coverage	11.04%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	
Modified on	07/02/2022 15:45

Name	Inspector gadget~ a data programming-based labeling system for industrial images
Number of Coding References	12
Number of Codes Coding	3
Coverage	11.04%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	
Modified on	07/02/2022 15:45

Name	Inspector gadget~ a data programming-based labeling system for industrial images
Number of Coding References	12
Number of Codes Coding	3
Coverage	11.04%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	
Modified on	07/02/2022 15:46

Name	Inspector gadget~ a data programming-based labeling system for industrial images
Number of Coding References	12
Number of Codes Coding	3
Coverage	11.04%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	A conventional solution is to collect enough labels manually and train say a convolutional neural network on the training data. However, fully relying on crowdsourcing for image labeling can be too expensive. In our application, we have heard of domain experts demanding six-figure salaries, which makes it infeasible to simply ask them to label images. In addition, relying on general crowdsourcing platforms like Amazon Mechanical Turk may not guarantee high-enough labeling quality.
Modified on	07/02/2022 15:44

Name	Inspector gadget~ a data programming-based labeling system for industrial images
Number of Coding References	12
Number of Codes Coding	3
Coverage	11.04%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	In this work, we expand the horizon of data programming by directly applying it to images without this conversion, which is a common scenario for industrial applications. We propose Inspector Gadget, an image labeling system that combines crowdsourcing, data augmentation, and data programming to produce weak labels at scale for image classification. We perform experiments on real industrial image datasets and show that Inspector Gadget obtains better performance than other weak-labeling techniques: Snuba, GOGGLES, and self-learning baselines using convolutional neural networks (CNNs) without pre-training.
Modified on	07/02/2022 15:43

Name	Inspector gadget~ a data programming-based labeling system for industrial images
Number of Coding References	12
Number of Codes Coding	3
Coverage	11.04%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43

Coded Text	So far, data programming has been shown to be effective in finding various relationships in text and structured data [29]. Data programming has also been successfully applied to images where they are usually converted to structured data beforehand [41, 43]. However, this conversion limits the applicability of data programming. As an alternative approach, GOGGLES [9] demonstrates that, on images, automatic approaches using pre-trained models may be more effective. Here the idea is to extract semantic prototypes of images using the pre-trained model and then cluster and label the images using the prototypes. However, GOGGLES also has limitations (see Section 6.2), and it is not clear if it is the only solution for generating training data for image classification.
Modified on	07/02/2022 15:45

Name	Inspector gadget~ a data programming-based labeling system for industrial images
Number of Coding References	12
Number of Codes Coding	3
Coverage	11.04%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	07/02/2022 15:43
Coded Text	We thus propose Inspector Gadget, which opens up a new class of problems for data programming by enabling direct image labeling at scale without the need to convert to structured data using a combination of crowdsourcing, data augmentation, and data programming techniques. Inspector Gadget provides a crowdsourcing workflow where workers identify patterns that indicate defects. Here we make the tasks easy enough for non-experts to contribute. These patterns are augmented using general adversarial networks (GANs) [13] and policies [7]. Each pattern effectively becomes a labeling function by being matched with other images. The similarities are then used as features to train a multi-layer perceptron (MLP), which generates weak labels. In our experiments, Inspector Gadget performs better overall than state-of-the-art methods: Snuba, GOGGLES, and self-learning baselines that use CNNs (VGG-19 [36] and MobileNetV2 [33]) without pre-training. We release our code as a community resource [1]. In the rest of the paper, we present the following: • The architecture of Inspector Gadget (Section 2). • The component details of Inspector Gadget: • Crowdsourcing workflow for helping workers identify patterns (Section 3). • Pattern augmenter for expanding the patterns using GANs and policies (Section 4). • Feature generator and labeler for generating similarity features and producing weak labels (Section 5). • Experimental results where Inspector Gadget outperforms other image labeling techniques – Snuba, GOGGLES, and self-learning baselines using CNNs – where there are few or no labels to start with (Section 6).
Modified on	07/02/2022 15:45

Name	Inspector gadget~ a data programming-based labeling system for industrial images
Number of Coding References	12
Number of Codes Coding	3
Coverage	11.04%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	31/01/2022 14:44
Coded Text	Among the possible methods for data labeling (see an extensive survey [32]), weak supervision is an important branch of research where the idea is to semi-automatically generate labels that are not perfect like manual ones. Thus, these generated labels are called weak labels, but they have reasonable quality where the quantity compensates for the quality. Data programming [30] is a representative weak supervision technique ofemploying humans to develop labeling functions (LFs) that individually perform labeling (e.g., identify a person riding a bike), perhaps not accurately. However, the combination of inaccurate LFs into a generative model results in probabilistic labels with reasonable quality. These weak labels can then be used to train an end discriminative model.
Modified on	07/02/2022 15:44

Name	Inspector gadget~ a data programming-based labeling system for industrial images
Number of Coding References	12
Number of Codes Coding	3
Coverage	11.04%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	31/01/2022 14:44
Coded Text	As machine learning for images becomes democratized in the Software 2.0 era, one of the serious bottlenecks is securing enough labeled data for training. This problem is especially critical in a manufacturing setting where smart factories rely on machine learning for product quality control by analyzing industrial images. Such images are typically large and may only need to be partially analyzed where only a small portion is problematic (e.g., identifying defects on a surface). Since manual labeling these images is expensive, weak supervision is an attractive alternative where the idea is to generate weak labels that are not perfect, but can be produced at scale. Data programming is a recent paradigm in this category where it uses human knowledge in the form of labeling functions and combines them into a generative model. Data programming has been successful in applications based on text or structured data and can also be applied to images usually if one can find a way to convert them into structured data. In
Modified on	07/02/2022 15:43

Name	Inspector gadget~ a data programming-based labeling system for industrial images
Number of Coding References	12
Number of Codes Coding	3
Coverage	11.04%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	31/01/2022 14:44
Coded Text	We focus on the problem of scalable labeling for classification where large images are partially analyzed, and there are few or no labels to start with. Although many companies face this problem, it has not been studied enough. Based on a collaboration with a large manufacturing company, we provide the following running example. Suppose there is a smart factory application where product images are analyzed for quality control (Figure 1). These images taken from industrial cameras usually have high-resolution. The goal is to look at each image and tell if there are certain defects (e.g., identify scratches, bubbles, and stampings). For convenience, we hereafter use the term defect to mean a part of an image of interest.
Modified on	07/02/2022 15:44

Name	Inspector gadget~ a data programming-based labeling system for industrial images
Number of Coding References	12
Number of Codes Coding	3
Coverage	11.04%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56
Coded Text	constructing LFs for entity resolution on structured data. Adversarial data programming [28] proposes a GAN-based framework for labeling with LF results and claims to be better than Snorkel-based approaches. In comparison, Inspector Gadget solves the different problem of partially analyzing large images. Automatic Image labeling. There is a variety ofgeneral automatic image labeling techniques. Data augmentation [35] is a general method to generate new labeled images. Generative adversarial networks (GANs) [13] have been proposed to generate fake, but realistic images based on existing images. Policies [7] were proposed to apply custom transformations on images as long as they remain realistic. Most of the existing work operate on the entire images. In comparison, Inspector Gadget is efficient because it only needs to augment patterns, which are much smaller than the images. Label propagation techniques [5] organize images into a graph based on their similarities and then propagates existing labels of images to their most similar ones. In comparison, Inspector Gadget is designed for images where only a small part of them are of interest while the main part may be nearly identical to other images, so we cannot utilize the similar method. There are also application-specific defect detection methods [17, 18, 37], some of which are designed for the datasets we used. In comparison, Inspector Gadget provides a general framework for image labeling. Recently, GOGGLES [9]
Modified on	07/02/2022 15:47

Name	Inspector gadget~ a data programming-based labeling system for industrial images
Number of Coding References	12
Number of Codes Coding	3
Coverage	11.04%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56
Coded Text	Crowdsourcing for machine learning and databases. Using hu- mans in advanced analytics is increasingly becoming mainstream [45] where the tasks include query processing [24], entity matching [12], active learning [26], data labeling [15], and feature engineering [6]. Inspector Gadget provides a new use case where the crowd identifies patterns. Data Programming. Data programming [30] is a recent para- digm where workers program labeling functions (LFs), which are used to generate weak labels at scale. Snorkel [4, 29] is a seminal system that demonstrates the practicality of data programming, and Snuba [42] extends it by automatically constructing LFs using primitives. In comparison, Inspector Gadget does not assume any accuracy guarantees on the feature generation function and directly labels images without converting them to structured data. Several systems have studied the problem of automating labeling function construction. CrowdGame [22] proposes a method for
Modified on	07/02/2022 15:47

Name	IntegratedML~ Every SQL Developer is a Data Scientist
Number of Coding References	8
Number of Codes Coding	4
Coverage	17.60%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	Automated Machine Learning (AutoML) [4] has surfaced as a new meta-technique that uses heuristics as well as iteration over this set of options in search of the combination yielding the most accurate model. AutoML software includes both packages built on top of existing ML frameworks, such as auto-sklearn and AutoKeras, as well as tools and services with sophisticated user interfaces such as DataRobot, where users simply upload a CSV file, indicate the column to be predicted, and have the service do the rest. AutoML clearly softens the learning curve, but the first category of tools is typically still based on the underlying frameworks' ML concepts, thus requiring at least some ML skillset; and the second one is targeting a much less technical persona than the application developers writing enterprise software today. Also, AutoML does not specifically address the deployment issue
Modified on	06/06/2022 10:28

Name	IntegratedML~ Every SQL Developer is a Data Scientist
Number of Coding References	8
Number of Codes Coding	4
Coverage	17.60%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Software Mantainability Aspects
Created on	04/02/2022 14:30
Coded Text	A Provider implements a set of APIs for consuming relational data from the embedding SQL database to produce model training and predictions. This common interface allows the differing functional capabilities and performance characteristics of diverse machine learning frameworks to be exposed with a uniform SQL syntax. Providers can be implemented in any language supported by the database runtime. IntegratedML currently ships with three providers: a Python-based AutoML engine developed in-house leveraging a variety of well-known packages including scikit-learn, XGBoost and TensorFlow a wrapper around the open source H2O [13] platform an interface to DataRobot [14], supporting both their cloud-based and on-premise solutions.
Modified on	06/06/2022 10:40

Name	IntegratedML~ Every SQL Developer is a Data Scientist
Number of Coding References	8
Number of Codes Coding	4
Coverage	17.60%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Software Mantainability Aspects
Created on	04/02/2022 14:30
Coded Text	The adoption of machine learning in business software is slowed down by a shortage of data science talent and challenges around efficient operationalization of machine learning models. We present IntegratedML, an embedded database capability for machine learning. This paper describes how IntegratedML provides developers with access to state-of-the-art machine learning platforms using intuitive SQL syntax. Its embedded feature extraction and algorithm selection enable fully automated model building, while model inferencing is exposed through a simple scalar function. The novelty of IntegratedML is in the deep integration into the embedding relational engine, which hides pipeline complexity from the user and guarantees high efficiencies, both at train and inference time.
Modified on	07/02/2022 15:38

Name	IntegratedML~ Every SQL Developer is a Data Scientist
Number of Coding References	8
Number of Codes Coding	4
Coverage	17.60%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Software Mantainability Aspects
Created on	04/02/2022 14:30
Coded Text	The relational database is where the majority of enterprise applications still find their data. Making model training and inferencing transparently available within such an environment simplifies embedding ML in applications for a broad audience. Offering SQL access to ML concepts is not new by itself, but the integration offered by IntegratedML is much deeper than with other solutions, which usually stick to a “containerized” execution model for code authored by data scientists, with the rigid semantics of stored procedure and table-valued functions. Deeper integration also enables higher efficiencies, as also demonstrated by [11]. Beyond model training and inference, we believe other ML tasks will also benefit from the database’s knowledge of data and access to it. Specifically, model explainability and (deployed) model performance can benefit from data distribution and query statistics naturally available in the DBMS. These themes are getting more and more attention in data science research, but tooling for these is only just emerging and rarely integrated in an end-to-end platform. IntegratedML does not replace data scientists, but is all about productivity: SQL developers can get started with ML quickly and address a sizeable portion of the problems that would otherwise pile up for what is typically an understaffed
Modified on	07/02/2022 15:40

Name	IntegratedML~ Every SQL Developer is a Data Scientist
Number of Coding References	8
Number of Codes Coding	4
Coverage	17.60%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Software Mantainability Aspects
Created on	04/02/2022 14:30
Coded Text	These frameworks, however, are typically focused on algorithm efficiency and less so on usability, leading to a complicated set of stored procedures or nonSQL APIs that still require specific ML knowledge. Many vendors, including Microsoft [9] and Oracle [10], offer deep integration of popular programming environments for ML such as R and Python, embedding procedural syntax within SQL. Microsoft's recently published Raven system [11] offers significant efficiency gains to such integrations by analyzing both Python ML pipelines and SQL queries, applying cross-optimizations to them to increase in-database model inferencing performance. However, these systems cater to users with an extensive background in ML programming languages to build the models and pipelines themselves. Google's BigQuery ML [12] offers integrated SQL syntax for some ML tasks, relying on a heavily parameterized API of primarily table-valued functions.
Modified on	07/02/2022 15:39

Name	IntegratedML~ Every SQL Developer is a Data Scientist
Number of Coding References	8
Number of Codes Coding	4
Coverage	17.60%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Software Mantainability Aspects
Created on	04/02/2022 14:30

Coded Text	We propose IntegratedML, an embedded database capability to integrate machine learning into SQL applications. First, IntegratedML builds on an extensible set of AutoML engines through a common API, allowing for fully automated feature engineering, model selection and hyperparameter tuning directly from the embedding SQL. Second, IntegratedML offers idiomatic SQL syntax for typical ML concepts, distilling model deployment down to a familiar abstraction: the scalar function. IntegratedML implicitly maps relational attributes to model features, providing flexible, ergonomic syntax for model training and prediction. Finally, IntegratedML exploits a novel internal tuple batching scheme during query processing, enabling efficient predictions, regardless of the context in which the scalar prediction functions are invoked. Such efficient scoring of models as part of the query runtime itself, also referred to as in -DBMS inferencing, has been cited as a trend for the new decade [5]. IntegratedML is provided as an embedded capability of the InterSystems IRIS Data Platform, a multi-model DBMS that includes ANSI standard support for DDL/DML operations and SQL queries. While InterSystems IRIS is a commercial product, IntegratedML presents an architecture and syntax the authors believe is of interest to the SQL community as a whole and hope this paper contributes to a discussion on synergies between SQL and ML.
Modified on	07/02/2022 15:39

Name	IntegratedML~ Every SQL Developer is a Data Scientist
Number of Coding References	8
Number of Codes Coding	4
Coverage	17.60%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	In machine learning, trial and error are an integral part of the methodology, with a vast number of options to choose from in feature engineering, model selection and hyperparameter tuning phases of the ML “development process”. Making the right choices requires experience, much of which is still in academia and specialist research divisions of big enterprises.
Modified on	06/06/2022 10:23
Name	IntegratedML~ Every SQL Developer is a Data Scientist
Number of Coding References	8
Number of Codes Coding	4
Coverage	17.60%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	06/01/2022 15:15
Coded Text	Deployment is challenging because ML models often rely on libraries and programming environments that are very different from those used for production applications. Specifically, data pipelines and model scoring code are often expressed in procedural languages that are not easily mapped to the SQL environment where the data is stored. Typical practices to overcome this problem include redeveloping ready-built models in the target programming environment, leveraging model definition standards (e.g. PMML [2] and ONNX [3]) or hosting the model from a different system. Redevelopment is obviously costly and error-prone compared to the simplicity of invoking a hosted model through a simple REST service, but the latter may add significant latency and security concerns. This comes on top of the challenges associated with managing a heterogeneous topology.
Modified on	06/06/2022 10:25

Name	Juneau~ data lake management for Jupyter
Number of Coding References	8
Number of Codes Coding	3
Coverage	17.45%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	04/02/2022 13:03
Coded Text	
Modified on	07/02/2022 15:33

Name	Juneau~ data lake management for Jupyter
Number of Coding References	8
Number of Codes Coding	3
Coverage	17.45%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	04/02/2022 13:03

Coded Text	EXAMPLE 1.3 (AUGMENTING FEATURES). Another commo- n task for data scientists is to find additional or alternative features for the given data instances that may lead to a better performance. Especially in the collaborative setting, one data scientist may perform a specific feature engineering on a data set, while another may do it in a different way. It can be helpful for data scientists to be recommended with other feature engineering possibilities. EXAMPLE 1.4 (FINDING WORKFLOWS FOR DATA). Given a widely used and related table, a data scientist may want to see examples of how the table is loaded or cleaned, what analysis have been performed on it, and so on. Generally, this requires us to search for workflows using the table or related tables, potentially featuring specific operations.
Modified on	07/02/2022 15:31

Name	Juneau~ data lake management for Jupyter
Number of Coding References	8
Number of Codes Coding	3
Coverage	17.45%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	04/02/2022 13:03
Coded Text	In collaborative settings such as multi-investigator laboratories, data scientists need improved tools to manage not their data records but rather their data sets and data products, to facilitate both provenance tracking and data (and code) reuse within their data lakes and file systems. We demonstrate the Juneau System, which extends computational notebook software (Jupyter Notebook) as an instrumentation and data management point for overseeing and facilitating improved dataset usage, through capabilities for indexing, searching, and recommending “complementary” data sources, previously extracted machine learning features, and additional training data.
Modified on	07/02/2022 15:30

Name	Juneau~ data lake management for Jupyter
Number of Coding References	8
Number of Codes Coding	3
Coverage	17.45%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	04/02/2022 13:03
Coded Text	In this demonstration, we present a prototype of JUNEAU system, which provides these capabilities. Our demonstration illustrates how indexing, searching, and reusing tabular data are supported for tabular, CSV, and relational datasets. JUNEAU addresses scientists’ need to search for prior tables (and related code) not merely by keyword, but by querying using an existing table and its provenance, to find other related tables. Within the Jupyter environment, users may select a table (dataframe) and directly search for related tables for different purposes. Motivating use cases. We outline the four use cases for finding related tables. EXAMPLE 1.1 (AUGMENTING TRAINING DATA). Often, data is captured in multiple sessions (perhaps by multiple users) using the same sensor device or tool. Given a table from one such session, the user may wish to augment his or her data, to form a bigger training or validation set for a machine learning algorithm. EXAMPLE 1.2 (LINKING DATA VIA ONTOLOGIES). Particul- arly in the life sciences, records in one database may have identifiers (e.g., “accession numbers”) linking to entries in another database or ontology. Such entries may transitively reference other entries, and each brings in additional fields that may be useful. It
Modified on	07/02/2022 15:31

Name	Juneau~ data lake management for Jupyter
Number of Coding References	8
Number of Codes Coding	3
Coverage	17.45%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	04/02/2022 13:03
Coded Text	The JUNEAU System replaces Jupyter Notebook's back-end and extends it user interface. Our back-end "data lake management" subsystem integrates relational and key-value stores to capture and index (1) any external files loaded by the notebooks; (2) intermediate data products produced by computational steps (cells) within the notebooks; (3) versioned cell content and notebook content, as in the right-hand side of Figure 1; (4) indices for rapidly retrieving tables and their provenance. We illustrate the basic architecture and functionality in Figure 4. As in the existing Jupyter Notebook software, the notebook interface interacts with a kernel (language interpreter) every time the user executes a cell. The cell contents are executed in the kernel, thus updating state in the kernel as well. JUNEAU fetches any new or changed tables (dataframes) from the kernel after each step, and it imports and indexes those in the backend. The user may interactively select any table within the notebook, and query the JUNEAU search engine for other tables already stored and indexed in the data lake which are related to the selected item. As we described in the introduction, users often want to search for other related tables using an existing table as a model, and possibly adding other filter criteria such as author, attribute name or content, or the name of a computational process that was involved in the provenance of the search result. The search may not purely be based on whether other tables have a common schema or joinable fields, but may also consider similarity of computational (provenance) steps.
Modified on	07/02/2022 15:32

Name	Juneau~ data lake management for Jupyter
Number of Coding References	8
Number of Codes Coding	3
Coverage	17.45%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	04/02/2022 13:05
Coded Text	
Modified on	07/02/2022 15:33

Name	Juneau~ data lake management for Jupyter
Number of Coding References	8
Number of Codes Coding	3
Coverage	17.45%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	04/02/2022 13:05
Coded Text	A challenge lies in transitioning from exploratory data analysis, possibly over a small amount of data, to something that can be regularized into a production workflow with full reproducibility and much larger datasets. Towards this goal, recent work has proposed using notebooks as a way of encoding repeated computational workflows [9], and others have developed extensions to ensure the code within notebooks is fully versioned and reproducible [10, 1, 6]. However, we argue that the next step must be to look not at notebooks as documents of code steps that access and produce data files — but rather as compilations of (possibly shared, possibly parameterized) computational steps operating on objects in a data lake. We seek to accelerate and regularize data science tasks by finding and recommending data related to current objects of interest to the user. We do this by tracking the relationships between data sets, data products, and code [5]. With the appropriate indexing and search capabilities, data import and data cleaning steps are made visible to future users to be reused; data scientists may find other related datasets with similar history provenance; users are able to query, based on a given source table or intermediate result, whether someone else has already linked two datasets or extracted sets of features. Ultimately, just as shared libraries and open-source repositories have accelerated and improved software engineering — reusable datasets, schemas, and computational workflow steps may improve the quality of data engineering.
Modified on	07/02/2022 15:31

Name	Juneau~ data lake management for Jupyter
Number of Coding References	8
Number of Codes Coding	3
Coverage	17.45%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Research gap in Maintaining ML systems
Created on	14/02/2022 15:40
Coded Text	We seek to accelerate and regularize data science tasks by finding and recommending data related to current objects of interest to the user. We do this by tracking the relationships between data sets, data products, and code [5]. With the appropriate indexing and search capabilities, data import and data cleaning steps are made visible to future users to be reused; data scientists may find other related datasets with similar history provenance; users are able to query, based on a given source table or intermediate result, whether someone else has already linked two datasets or extracted sets of features. Ultimately, just as shared libraries and open-source repositories have accelerated and improved software engineering — reusable datasets, schemas, and computational workflow steps may improve the quality of data engineering.
Modified on	14/02/2022 16:03

Name	Kafka-ML~ Connecting the data stream with ML~AI frameworks
Number of Coding References	3
Number of Codes Coding	1
Coverage	7.22%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Kafka-ML is an open-source framework that enables the man- agement of the pipeline of ML/AI applications through data streams. Kafka-ML is a novel framework for integrating ML frameworks and data streams, which are continuously growing thanks to disruptive and massive data production paradigms such as the IoT. It presents a paradigm shift from traditional and static datasets used in ML/AI frameworks into continuous and dynamic data streams, offering a user-friendly, ready-to-use3 and open 3 https://github.com/ertis-research/kafka-ml. 19 platform to the community that allows managing ML/AI pipeline steps such as the inference deployment in productions environments. Kafka-ML works with configurations. A configuration is a logical set of ML models that can be grouped for training and evaluation. This can be useful when it is required to evaluate and compare metrics (e.g., loss and accuracy) of a set of ML models or just to define a group of them that can be trained with the same and unique data stream in parallel. Therefore, in case of having n ML models, all of which require a data stream for training, only one data stream has to be sent to Apache Kafka if a configuration has been defined with them. This can also apply to data scientists, which may train hundreds of ML models for an application and keep only one to be used in production [50]. And it is particularly true when they perform hyperparameter optimization [7]. Fig. 2 depicts the pipeline to manage these ML models and configurations through Kafka-ML: (1) designing and defining ML models with a few lines of ML model source code; (2) creating a training configuration for ML models, i.e., selecting a set of ML models to be trained and evaluated; (3) deploying the configuration for training and evaluation by selecting corresponding hyperparameters; (4) ingesting the deployed configuration with training and optionally evaluation data stream; (5) deploying trained ML models in production for inference; and, finally, (6) feeding deployed trained models for inference to make predictions with data streams. All the steps related to feeding the ML model (training and inference) can be carried out with data streams. Moreover, most of the previous steps use a RESTful API, so the pipeline can be automatized. Datastores might not be needed anymore with the management of data streams in Kafka-ML (Section 4.7). Fig. 3 shows an overview of the Kafka-ML architecture. One of its main components is the back-end, which is responsible, among other things, for creating training jobs (Model training) and inference jobs for the trained models (Model inference) upon user request. As in many convectional web architectures, users do not interact directly with the back-end, but with the frontend, which communicates users with the back-end and provides a user-friendly UI for it. Data streams in Kafka-ML are received
Modified on	28/02/2023 12:09

Name	Kafka-ML~ Connecting the data stream with ML~AI frameworks
Number of Coding References	3
Number of Codes Coding	1
Coverage	7.22%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	OpenML is a web platform where users can openly share, upload, and explore results, scientific tasks, data analysis flows, and datasets. Results and metrics ofML models can also be shared and compared in Kafka-ML. Moreover, data streams can also be managed and shared, as we will see in Section 4.7. Bazaar provides an AutoML system that – through ML primitives and a specification for data processing – automates and facilitates tasks such as the selection of the learning algorithm and hyper parameter tuning. This is an interesting approach to reduce the complexity of such required steps in ML/AI applications. However, it is tailored to a specific solution. In Kafka-ML we have adopted a state-ofthe-art and well-known ML framework like TensorFlow for ML definition. We will consider how to optimize these steps in ML/AI pipelines and frameworks supported by Kafka-ML in the near future. Google Cloud AutoML provides high-quality ML models with little effort and no advanced knowledge of the subject. Reaching the quality of these models is beyond the scope of Kafka-ML. However, Kafka-ML provides an accessible platform, where only a few lines of ML model source code are required to start an ML/AI pipeline with data streams. Furthermore, Kafka-ML is an open-source project available for both experts and non-experts on ML/AI. To conclude this section, Table 1 summarizes and compares previous approaches with Kafka-ML.
Modified on	28/02/2023 12:09

Name	Kafka-ML~ Connecting the data stream with ML~AI frameworks
Number of Coding References	3
Number of Codes Coding	1
Coverage	7.22%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	To the best of our knowledge, Kafka-ML is the first open- source framework to provide an ML/AI pipeline solution to integrate ML/AI and data streams. Nevertheless, other approaches have similar goals or have provided some of the functionalities offered by Kafka-ML as described below. Kubeflow [13] is a powerful ML toolkit for Kubernetes. In Kubeflow, users can configure multiple steps of an ML/AI pipeline such as hyper-parameters, pre-processing, training and inference. However, when running a Kubeflow pipeline, such as the official example for the Google Cloud Platform,2 there may be some steps that are not required in the Kafka-ML pipeline, especially the ones that require building containers for training, and inference. In Kafka-ML, users merely need to interact with the Web UI (User Interface) for training and inference. In addition, data stream support has to be manually developed by Kubeflow ML developers and users. In Kafka-ML, the data stream management through Apache Kafka is supported in all of the pipeline. Kubeflow provides great support for Kubernetes and ML multi-frameworks, which are supported by a large ecosystem and community that are far beyond the scope and functionalities offered by Kafka-ML. Therefore, it may be worth studying the way of integrating both systems in the near future. NVIDIA Deep Learning GPU Training System (DIGITS) [14] provides an interactive Web UI for training and inference of deep neural networks (DNNs) on multi-GPU systems. Unlike Kafka-ML, DIGITS is not a framework in itself, but a wrapper for NVCaffe, Torch, and TensorFlow, which provides a Web UI to these frameworks rather than dealing with them directly on the commandline. The main advantages of DIGITS are its native support for GPUs and three ML frameworks, the release of pre-trained models, and the functionality to see the accuracy and loss in real-time. Nevertheless, DIGITS does not support training and inference through data streams (datasets have to be imported instead) or the deployment of these tasks through containers for scaling, it has a dependency on GPUs and may require writing a source code on top of these frameworks. Regarding enterprise solutions, Amazon SageMaker [15,25] is an ecosystem provided as part of Amazon Web Services (AWS) which offers a continuous integration and continuous delivery (CI/CD) service for ML/AI. Among its countless services, the following stand out: hyperparameter optimization, incremental training, and elastic (pausing and resuming) learning. SageMaker provides supports for ML frameworks such as Tensorflow, MXNet and Pytorch, and Amazon Kinesis helps with real-time data ingestion at scale. Although data streams can be seamlessly integrated 2 https://www.kubeflow.org/docs/gke/gcp-e2e/. C. Martín, P. Langendoerfer, P.S. Zarrin et al. Future Generation Computer Systems 126 (2022) 15–33 for inference, for training they rely on data lakes such as Amazon S3, unlike Kafka-ML, an open-source project where both phases can be performed with data streams, without the need for data lakes. Despite the fact that Algoritmia [16] does not provide much information, it is apparently more focused on the delivery of models for inference and the automation of some pipeline processes and their monitoring. Valohai [17] enables the management of ML/AI pipelines in Kubernetes like Kubeflow. Valohai helps building complex ML/AI pipelines that can be automated including steps such as hyperparameter optimization. No information is provided regarding data streams. However, one of the most notable features of this platform is its automated and extensive version control for ML model traceability. MOA [26] is a framework for online learning and data stream mining. MOA provides a graphical interface where users can execute and visualize ML tasks, including a collection of ML algorithms implementations for classification, regression, and clustering among others. Although Kafka-ML supports data streams, Kafka-ML and TensorFlow are not (yet) good at supporting online learning. On the other hand, Kafka-ML provides support for TensorFlow/Keras models and their large community, instead of creating a new framework with its own source code that could limit its adoption. Scikit-multiflow [27] is another framework for online learning, in this case for the popular framework scikitlearn, however it does not provide a Web interface or a full control of an ML/AI pipeline. Ullah et al. [28] propose an ML-based system for real-time pro- cessing of data streams and action recognition. The data stream processing is performed in different ML steps for action recognition, including a pre-trained convolutional neural network for feature extraction, a deep autoencoder to learn temporal changes of actions, and a non-linear learning approach for

classification. This solution also includes an online learning phase to continuously improve the model. This solution has been defined for action recognition and does not provide a generic framework for the deployment and management of ML/AI applications and data streams like Kafka-ML. Moreover, this solution lacks a distributed stream integration like Kafka-ML for better management and scalability of data streams. Kafka-ML follows a different approach as compared to other distributed data stream frameworks [29,30] that are growing in the era of big data and data streams, such as Apache SAMOA [31], Apache Flink [32], Apache Spark and Spark Streaming [33], and the Lambda architecture [34,35]. Apache SAMOA is currently undergoing incubation at Apache and aims at enabling the development of ML algorithms through data streams without directly dealing with the complexity of underlying processing engines (e.g., Apache Storm [36] and Apache Samza [37]). Although without native support for data streams, SystemML [38] also follows a similar approach than SAMOA and provides a declarative ML language to abstract the development of ML/AI applications, which can be finally deployed on distributed systems such as Spark. Apache Flink provides a framework to perform computation over data streams at an in-memory speed and at any scale. Apache Spark is an engine for large-scale data processing, and Spark streaming is an extension of it for scalable and high-performance stream processing. In addition, the Lambda architecture allows the processing of large amounts of data in real time by having real-time and batch layers of processing. In general, these frameworks provide distributed engines for distributing any kind of computation with data streams, although they have limited support for, or do not have a special focus on, facilitating ML/AI pipelines and popular ML/AI frameworks such as TensorFlow and their large range of ML/AI solutions and community, as Kafka-ML does. Moreover, Kafka-ML can also enable the deployment of high availability and fault-tolerant ML/AI pipelines. MLlib [39], another

extension of Apache Spark, supports ML/AI algorithms for common learning settings including regression, collaborative filtering and clustering in Spark deployments. Nevertheless, this support is limited for a number of common algorithms and does not support, for example, deep learning ML models such as Kafka-ML with TensorFlow. Kafka Streams [40] is a Java library which allows building

real-time processing applications with data streams allocated in Apache Kafka, i.e., the input and output of Kafka Streams applications are Kafka topics. Kafka Streams provides abstractions and operators (stateless and stateful) to work both with data streams and tables (data stream aggregations) for the development of streaming and microservice applications. Faust [41] is another open-source stream processing library which ports the ideas from Kafka Streams to Python. Like Kafka Stream, Faust provides support for data stream processing, sliding windows, and aggregate counts. Its interface is less verbose than Kafka Streams, and applications can be developed with very few lines of source code. While not directly supporting ML/AI applications, these streaming libraries rely on Apache Kafka as a distribution core as Kafka-ML does. Despite its name, TensorFlow Serving [42] is an agnostic system to serve ML models in general and TensorFlow in particular. TensorFlow Serving provides a canonical Remote Procedure Call (RPC) server that serves ML models from a hosted service, where models can be uploaded and updated when required. Users have full control over the ML models submitted, and this allows to have multiple ML model versions simultaneously or rollback a version. This inference module was designed by Google TensorFlow for production environments. Regarding Kafka-ML, this module works with RPC requests instead of data streams, which can be fully controlled and reliably stored in the distributed log provided by Apache Kafka. According to Assuncao et al. [43], we are facing the fourth generation of distributed data stream processing frameworks, where certain elements are located on the edge of the network to reduce latency. Calo et al. [44] propose a system for serving ML models on edge servers and optimizing IoT communications. Spanedge [45] is presented as a system to distribute stream processing across a central and some near-the-edge data centers. Going further, Pisani et al. [46] propose cross-platform code execution directly in IoT devices. The distribution of computation in all these layers, namely edge, fog, and cloud, what is known as the Cloud-to-Things continuum [47], will definitely optimize the execution of ML/AI applications, and Kafka-ML will deploy ML/AI applications in this continuum in future work. MODELDB [7] was one of the first approaches in providing an open-source ML model management system, i.e., a system for ML model, metadata, and experiment versioning and management. All the information related to ML models (hyperparameters, type, author, and so on) can be uploaded to MODELDB to have full control of the experiments and optimizations carried out. ModelHub [9] provides a service for publishing, reusing and discovering Deep Learning models, such as a GitHub repository for software components. This is of special interest to have a better control of the developed ML models and to let users explore the ML model space through the network architecture and hyperparameter values. These projects are basically a metadata storage (or a git system) to track information about ML models and experiments, thus they do not manage or deploy any step of ML/AI pipelines like Kafka-ML. Although Kafka also allows obtaining information about the performance of ML models (automatically), it will take ideas from these projects to improve the information provenance and to have a better control of ML model versions. Finally, Kafka-ML is related to some extent to AutoML projects such as OpenML [48], Bazaar [49], and Google Cloud AutoML [18].

Modified on 28/02/2023 12:09

Name	Kafka-ML~ Connecting the data stream with ML~AI frameworks Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	KFIML~ Kubernetes-Based Fog Computing IoT Platform for Online Machine Learning
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	KFIML~ Kubernetes-Based Fog Computing IoT Platform for Online Machine Learning Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	KOI~ An Architecture and Framework for Industrial and Academic Machine Learning Applications
Number of Coding References	8
Number of Codes Coding	3
Coverage	15.59%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	
Modified on	15/02/2022 12:32

Name	KOI~ An Architecture and Framework for Industrial and Academic Machine Learning Applications
Number of Coding References	8
Number of Codes Coding	3
Coverage	15.59%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	
Modified on	15/02/2022 12:33

Name	KOI~ An Architecture and Framework for Industrial and Academic Machine Learning Applications
Number of Coding References	8
Number of Codes Coding	3
Coverage	15.59%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	
Modified on	15/02/2022 12:33

Name	KOI~ An Architecture and Framework for Industrial and Academic Machine Learning Applications
Number of Coding References	8
Number of Codes Coding	3
Coverage	15.59%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	the components that are part of the high-level architec- ture. The central KOI-API component handles the persistence of the objects described in Sect. 4.1 on top of a role-based authorization schema. The usage of a relational database meets our requirement to express the object structure from Fig. 1. Since there is the need to save multiple megabytes or even gigabytes of data, there is also a file-based storage, which will be referenced by the relational database through unique identifiers. The API-component provides an interface
Modified on	15/02/2022 12:32

Name	KOI~ An Architecture and Framework for Industrial and Academic Machine Learning Applications
Number of Coding References	8
Number of Codes Coding	3
Coverage	15.59%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	There are runtime environments and inference frameworks, like NVidia Ten- sorRT [9], available, alongside open standards for exchanging trained models, like ONNX [10]. As they focus on the performant deployment of pre-trained models, they are out of scope for this research as this paper aims towards the lifecycle management of iteratively training machine learning systems. Cortex.dev [11] is an open-source stack for machine learning engineering and deployment. The critical difference between the proposed approach and said software is that Cortex.dev deploys machine learning models as web-based APIs, while the proposed architecture aims for local execution of inference tasks. Therefore, this paper provides an edge computing solution instead of a cloud computing solution. Tensorflow Serving is the official software toolkit for serving machine learning models build and trained using the TensorFlow software system [1,2]. Like Cortex.dev, TensorFlow Serving aims for a web-based solution for running models for inference. The documentation suggests the use of Docker images containing the serving API for local inference [12]. Both systems are designed for deployment and miss the continual learning focus in their design, even though they do not prevent the user from doing so. The open-source library horovod [13] enables the user to scale single GPU training software up towards multiple GPUs to speed up the training process. Distributing computation between multiple GPUs is not the aim of this project, and horovod does not provide deployment to the previously described edge systems. Furthermore, horovod could be used together with the proposed system. Allegro.ai Trains [14] is a set of tools for machine learning, which aims for cooperative and concurrent development and training of machine learning models. It has a rich feature set available for academic and industrial developers to organize developing machine learning code and optimizing hyperparameters. As a development tool, it lacks the focus on integration into products, edge computing, and build-in mechanics for continual learning.
Modified on	15/02/2022 12:31

Name	KOI~ An Architecture and Framework for Industrial and Academic Machine Learning Applications
Number of Coding References	8
Number of Codes Coding	3
Coverage	15.59%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	This paper proposes a new software architecture and system called “KOI”. The name is a German abbreviation for “Kognitive Objektorientierte Inspektion”, which translates to object-oriented cognitive inspection. Key features of this architecture are its simple object-oriented design, separation of concerns through multiple components, and the vast amount of deployment scenarios it supports. The following section describes the architectural decisions made while developing this software. Layered C4-Diagrams [15] describe every main component of the system. Some deployment scenarios will conclude this section.
Modified on	15/02/2022 12:32

Name	KOI~ An Architecture and Framework for Industrial and Academic Machine Learning Applications
Number of Coding References	8
Number of Codes Coding	3
Coverage	15.59%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	06/01/2022 15:05
Coded Text	Single System Deployment. All described components run on the same target system. The system collects its dataset and trains an individual instance of a model using its computation hardware. Although the benefits of collaboration, shared datasets, and training distribution are lost using this approach, it enables the same model deployment mechanisms as the other examples. Furthermore, the user can easily upgrade the system to become part of one of the other deployment scenarios. Multiple Systems/Personal Cloud. Multiple systems fulfilling the same inspection task, work cooperatively. By sharing data collected on a subset of systems, the overall dataset gives a broader view of the given problem. Transferring the training task to one of the systems or an external training server frees the target systems from the additional computational load. Through sharing a dataset and training, all connected systems have the same performance when executing for inference. Machine Learning as a Service. A machine learning expert develops a model and serves individual instances to a set of customers. Each customer connects an arbitrary number of systems to his instance and collects his dataset while sharing only the model with the others. Personal workers perform the training procedure. These workers run in a cloud service or onsite, hosted by the machine learning expert or the consumer himself.
Modified on	15/02/2022 12:33

Name	KOI~ An Architecture and Framework for Industrial and Academic Machine Learning Applications
Number of Coding References	8
Number of Codes Coding	3
Coverage	15.59%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:22
Coded Text	Functional Requirements – Deployment of different machine learning models in productive environments Rationale: Deploy new machine learning models on a variety of inspection systems with different applications and test strategies. The system should be able to manage and deploy multiple models on different systems at the same time. – Iterative training of models using additional insights and data Rationale: As complete datasets are hard to acquire, the system should be able to train models on incomplete datasets and continue or re-run the training at a later time, while preserving the already achieved classification quality. This early training enables fast deployment of machine learning in productive environments, with as minimal initial effort as possible. – Asynchronously collecting samples from different sources Rationale: Different processes and systems can become a valuable source of training data. The ability to asynchronously accumulate training samples frees the developer from searching and composing training sets in advance. – Support for label-requests for continual and active learning Rationale: The models can, at any time, request additional insight into the data. As this feature is part of the deployment system itself, the programming overhead for the machine learning developer minimizes. In an environment where the validity of labels can not be guaranteed, the active request for additional data insight and the distribution of workload between users increases the value of the training samples over time. KOI: An Architecture and Framework for Machine Learning Applications 117 – User- and role-management to control access to models and instances Rationale: Controlling what models and instances a user can see or interact with, enables many deployment scenarios, where different people work cooperatively using the same model. The role management prevents malicious changes and adversarial training input to productive machine learning models. The user management enables different persons to share the same computation hardware without sharing models, instances, or samples. – Lifecycle management of models and instances Rationale: When using continual training for machine learning, users want to preserve already achieved classification quality. Especially in a productive environment, the ability to go back to previous states of training is crucial. When wrong inputs lead to stuck training progress, the lifecycle management enables to user to continue training from a previous much more viable state. Non-functional Requirements – Edge computing allowing for inference execution right on the target system Rationale: Using the typically performant computational hardware of the inspection systems instead of a centralized inference system lowers the overall cost of setup. It makes the classification available where it is needed, even when offline. – Graphical user-interface for most common actions and observation Rationale: Being able to interact with the system and performing the most common task using a graphical interface increases the acceptance of the system and flattens the learning curve when first working with the proposed system. – Adaptive deployment for different heterogeneous systems Rationale: The support for heterogeneous systems and the use of platformindependent software enables the user to integrate the proposed system into many existing inspection systems and environments, without being forced to use a specific operating system or library. – Minimum boilerplate code for unhindered development Rationale: If the proposed system has as little as possible impact on the model development, the users are free to implement whatever they need. As there is no general approach to artificial intelligence, and there are many libraries and frameworks with different development paradigms available, the proposed system should not burden the user with unnecessary restrictions. – Distributed computation to enable the concurrent training of multiple models Rationale: By controlling which hardware computes which model, the user can steer the performance of the overall system. Crucial or frequently updating models can run on dedicated high-performance hardware. Multiple users can share the same computation hardware for non-time-critical problems.

Modified on	22/07/2022 11:55
Name	Kubric~ A scalable dataset generator
Number of Coding References	4
Number of Codes Coding	2
Coverage	1.98%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	22/07/2022 12:21
Coded Text	Data is the driving force ofmachine learning, with the amount and quality oftraining data often being more important for the performance ofa system than architecture and training details. But collecting, processing and annotating real data at scale is difficult, expensive, and frequently raises additional privacy, fairness and legal concerns. Synthetic data is a powerful tool with the potential to address these shortcomings: 1) it is cheap 2) supports rich ground-truth annotations 3) offers full control over data and 4) can circumvent or mitigate problems regarding bias, privacy and licensing. Unfortunately, software tools for effective data generation are less mature than those for architecture design and training, which leads to fragmented generation efforts.
Modified on	16/02/2023 09:37
Name	Kubric~ A scalable dataset generator
Number of Coding References	4
Number of Codes Coding	2
Coverage	1.98%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	22/07/2022 12:22
Coded Text	be able to model as much as possible of the structure and complexity of real data. The Cycles raytracing engine of Blender supports a high level of realism and can model complex visual phenomena such as reflection, refraction, indirect illumination, subsurface-scattering, motion-blur, depth of field, etc. Studying these effects is important, and they also help to reduce the generalization gap. Scalability. Data generation workloads can range from simple toy-data prototyping all the way to generating massive amounts of high-resolution video data. To support this entire range of usecases, Kubric is designed to seamlessly scale from a local workflow to running large jobs on thousands of machines in the cloud. Portable and Reproducible. To facilitate reuse of datageneration code, it is important that the pipeline is easy to setup and produces the same results — even when executed on different machines. This is especially important due to the difficulty in installing the Blender Python module [31] and the substantial variations between versions. By distributing a Kubric Docker image, we ensure portability and remove the bulk of the installation pain. Data Export. Kubric by default exports a rich set of ground truth annotations from segmentation, optical flow, surface normals and depth maps, to object trajectories, collision events and camera parameters. We also introduce SunDs (see Sec. 3.4), a unified multi-task frontend for richly annotated scene-based data.
Modified on	16/02/2023 09:40

Name	Kubric~ A scalable dataset generator
Number of Coding References	4
Number of Codes Coding	2
Coverage	1.98%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	22/07/2022 12:22
Coded Text	Kubric is a high-level python library that acts as glue between: a rendering engine, a physics simulator, and data export infrastructure; see Figure 2. Its main contribution is to streamline the process and reduce the hurdle and friction for researchers that want to generate and share synthetic data. 3.1. Design Principles Openness. Data-generation code should be freely usable by researchers both in academia and in industry. Kubric addresses this by being open-source with an Apache2 licence, and by only using software with similarly permissive licenses. Together with the use of free 3D assets and textures, this enables researchers to share not just the data, but also enable others to reproduce and modify it. Ease of use. The fragmentation of computer graphics formats, conventions and interfaces is a major pain point for setting up and reusing data-generation code. Kubric minimizes this friction by offering a simple object-oriented API interface with PyBullet and Blender behind the scenes, hiding the complexities of setup, data transfer, and keeping them in sync. We also provide pre-processed 3D assets from a variety of data sources, that can be used with minimal effort. Realism. To be maximally useful, a data-generator should
Modified on	16/02/2023 09:40

Name	Kubric~ A scalable dataset generator
Number of Coding References	4
Number of Codes Coding	2
Coverage	1.98%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	22/07/2022 12:22
Coded Text	To address these problems we introduce Kubric, an open-source Python framework that interfaces with PyBullet and Blender to generate photo-realistic scenes, with rich annotations, and seamlessly scales to large jobs distributed over thousands of machines, and generating TBs of data. We demonstrate the effectiveness ofKubric by presenting a series of13 different generated datasets for tasks ranging from studying 3D NeRF models to optical flow estimation. We release Kubric, the used assets, all ofthe generation code, as well as the rendered datasets for reuse and modification
Modified on	16/02/2023 09:38

Name	Kubric~ A scalable dataset generator Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Large-scale machine learning systems in real-world industrial settings~ A review of challenges and solutions
Number of Coding References	10
Number of Codes Coding	5
Coverage	4.61%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	
Modified on	11/02/2022 14:22

Name	Large-scale machine learning systems in real-world industrial settings~ A review of challenges and solutions
Number of Coding References	10
Number of Codes Coding	5
Coverage	4.61%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	Background : Developing and maintaining large scale machine learning (ML) based software systems in an industrial setting is challenging. There are no well-established development guidelines, but the literature contains reports on how companies develop and maintain deployed ML-based software systems. Objective : This study aims to survey the literature related to development and maintenance of large scale MLbased systems in industrial settings in order to provide a synthesis of the challenges that practitioners face. In addition, we identify solutions used to address some of these challenges. Method : A systematic literature review was conducted and we identified 72 papers related to development and maintenance of large scale ML-based software systems in industrial settings. The selected articles were qualitatively analyzed by extracting challenges and solutions. The challenges and solutions were thematically synthesized into four quality attributes: adaptability, scalability, safety and privacy. The analysis was done in relation to ML workflow, i.e. data acquisition, training, evaluation, and deployment. Results : We identified a total of 23 challenges and 8 solutions related to development and maintenance of large scale ML-based software systems in industrial settings including six different domains. Challenges were most often reported in relation to adaptability and scalability. Safety and privacy challenges had the least reported solutions.
Modified on	11/02/2022 14:20

Name	Large-scale machine learning systems in real-world industrial settings~ A review of challenges and solutions
Number of Coding References	10
Number of Codes Coding	5
Coverage	4.61%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13
Coded Text	
Modified on	11/02/2022 14:22

Name	Large-scale machine learning systems in real-world industrial settings~ A review of challenges and solutions
Number of Coding References	10
Number of Codes Coding	5
Coverage	4.61%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	
Modified on	11/02/2022 14:21

Name	Large-scale machine learning systems in real-world industrial settings~ A review of challenges and solutions
Number of Coding References	10
Number of Codes Coding	5
Coverage	4.61%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	
Modified on	11/02/2022 14:22

Name	Large-scale machine learning systems in real-world industrial settings~ A review of challenges and solutions
Number of Coding References	10
Number of Codes Coding	5
Coverage	4.61%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	
Modified on	11/02/2022 14:23

Name	Large-scale machine learning systems in real-world industrial settings~ A review of challenges and solutions
Number of Coding References	10
Number of Codes Coding	5
Coverage	4.61%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Background : Developing and maintaining large scale machine learning (ML) based software systems in an industrial setting is challenging. There are no well-established development guidelines, but the literature contains reports on how companies develop and maintain deployed ML-based software systems. Objective : This study aims to survey the literature related to development and maintenance of large scale MLbased systems in industrial settings in order to provide a synthesis of the challenges that practitioners face. In addition, we identify solutions used to address some of these challenges. Method : A systematic literature review was conducted and we identified 72 papers related to development and maintenance of large scale ML-based software systems in industrial settings. The selected articles were qualitatively analyzed by extracting challenges and solutions. The challenges and solutions were thematically synthesized into four quality attributes: adaptability, scalability, safety and privacy. The analysis was done in relation to ML workflow, i.e. data acquisition, training, evaluation, and deployment. Results : We identified a total of 23 challenges and 8 solutions related to development and maintenance of large scale ML-based software systems in industrial settings including six different domains. Challenges were most often reported in relation to adaptability and scalability. Safety and privacy challenges had the least reported solutions.
Modified on	11/02/2022 14:20

Name	Large-scale machine learning systems in real-world industrial settings~ A review of challenges and solutions
Number of Coding References	10
Number of Codes Coding	5
Coverage	4.61%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56
Coded Text	2.1. SE and AI SE focuses on the processes and methods that are used in designing, developing and maintaining software. SE research seeks to create (toolsupported) methods and approaches for ensuring robust and reliable design of software. ML techniques aim to identify useful patterns in data and perform inference using the learned patterns. The ML inference can be based on a classification or regression problem, in which unseen data is determined to belong to one out of a number of classes in classification or is used to predict outcome in regression.
Modified on	11/02/2022 14:20

Name	Large-scale machine learning systems in real-world industrial settings~ A review of challenges and solutions
Number of Coding References	10
Number of Codes Coding	5
Coverage	4.61%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56
Coded Text	2.2. The development process of ML-based systems Few studies report about a general end-to-end development process of ML-based systems in industrial contexts [11] . Currently there is much focus on describing best practices for developing ML systems by practitioners with experiences in developing large -scale ML systems, for example, at Google [18] and Microsoft [11] . Data, including its collection and management, is a crucial element that consumes most of the development efforts of ML-based system [11] . In ML systems, the behaviour of the system is not specified directly in code, but rather it is continuously learned from the data. This aspect of ML systems challenges the application of conventional SE processes and practices, such as software testing [17,19] . Increasingly there is a need to integrate the development workflow of ML system into existing SE processes and
Modified on	11/02/2022 14:20

Name	Large-scale machine learning systems in real-world industrial settings~ A review of challenges and solutions
Number of Coding References	10
Number of Codes Coding	5
Coverage	4.61%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56
Coded Text	2.3. Related work State-of-the-art surveys have been conducted in the area of ML. Their focus have ranged from general use of ML techniques [29] and application in specific cases, such as automotive [30] and telecommunication [31] to specific aspects of ML, such as understanding how explanations are generated from an ML algorithm [32] or ML frameworks [24] . Within the field of SE, surveys have been conducted on the use of AI techniques to support SE activities [15,33] . In addition, an overview of ML testing research is provided
Modified on	11/02/2022 14:21

Name	Looper~ An End-to-End ML Platform for Product Decisions
Number of Coding References	9
Number of Codes Coding	1
Coverage	8.91%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	Data-centric ML development is a recent concept of refocusing ML development from models to data [35]. It supports software personalization with off-the-shelf models, where collecting the right data and selecting the appropriate class of models become primary differentiators [37]. Compared to developing and training ML models, data adequacy is often overlooked [45], and product platforms must use automation to compensate. Per Andrew Ng, “everyone jokes that ML is 80% data preparation, but no one seems to care” [44]. Yet, directly handling data sets and ML models in product code is cumbersome. Instead, software-centric ML integration with data collection and decision-making APIs offers a front-end to MLOps automation (Sections 2.2 and 3.2). Additionally, ML development often neglects structure in product evaluation data (Section 3.4). Vertical ML platforms lower barriers to entry and support the entire lifecycle ofML models (Figure 1) in a repeatable way. Horizontal ML platforms provide storage, support data pipelines and offer basic services, whereas vertical platforms foster the reuse of not only ML components, but also workflows. At firms like Google, Meta, LinkedIn, Netflix, specialized end-to-end vertical platforms
Modified on	16/02/2023 08:54

Name	Looper~ An End-to-End ML Platform for Product Decisions
Number of Coding References	9
Number of Codes Coding	1
Coverage	8.91%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	drive flagship product functionalities, such as recommendations. They have also been applied to software development, code quality checks, and even to optimize algorithms such as sorting and searching [15]. Platforms are built on ML frameworks like TensorFlow [1] and PyTorch [33] that focus on modeling for generic ML tasks, support hardware accelerators, and act as toolboxes for application development [24, 36]. Supporting smart strategies requires general-purpose vertical platforms to offer end-to-end ML lifecycle management. General-purpose vertical ML platforms can be internal to a company — Apple's Overton [41] and Uber's Michelangelo [27], — or broadly available to cloud customers — Google's Vertex, Microsoft's Azure Personalizer [2] and Amazon Personalize. A common theme is to help engineers "build and deploy deep-learning applications without writing code" via high-level, declarative abstractions [37]. Improving user experience and system performance with ML remains challenging [40] as correlations in data found by ML models might not lead to causal improvements. Little is known about optimizing for product goals [37, 52]. We develop support for data-driven real-time smart strategies via a general-purpose vertical end-to-end ML platform called Looper, internal to Meta, for rapid, low-effort deployment of moderatesized models. Looper is a declarative ML system [27, 36, 37, 41] with coding-free full-lifecycle management of smart strategies via a GUI
Modified on	16/02/2023 08:55

Name	Looper~ An End-to-End ML Platform for Product Decisions
Number of Coding References	9
Number of Codes Coding	1
Coverage	8.91%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	In contrast to heavy-weight ML models for vision, speech and NLP that favor offline inference (with batch processing) and motivate applications built around them, we address the demand for smart strategies within software applications and products. These smart strategies operate on metadata — a mix of categorical, sparse, and dense features, often at different scales. Respective ML models are lightweight, they can be re-trained regularly and deployed quickly on shared infrastructure in large numbers. Downside risks are reduced via (□) simpler data stewardship, (□□) tracking product impact, (□□□) failsafe mechanisms to withdraw poorly performing models. Smart strategies have a good operational safety record and easily improve naive default behaviors. The human labeling process common for CV and NLP fails for metadata because relevant decisions and predictions (a) only make sense in an application context, (b) in cases like data prefetch (Section 4.3) only make sense to engineers, (c) may change seasonally, and even daily. Instead of human labeling, our platform interprets user-interaction and system-interaction metadata as either labels for supervised learning or rewards for reinforcement learning. To improve operatonal safety and training efficiency, we rely on batchmode (offline) training, even for reinforcement learning. Given real-time inference, model agility beyond daily re-training is supported by real-time engineered features, such as event counters. Our platform ensures fast onboarding, robust deployment and low-effort maintenance of multiple smart strategies where positive impacts are measured and optimized directly in application terms (Appendix B). To this end, we separate application code from platform code, and leverage existing horizontal ML platforms with interchangeable models for ML tasks (Figure 1). Intended for company engineers, our platform benefits from high-quality data and engineered features in the company-wide feature store [39]. To simplify onboarding for product teams and keep developers productive, we automate and support • Workflows avoided by engineers [45], e.g., feature selection and preprocessing, and tuning ML models for metadata. • Workflows that are difficult to reason about, e.g., tuning ML models to product metrics. We first introduce several concepts for platform design. The decision space captures the shape of decisions within an application which can be made by a smart strategy. It can be just {0,1}
Modified on	16/02/2023 09:04

Name	Looper~ An End-to-End ML Platform for Product Decisions
Number of Coding References	9
Number of Codes Coding	1
Coverage	8.91%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	in product software. Product engineers delegate trainingdata quality concerns (missing or delayed labels, etc) to the platform. Missing data are represented by special values. • An online-first approach. Looper API logs live features and labels at the decision and feedback points, then joins and filters them via real-time stream processing. This immediate materialization avoids data hygiene issues [2] and storage overhead: it keeps training and inference consistent and limits label leakage by separating features and labels in time. Looper's complete chain ofcustody fordata (without exposing data tables or files) helps prevent engineering mistakes.
Modified on	16/02/2023 09:04
Name	Looper~ An End-to-End ML Platform for Product Decisions
Number of Coding References	9
Number of Codes Coding	1
Coverage	8.91%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	Modern software systems and products increasingly rely on machine learning models to make data-driven decisions based on interactions with users, infrastructure and other systems. For broader adoption, this practice must (□) accommodate product engineers without ML backgrounds, (□□) support finegrain product-metric evaluation and (□□□) optimize for product goals. To address shortcomings of prior platforms, we introduce general principles for and the architecture of an ML platform, Looper, with simple APIs for decision-making and feedback collection. Looper covers the end-to-end ML lifecycle from collecting training data and model training to deployment and inference, and extends support to personalization, causal evaluation with heterogenous treatment effects, and Bayesian tuning for product goals
Modified on	16/02/2023 08:52

Name	Looper~ An End-to-End ML Platform for Product Decisions
Number of Coding References	9
Number of Codes Coding	1
Coverage	8.91%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	Platform architecture: the core Traditional ML pipelines build training data offline, but our platform uses a live feature store and differs in two ways: • Software-centric vs. data-centric interfaces. Rather than passed via files or databases, training data are logged from product surfaces as Looper APIs intercept decision points
Modified on	16/02/2023 09:04

Name	Looper~ An End-to-End ML Platform for Product Decisions
Number of Coding References	9
Number of Codes Coding	1
Coverage	8.91%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	Specialized Vertical Platform for App Performance & Scalability used for images, videos, speech, custom recommendation systems Problem type Data sources & collection domain-specific ML tasks, e.g., ranking for a product surface ● domain- & problem-specific data sources ● data collection via custom APIs ● extreme data capacity possible Data preparation Model selection & optimization app specific data transforms; extensive customization ● large custom models ● manual feature selection & eng ● custom model optimization ● transfer learning possible ● a small set of use cases General-purpose Vertical Platform for Usability & End-To-End Management used for smart strategies, configurable recommendation systems different ML tasks across variety of product domains ● diff. data sources for diff. apps ● data collection via standard APIs ● data capacity often limited by co-hosting Horizontal ML Platforms for Problem Space Coverage, ML Performance and Scalability multiple transforms available for different apps; limited customization ● modular architecture ● many standard model types ● smaller model sizes ● automated feature selection ● automated model optimization ● large variety of use cases Deployment & maintenance Product impact eval. & opt. ● app specific monitoring and alerts ● custom deployment; co-optimized with app stack ● end metrics rarely changed ● flexible monitoring and alerts ● co-hosted deployment; modular interface with multiple apps ● modularized & configurable APIs ● easy to add new metrics
Modified on	16/02/2023 08:57

Name	Looper~ An End-to-End ML Platform for Product Decisions
Number of Coding References	9
Number of Codes Coding	1
Coverage	8.91%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	to indicate whether a notification is shown. It can be a continuousvalue space for time-to-live (TTL) of a cache entry. It can be a data structure with configuration values for a SW system, such as a livevideo stream encoder. With reinforcement learning, the decision space matches well with the concept of action space. Application context captures necessary key information provided by a software system at inference time to make a choice in the decision space. The application context may be directly used as features or it may contain ID keys to extract the remaining features from the feature store (Section 3.3). Product metrics evaluate the performance of an application and smart strategies. When specific decisions can be judged by product metrics, one can generate labels for supervised learning, unlike for metrics that track long-term objectives. A proxy ML task casts product goals in mathematical terms to enable (□) reusable ML models that optimize formal objectives and (□□) decision rules that map ML predictions into decisions (Section 2.1). Setting proxy tasks draws on domain expertise, but our platofrm simplifies this process. Evaluation of effects on live data verifies that solving the proxy task indeed improves product metrics. Access to Meta's monitoring infrastructure helps detect unforeseen side effects. As in medical trials, (1) we need evidence of a positive effect, (2) side-effects should be tolerable, and (3) we should not overlook evidence of side-effects. On our platform, product developers define the decision space, allowing the platform to automatically select model type and hyperparameter settings. The models are trained and evaluated on live data without user impact, and improved until they can be deployed. Newly trained models are canaried (deployed on shadow traffic) before product use – such models are evaluated on a sampled subset oflogged features and observations, and offline quality metrics (e.g., MSE for regression tasks) are computed. This helps avoid degrading model quality when deploying newer models.
Modified on	16/02/2023 09:04

Name	Looper~ An End-to-End ML Platform for Product Decisions
Number of Coding References	9
Number of Codes Coding	1
Coverage	8.91%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	Traditional end-to-end ML systems go as far as to cover model publishing and serving [27, 36, 37, 41], but to our knowledge rarely track how the model is used in the software stack. Assessing and optimizing the impact of smart strategies, especially with respect to product goals, requires experimentation on all aspects of the modeling framework – from metric and model selection to policy optimization. To streamline this experimentation and reap its benefits, smart-strategies platforms must extend the common definition of end-to-end into the software layer. Software-centric ML integration [2, 15] – where data collection and decision-making are fully managed through platform APIs – enables both high-quality data collection and holistic experimentation. Notably, the platform can now keep track of all decision points and support A/B tests between different configurations. Well-defined APIs improve adoption among product engineers with limited ML background, and ML configuration can be abstracted via declarative programming or GUI without requiring coding [37]. End-to-end AutoML. Hyperparameter tuning is often automated via black-box optimization [11]. But optimizing the loss function of SOTA models by 1% often brings no long-term product gains, whereas tuning decision policy params usually helps, e.g., by better reflecting penalties for false positives/negatives. In our full-stack (extended end-to-end) regime, we enable AutoML for the entire ML pipeline via declarative strategy blueprints (Section 3.3) and an adaptive experiments framework tied to product metrics [9].
Modified on	16/02/2023 08:55

Name	Looper~ An End-to-End ML Platform for Product Decisions Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Machine Learning Model Development from a Software Engineering Perspective~ A Systematic Literature Review
Number of Coding References	9
Number of Codes Coding	4
Coverage	10.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	31/01/2022 14:42
Coded Text	What are the techniques applied in each of phases of the machine learning model development phases Although we did not find any mention of any specific techniques regarding model requirements stages and training stages, we were able to collect several insights about the stages of data processing, future engineering, model evaluation, and model deployment. According to Correia et al. [2] the unique Data Processing method is the use of charts, such as box plots and histograms, to aid with the verification of data quality. Additionally, the main reason for the adoption of such visual tools is to avoid the use of inappropriate data so the data scientist can avoid the risk of increasing development costs by re-execution of Data Processing in case of error identification in later stages like Feature Engineering.
Modified on	07/02/2022 13:22

Name	Machine Learning Model Development from a Software Engineering Perspective~ A Systematic Literature Review
Number of Coding References	9
Number of Codes Coding	4
Coverage	10.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	31/01/2022 14:42
Coded Text	What are the trends regarding the techniques applied in the machine learning model development life cycle? We have identified two distinct trends regarding Software Engineering Model Development life cycle applied to Machine Learning Model Development. The first trend states that the integration of SE development process into Machine Learning modeling must consider the intrinsic differences between Machine Learning based systems and other applications such as data dependency. This pattern is clear in the works presented in [1]–[3]. The second trend states that considering data dependency in machine learning model development leads to different processes/adaptations and partial solutions. Likewise, some articles defend the development of a single general Machine
Modified on	07/02/2022 13:23

Name	Machine Learning Model Development from a Software Engineering Perspective~ A Systematic Literature Review
Number of Coding References	9
Number of Codes Coding	4
Coverage	10.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	31/01/2022 14:42
Coded Text	
Modified on	07/02/2022 13:23

Name	Machine Learning Model Development from a Software Engineering Perspective~ A Systematic Literature Review
Number of Coding References	9
Number of Codes Coding	4
Coverage	10.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	31/01/2022 14:42

Coded Text	A study conducted by Zhang et al. [3] investigated 138 research papers in search of methods for testing and debugging the ML code. Their findings show that only a few contributions focus on testing interpretability, privacy, or efficiency. Zhang et al. [3] focus on exclusively analyzing the stage of Model Evaluation, as opposed to targeting all ML stages. In contrast to the partial information in works such as Amershi et al. [1], authors such as Hesenius et al. [4] argue that although there are challenges faced by software engineers when developing data-driven applications, the data dependency of ML/AI applications does not constitute an issue to the adoption of a common integrated Software Engineering (SE) process, upon which the project's overall success would depend. As a consequence, by defining a set of roles (Software Engineer, Data Scientist, Data Domain Expert, and Domain Expert), stages, and responsibilities to structure the necessary work, decisions and documents, the authors provided a structured engineering process that suits all data-driven applications, ultimately filling the gap found in the literature. It is worth mentioning that although the article presents a general framework in contrast to specific solutions aiming to solve particular and partial issues related to ML/AI modeling, the adoption of the proposed model does
Modified on	07/02/2022 13:20

Name	Machine Learning Model Development from a Software Engineering Perspective~ A Systematic Literature Review
Number of Coding References	9
Number of Codes Coding	4
Coverage	10.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	31/01/2022 14:42
Coded Text	A. RQ1: What are the phases addressed in terms of the machine learning model development? The main purpose is to adapt or integrate Machine Learning framework into Software Development processes' stages, namely: requirements, design, implementation, testing, deployment, and maintenance. In this context, although there are some papers such as Nascimento et al. [9] that have developed Machine Learning Model workflows, the work of Amershi et al. [1] presented the most comprehensive and accepted Machine Learning workflow which was mentioned and used in other articles (such as Correia et al. [2]). The stages addressed in terms of Machine Learning Model Development were: • A Model requirements stage which is related to the agreement between stakeholders and the way the model should work. • Data processing stage which involves data collection, cleaning and labelling (in case of supervised learning). • Feature engineering stage which involves the modification of the selected data. • Model training stage which is related to the way the selected model is trained and tuned on the (labeled) data. • Model evaluation stage which regards to the measurements used in order to evaluate the model. • Model deployment stage which includes deploying, monitoring and maintaining the model.
Modified on	07/02/2022 13:22

Name	Machine Learning Model Development from a Software Engineering Perspective~ A Systematic Literature Review
Number of Coding References	9
Number of Codes Coding	4
Coverage	10.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	31/01/2022 14:42
Coded Text	In this context, we investigate the challenges and practices that emerge during the development of ML models from the software engineering perspective. By focusing our analysis on the well-known stages presented in the Software Engineering development process, we focus on investigating how software developers could benefit from applying or adapting these processes to the ML workflow since one might argue that data scientists would benefit from adopting classical software engineering disciplines (e.g. systems design, quality assurance, and verification) to build their models properly.
Modified on	07/02/2022 13:20

Name	Machine Learning Model Development from a Software Engineering Perspective~ A Systematic Literature Review
Number of Coding References	9
Number of Codes Coding	4
Coverage	10.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	31/01/2022 14:42
Coded Text	RQ3: What are the pros and cons ofeach machine learning model development technique? The issue covered in RQ3 (where there were a significant number of venues with a perspective different from our main research purpose, addressing the question) is more prominent here. In fact, we could only find limited answers in articles from a perspective of Machine Learning Techniques applied to Software Engineering tasks or applied to the stages of the Software Engineering Model Development life-cycle. In other words, we only found limited answers from articles that were mostly unrelated to our research purposes. In this way, the work of Hesenius et al. [4] discusses different Machine Learning Model Techniques from supervised to unsupervised learning, but without specifying the pros and cons of each one.
Modified on	07/02/2022 13:22

Name	Machine Learning Model Development from a Software Engineering Perspective~ A Systematic Literature Review
Number of Coding References	9
Number of Codes Coding	4
Coverage	10.02%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Research gap in Maintaining ML systems
Created on	31/01/2022 14:42
Coded Text	What are the gaps in terms ofthe model development life-cycle? We have identified some gaps in terms of model development life cycle which were mentioned in the literature since the processes adopted by data scientists in their companies are non-linear, requiring too much rework to satisfy customers' needs (Correia et al. [2]). Additionally Correia et al. [2] did not find any activities that address the verification and the validation of the artifacts generated during the workflow stages. As a consequence, the gaps identified in the developing stages were attributed to the existence particularities identified in the Machine Learning model development. According to the authors, practitioners should anticipate problems and save resources in order to mitigate recurrent feedback loops
Modified on	07/02/2022 13:23

Name	Machine Learning Model Development from a Software Engineering Perspective~ A Systematic Literature Review
Number of Coding References	9
Number of Codes Coding	4
Coverage	10.02%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56
Coded Text	In a similar fashion, by presenting methods for measuring the best practices degree of adoption when investigating the relationship between different groups of practices and assessing/predicting their effects by performing regression models, Serban's [5] article reaches conclusions that are in line with Hesenius et al. [4] in a way that there is a set of best practices which is applicable to any ML application development, regardless of the type of data under consideration. Additionally, the author has contributed to the evolution of such practices by presenting a methodology in which each practice is related to its effects and adoption rate. Washizaki et al. [6] presented a related work which clearly complements both works of Hesenius et al. [4] and Serban et al. [5]. The paper addresses the classification of software engineering design patterns by conducting a systematic study which collects, classifies, and analyzes SE architecture and the design of "bad" patterns for ML systems which links traditional software systems and ML systems architecture and design. One interesting result presented by the article is the understanding that SE patterns for ML systems are divided between two processes: The Machine Learning pipeline and SE development. Between these two very distinct approaches, one regarding customized ML workflows related to the specifics of each ML based system and the other oriented towards a broadly general workflow (which would be suitable to any ML application regardless of its specifics), there are several studies with different approaches for the relation/application of software engineering development process to ML modeling. Some of them deal with specifics like the adoption of software engineering best practices related to the development of application programming interfaces (Reimann, Kniesel-W¨unsch [7]), while others address the accountability gap in ML/AI by proposing a framework based on software development best practices (Hutchinson et al. [8]). Further, Nascimento et al. [9] pointed out that the dif- ferences between Traditional systems and Machine Learning systems can be identified by observing the differences between their respective software development activities. In fact, the authors identified that SE activities are more challenging for ML systems which follow specific four-stage software development process, namely: understanding the problem, handling data, building models and monitoring those models. Other works focus on well-known stages of SE lifecycle or the ML Workflow by identifying and addressing many different types of gaps, such as in the works presented in [10]– [16].
Modified on	07/02/2022 13:21

Name	Machine Learning Systems in the IoT~ Trustworthiness Trade-offs for Edge Intelligence
Number of Coding References	9
Number of Codes Coding	1
Coverage	9.17%
Folder Location	Codes\\Tradeoff between Maintainability and Scalability
Parent Name	Codes\\Tradeoff between Maintainability and Scalability\\effect and influence between Maintainbility and scalability
Created on	01/06/2022 13:41
Coded Text	At present, however, machine learning systems assume homogeneous and context-independent cloud computing infrastructure with scalable data processing and storage, uninterrupted and unrestricted power supply, and low latency and high bandwidth networks. This stable and consistent environment does not exist for the billions of connected devices in the IoT, where data offloading to wireless networks, distributed, heterogeneous computing infrastructure and resource-constrained devices with physical hardware limitations present trade-offs against each other and the performance of algorithms. To achieve scale, components in the IoT must also be reusable.
Modified on	28/02/2023 11:56

Name	Machine Learning Systems in the IoT~ Trustworthiness Trade-offs for Edge Intelligence
Number of Coding References	9
Number of Codes Coding	1
Coverage	9.17%
Folder Location	Codes\\Tradeoff between Maintainability and Scalability
Parent Name	Codes\\Tradeoff between Maintainability and Scalability\\effect and influence between Maintainbility and scalability
Created on	01/06/2022 13:41
Coded Text	Classical machine learning algorithms are developed under static, benign, closed-world assumptions: they assume that the world does not change, that the environment is good-natured [17], and that all categories to be predicted were known during training and contained in the training data [18]. Obviously this does not correspond with reality. For example, in medical image classification, it has been established that training data can contain unrecognised categories that are not in the labels but that affect predictive outcomes [19]. While adversarial machine learning [20] can be used to improve the robustness of models under the malicious attack of an adversary, and lifelong learning [21] provides methods for continuous learning by accumulating and maintaining knowledge which can be used to improve future learning, the conditions under which different paradigms can be combined, and what vulnerabilities this may result in, are not obvious. Despite the success of machine learning algorithms, many challenges thus remain to train models that generalize well and have good predictive accuracy while also being resource-efficient, robust, and adaptive in new, real-world environments
Modified on	28/02/2023 11:54

Name	Machine Learning Systems in the IoT~ Trustworthiness Trade-offs for Edge Intelligence
Number of Coding References	9
Number of Codes Coding	1
Coverage	9.17%
Folder Location	Codes\\Tradeoff between Maintainability and Scalability
Parent Name	Codes\\Tradeoff between Maintainability and Scalability\\effect and influence between Maintainbility and scalability
Created on	01/06/2022 13:41
Coded Text	In addition to the resources consumed during training, supervised machine learning requires labelled training data, which can be expensive and time consuming to collect [13]. When this is not possible, unsupervised learning which requires no ground truth labels, weak supervision with automated label generation [13], and approaches that reduce the amount of labels required [14] can be considered. Generally these present trade-offs against predictive performance. Due to the infrastructure requirements and cost of model training and data labelling, many applications download pre-trained models from online repositories, which can sometimes be used off-the-shelf, or otherwise adapted to new domains or datasets with transfer learning [15]. While pre-trained models speed up the development of new applications, they present significant security risks [16].
Modified on	28/02/2023 11:53

Name	Machine Learning Systems in the IoT~ Trustworthiness Trade-offs for Edge Intelligence
Number of Coding References	9
Number of Codes Coding	1
Coverage	9.17%
Folder Location	Codes\\Tradeoff between Maintainability and Scalability
Parent Name	Codes\\Tradeoff between Maintainability and Scalability\\effect and influence between Maintainbility and scalability
Created on	01/06/2022 13:41
Coded Text	Network interruptions are bound to affect IoT applications. At worst, machine learning systems must consider the risk of completely loosing connectivity during training or inference, making fault tolerance a necessary consideration [28]. At best, wireless connections introduce latency, variability, uncertainty and costs to machine learning systems, which historically have abstracted away their iterative communication requirements. Offloading thus weighs against privacy and real-time inference requirements, and constrains the frequency, size and data distribution of training updates of machine learning systems. While the data path, timing and transfer volumes can be optimized through routing schemes, scheduling and data compression to minimize bottlenecks and communication costs [35], this can reduce predictive accuracy and may be limited by the power supply and computing capabilities of devices [32].
Modified on	28/02/2023 11:56

Name	Machine Learning Systems in the IoT~ Trustworthiness Trade-offs for Edge Intelligence
Number of Coding References	9
Number of Codes Coding	1
Coverage	9.17%
Folder Location	Codes\\Tradeoff between Maintainability and Scalability
Parent Name	Codes\\Tradeoff between Maintainability and Scalability\\effect and influence between Maintainbility and scalability
Created on	01/06/2022 13:41
Coded Text	Practically, the training process determines the values of those model parameters that minimize the error between a predicted value and its corresponding real value in the training data. In addition to parameter coefficients, a model can also have hyperparameters that control its complexity. To find the best model, a range of different model types, parameters and hyperparameters must be explored so that the best model can be selected. However, exploring each of these choices requires computational power, time and energy, resulting in trade-offs between predictive performance and resource consumption. State-of-the-art machine learning models, in particular deep neural networks, can have millions of parameters. Training them takes weeks or even months, and the computing and energy resources required are substantial. Two important approaches for improving the performance of machine learning systems are designing them together with specialized hardware and distributing model training for massive, parallel deployment across cloud servers [12].
Modified on	28/02/2023 11:52

Name	Machine Learning Systems in the IoT~ Trustworthiness Trade-offs for Edge Intelligence
Number of Coding References	9
Number of Codes Coding	1
Coverage	9.17%
Folder Location	Codes\\Tradeoff between Maintainability and Scalability
Parent Name	Codes\\Tradeoff between Maintainability and Scalability\\effect and influence between Maintainbility and scalability
Created on	01/06/2022 13:41
Coded Text	The code responsible for model training and inference is only a small component of the greater system, which includes components for configuration, data collection, data verification, feature extraction, machine resource management, analysis, process management, serving infrastructure and monitoring. Even though machine learning systems are constructed from these different components, models are not modular in the way that software is [25]. Model parameters are learned iteratively, and as dependent on the data distribution as on the features used for training and the hyperparameters. Due to these dependencies, individual models are not extensible and multiple models interact in non-obvious ways. However, models evolve as data changes, methods improve or software dependencies change [30]. Ongoing deployment, customisation, reuse and tracking are thus continuous challenges. Machine learning systems require end-to-end software support that facilitates the development, testing, configuration, deployment, management and maintenance of all components that affect data provenance, model training and inference [31].
Modified on	28/02/2023 11:55

Name	Machine Learning Systems in the IoT~ Trustworthiness Trade-offs for Edge Intelligence
Number of Coding References	9
Number of Codes Coding	1
Coverage	9.17%
Folder Location	Codes\\Tradeoff between Maintainability and Scalability
Parent Name	Codes\\Tradeoff between Maintainability and Scalability\\effect and influence between Maintainbility and scalability
Created on	01/06/2022 13:41
Coded Text	The edge varies in computing capabilities and connectivity from sensing and actuator devices that observe and control the environment at the lowest level, to gateways and cloud servers. Data processing, model training and inference on the edge can be device, gateway or cloud-centric [32]. Device-centric approaches reduce offloading challenges, but processing is limited by the computing capabilities and power supply of devices. Gateway-centric computation requires wireless communication, and introduces associated variability and uncertainty. Cloud-centric approaches offer unlimited storage and data processing capabilities, but come with copious communication overheads. Edge servers present an intermediate solution that offers stable power supply and processing closer to the points of data collection, while reducing the data transfer requirements that would be required by the cloud. A simple heuristic is that the availability of data processing, memory, storage and communication overheads all rise with increasing distance from devices. Increasing the former is desirable, while increasing communication overheads is not. The key challenge of distributing machine learning systems in the IoT is to decide whether, when and how to offload computations; that is, to find the optimal balance between local processing and computation offloading given unpredictable networks, and constrained and diverse devices and servers.
Modified on	28/02/2023 11:56
Name	Machine Learning Systems in the IoT~ Trustworthiness Trade-offs for Edge Intelligence
Number of Coding References	9
Number of Codes Coding	1
Coverage	9.17%
Folder Location	Codes\\Tradeoff between Maintainability and Scalability
Parent Name	Codes\\Tradeoff between Maintainability and Scalability\\effect and influence between Maintainbility and scalability
Created on	01/06/2022 13:41
Coded Text	The quality of a machine learning model is strongly influ- enced by the quality and underlying distribution of the data that was used to train it [22]. Due to this central role of data in machine learning, common features of raw observational data, like missing values, data redundancy and noise, significantly impact the performance of the model that is trained. Noise, for example, obscures the data signal and can result from random or systematic errors in the observations, or from data that has been tampered with. Model performance can be degraded further by propagating data errors that were generated during data processing through the entire machine learning workflow. To extend a software metaphor, such data errors are to machine learning systems what bugs are to code [23]. Once deployed, data discovery and management are a particular challenge. Datasets are often taken from different sources. As projects grow, so do dependencies between datasets. Over time the training data becomes increasingly complex to track and version [24]. Data dependencies and feedback loops are often hidden and can have unexpected effects that make machine learning systems brittle and error diagnosis expensive [25]. Machine learning systems are also vulnerable to attacks that exploit their dependency on data by polluting training data
Modified on	28/02/2023 11:55

Name	Machine Learning Systems in the IoT~ Trustworthiness Trade-offs for Edge Intelligence
Number of Coding References	9
Number of Codes Coding	1
Coverage	9.17%
Folder Location	Codes\\Tradeoff between Maintainability and Scalability
Parent Name	Codes\\Tradeoff between Maintainability and Scalability\\effect and influence between Maintainbility and scalability
Created on	01/06/2022 13:41
Coded Text	Trained models infer output values for new data inputs to inform decisions or take actions on an ongoing basis. A trained model is a reusable asset that will make thousands or even millions of predictions before it is retrained. Unlike model training which happens in the background of an application, inference usually serves users directly and consequently needs to be efficient, reliable and interpretable. Even though the resource requirements for a single prediction are negligible in comparison to those of training a model, the scale at which inference happens requires efficient and optimized processes with high throughput, low latency and graceful performance degradation [26]. Traditionally, more attention has been devoted to optimizing the training process rather than inference. Recent releases of popular machine learning platforms like TensorFlow and MXnet now offer libraries for model optimization, but efficiency alone is not enough. When machine learning systems make decisions and act on our behalf, inference must also be reliable [27]. Current machine learning systems do not offer predictable throughput, latency and accuracy. Methods that guarantee model outputs and offer reliable uncertainty estimates are needed to provide inference with quality assurance [28]. Additionally, interpretable inference, which can be likened to the ability of humans to understand how a model works, is necessary for trusted, fair and ethical decision-making based on predictions [29].
Modified on	28/02/2023 11:55

Name	Machine Learning Systems in the IoT~ Trustworthiness Trade-offs for Edge Intelligence Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	
Modified on	07/02/2022 13:01

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	
Modified on	07/02/2022 13:02

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	
Modified on	07/02/2022 13:04

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	
Modified on	07/02/2022 13:07

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	For example, DeepXplore [1], a differential white-box T testing technique for deep learning, revealed thousands of incorrect corner case behaviours in autonomous driving learning systems; Themis [5], a fairness testing technique for detecting causal discrimination, detected significant ML model discrimination towards gender, marital status, or race for as many as 77.2 percent of the individuals in datasets to which it was applied.
Modified on	07/02/2022 12:59

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Open-Source Tool Support in ML Testing There are several tools specially designed for ML testing. Angell et al. presented Themis [213], an open-source tool for testing group discrimination.14 There is also an ML testing framework for tensorflow, named mltest,15 for writing simple ML unit tests. Similar to mltest, there is a testing framework for writing unit tests for pytorch-based ML systems, named torchtest.16 Dolby et al. [237] extended WALA to enable static analysis for machine learning code using TensorFlow.
Modified on	22/06/2022 11:19

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Software Testing versus ML Testing Traditional software testing and ML testing are different in many aspects. To understand the unique features of ML testing, we summarise the primary differences between traditional software testing and ML testing in Table 1. 1) Component to test (where the bug may exist): traditional software testing detects bugs in the code, while ML testing detects bugs in the data, the learning program, and the framework, each of which play an essential role in building an ML model. 2) Behaviours under test: the behaviours of traditional software code are usually fixed once the requirement is fixed, while the behaviours of an ML model may frequently change as the training data is updated. 3) Test input: the test inputs in traditional software testing are usually the input data when testing code; in ML testing, however, the test inputs in may have more diverse forms. Note that we separate the definition of 'test input' and 'test data'. In particular, we use 'test input' to refer to the inputs in any form that can be adopted to conduct machine learning testing; while 'test data' specially refers to the data used to validate ML model behaviour (see more in Section 2). Thus, test inputs in ML testing could be, but are not limited to, test data. When testing the learning program, a test case may be a single test instance from
Modified on	22/06/2022 11:12

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	the test data or a toy training set; when testing the data, the test input could be a learning program. 4) Test oracle: traditional software testing usually assumes the presence of a test oracle. The output can be verified against the expected values by the developer, and thus the oracle is usually determined beforehand. Machine learning, however, is used to generate answers based on a set of input values after being deployed online. The correctness of the large number of generated answers is typically manuallyconfirmed. Currently, the identification of test oracles remains challenging, because many desired properties are difficult to formally specify. Even for a concrete domain specific problem, the oracle identification is still time-consuming and labour-intensive, because domain-specific knowledge is often required. In current practices, companies usually rely on third-party data labelling companies to get manual labels, which can be expensive.Metamorphic relations [71] are a type of pseudo oracle adopted to automatically mitigate the oracle problemin machine learning testing. 5) Test adequacy criteria: test adequacy criteria are used to provide quantitative measurement on the degree of the target software that has been tested. Up to present, many adequacy criteria are proposed and widely adopted in industry, e.g., line coverage, branch coverage, dataflow coverage. However, due to fundamental differences of programming paradigm and logic representation format for machine learning software and traditional software, new test adequacy criteria are required to take the characteristics of machine learning software into consideration. 6) 7) False positives in detected bugs: due to the difficulty in obtaining reliable oracles, ML testing tends to yield more false positives in the reported bugs. Roles of testers: the bugs in ML testing may exist not only in the learning program, but also in the data or the algorithm, and thus data scientists or algorithm designers could also play the role of testers.
Modified on	22/06/2022 11:12

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	
Modified on	07/02/2022 13:01

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	
Modified on	07/02/2022 13:02

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	
Modified on	07/02/2022 13:04

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	3.2.3 Online Testing Offline testing tests the model with historical data without in the real application environment. It also lacks the data collection process of user behaviours. Online testing complements the shortage of offline testing, and aims to detect bugs after the model is deployed online.
Modified on	07/02/2022 13:03

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	5.1 Test Input Generation
Modified on	07/02/2022 13:05

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	5.2 Test Oracle
Modified on	07/02/2022 13:05

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	5.3 Test Adequacy
Modified on	07/02/2022 13:05

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	5.4 Test Prioritisation and Reduction
Modified on	07/02/2022 13:05

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	5.5 Bug Report Analysis
Modified on	07/02/2022 13:05

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	5.6 Debug and Repair
Modified on	07/02/2022 13:05

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	5.7 General Testing Framework and Tools
Modified on	07/02/2022 13:05

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	6.3 Robustness and Security
Modified on	07/02/2022 13:06

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	6.4 Efficiency
Modified on	07/02/2022 13:06

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	6.5 Fairness
Modified on	07/02/2022 13:06

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	6.6 Interpretability
Modified on	07/02/2022 13:06

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	6.7 Privacy
Modified on	07/02/2022 13:06

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	6ML PROPERTIES TO BE TESTED
Modified on	07/02/2022 13:05

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	Challenges in ML Testing As this survey reveals, ML testing has experienced rapid recent growth. Nevertheless, ML testing remains at an early stage in its development, with many challenges and open questions lying ahead. Challenges in Test Input Generation. Although a range of test input generation techniques have been proposed (see more in Section 5.1), test input generation remains challenging because of the large behaviour space of ML models.
Modified on	07/02/2022 13:08
Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	In this paper, we use the term ‘Machine Learning Testing’ (ML testing) to refer to any activity aimed at detecting differences between existing and required behaviours of machine learning systems. ML testing is different from testing approaches that use machine learning or those that are guided by machine learning, which should be referred to as ‘machine learning-based testing’. This nomenclature accords with previous usages in the software engineering literature. For example, the literature uses the terms ‘state-based testing’ [16] and ‘search-based testing’ [17], [18] to refer to testing techniques that make use of concepts of state and search space, whereas we use the terms ‘GUI testing’ [19] and ‘unit testing’ [20] to refer to test techniques that tackle challenges of testing Graphical User Interfaces (GUIs) and code units.
Modified on	07/02/2022 13:01

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	Machine learning testing poses challenges that arise from the fundamentally different nature and construction of machine learning systems, compared to traditional (relatively more deterministic and less statistically-orientated) software systems. For instance, a machine learning system inherently follows a data-driven programming paradigm, where the decision logic is obtained via a training procedure from training data under the machine learning algorithm's architecture [8]. The model's behaviour may evolve over time, in response to the frequent provision of new data [8]. While this is also true of traditional software systems, the core underlying behaviour of a traditional system does not typically change in response to new data, in the way that a machine learning system can. Testing machine learning also suffers from a particularly pernicious instance of the Oracle Problem [9]. Machine learning systems are difficult to test because they are designed to provide an answer to a question for which no previous answer exists [10]. As Davis and Weyuker said [11], for these kinds of systems 'There would be no need to write such programs, if the correct answer were known'. Much of the literature on testing machine learning systems seeks to find techniques that can tackle the Oracle problem, often drawing on traditional software testing approaches.
Modified on	07/02/2022 13:00

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	ML Testing Properties Testing properties refer to what to test in ML testing: for what conditions ML testing needs to guarantee for a trained model. This section lists some typical properties that the literature has considered. We classified them into basic functional requirements (i.e., correctness and model relevance) and non-functional requirements (i.e., efficiency, robustness,3 fairness, interpretability). These properties are not strictly independent of each other when considering the root causes, yet they are different external manifestations of the behaviours of an ML system and deserve being treated independently in ML testing.
Modified on	07/02/2022 13:03

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	Offline Testing The workflow of offline testing is shown by the top dotted rectangle of Fig. 5. At the very beginning, developers need to conduct requirement analysis to define the expectations of the users for the machine learning system under test. In requirement analysis, specifications of a machine learning system are analysed and the whole testing procedure is planned.
Modified on	07/02/2022 13:03

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	Role ofTesting in ML Development Fig. 4 shows the life cycle of deploying a machine learning system with ML testing activities involved. At the very beginning, a prototype model is generated based on historical data; before deploying the model online, one needs to conduct offline testing, such as cross-validation, to make sure that the model meets the required conditions. After deployment, the model makes predictions, yielding new data that can be analysed via online testing to evaluate how the model interacts with user behaviours. There are several reasons that make online testing essential. First, offline testing usually relies on test data, while test data usually fails to fully represent future data [42]; Second, offline testing is not able to test some circumstances that may be problematic in real applied scenarios, such as data loss and call delays. In addition, offline testing has no access to some business metrics such as open rate, reading time, and click-through rate.
Modified on	07/02/2022 13:02

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	Safety-critical applications such as self-driving systems [1], [2] and medical treatments [3], increase the importance of behaviour relating to correctness, robustness, privacy, efficiency and fairness. Software testing refers to any activity that aims to detect the differences between existing and required behaviour [4]. With the recent rapid rise in interest and activity, testing has been demonstrated to be an effective way to expose problems and potentially facilitate to improve the trustworthiness of machine learning systems.
Modified on	07/02/2022 12:59

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	The behaviours of interest for machine learning systems are also typified by emergent properties, the effects of which can only be fully understood by considering the machine learning system as a whole. This makes testing harder, because it is less obvious how to break the system into smaller components that can be tested, as units, in isolation. From a testing point of view, this emergent behaviour has a tendency to migrate testing challenges from the unit level to the integration and system level. For example, low accuracy/ precision of a machine learning model is typically a composite effect, arising from a combination of the behaviours of different components such as the training data, the learning program, and even the learning framework/library [8]. Errors may propagate to become amplified or suppressed, inhibiting the tester’s ability to decide where the fault lies.
Modified on	07/02/2022 13:00

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	This section organises ML testing research based on the test- ing workflow as shown by Fig. 5. ML testing includes offline testing and online testing. Albarghouthi and Vinitsky [75] developed a fairness specification language that can be used for the development of runtime monitoring, in detecting fairness issues. Such a kind of run-time monitoring belongs to the area of online testing. Nevertheless, current research mainly centres on offline testing as introduced below. The procedures that are not covered based on our paper collection, such as requirement analysis and regression testing and those belonging to online testing are discussed as research opportunities in Section 10.
Modified on	07/02/2022 13:04

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Research gap in Maintaining ML systems
Created on	07/02/2022 10:23
Coded Text	There remain many research opportunities in ML testing. These are not necessarily research challenges, but may greatly benefit machine learning developers and users as well as the whole research community. Testing More Application Scenarios. Much current research focuses on supervised learning, in particular classification problems. More research is needed on problems associated with testing unsupervised and reinforcement learning. The testing tasks currently tackled in the literature, pri- marily centre on image classification. There remain open exciting testing research opportunities in many other areas, such as speech recognition, natural language processing and agent/game play. Testing More ML Categories and Tasks. We observed pro- nounced imbalance regarding the coverage of testing techniques for different machine learning categories and tasks, as demonstrated by Table 4. There are both challenges and research opportunities for testing unsupervised and reinforcement learning systems. For instance, transfer learning, a topic gaining much recent interest, focuses on storing knowledge gained while solving one problem and applying it to a different but related
Modified on	07/02/2022 13:08

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56
Coded Text	In 2009, Xie et al. [118] also applied metamorphic testing on supervised learning applications. Fairness testing was proposed in 2012 by Dwork et al. [138]; the problem of interpretability was proposed in 2016 by Burrell [252]. In 2017, Pei et al. [1] published the first white-box testing paper on deep learning systems. Their work pioneered to propose coverage criteria for DNN. Enlightened by this
Modified on	07/02/2022 13:08

Name	Machine Learning Testing~ Survey, Landscapes and Horizons
Number of Coding References	37
Number of Codes Coding	4
Coverage	4.31%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56

Coded Text	paper, a number of machine learning testing techniques have emerged, such as DeepTest [76], DeepGauge [92], DeepConcolic [111], and DeepRoad [79]. A number of software testing techniques has been applied to ML testing, such as different testing coverage criteria [76], [92], [145], mutation testing [152], combinatorial testing [149], metamorphic testing [100], and fuzz testing [89].
Modified on	07/02/2022 13:08

Name	MLadder~ An Online Training System for Machine Learning and Data Science Education
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	MLadder~ An Online Training System for Machine Learning and Data Science Education Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	MLCask~ Efficient Management of Component Evolution in Collaborative Data Analytics Pipelines
Number of Coding References	17
Number of Codes Coding	4
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	
Modified on	07/02/2022 11:43

Name	MLCask~ Efficient Management of Component Evolution in Collaborative Data Analytics Pipelines
Number of Coding References	17
Number of Codes Coding	4
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	
Modified on	07/02/2022 11:43

Name	MLCask~ Efficient Management of Component Evolution in Collaborative Data Analytics Pipelines
Number of Coding References	17
Number of Codes Coding	4
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	In this paper, we identify two main challenges that arise during the deployment of machine learning pipelines, and address them with the design of versioning for an end-to-end analytics system MLCask. The system supports multiple user roles with the ability to perform Git-like branching and merging operations in the context of the machine learning pipelines. We define and accelerate the metric-driven merge operation by pruning the pipeline search tree using reusable history records and pipeline compatibility information. Further, we design and implement the prioritized pipeline search, which gives preference to the pipelines that probably yield better performance. The effectiveness of MLCask is evaluated through an extensive study over several real-world deployment cases. The performance evaluation shows that the proposed merge operation is up to 7.8x faster and saves up to 11.9x storage space than the baseline method that does not utilize history records.
Modified on	07/02/2022 11:36

Name	MLCask~ Efficient Management of Component Evolution in Collaborative Data Analytics Pipelines
Number of Coding References	17
Number of Codes Coding	4
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	In this section, we share our experience on the deploy- ment of MLCask at National University Hospital6(NUH). We have been working with NUHS7 since 2010 on data cleaning, data integration, modeling and predictive analytics for various diseases [10], [24], [25], as a collaboration to develop solutions for existing and emerging healthcare needs. Due to the sensitivity of the data and the critical nature of healthcare applications, hospitals must manage the database and model development for accountability and verifiability purposes. MLCask has been designed towards fulfilling such requirements. In deployment, the production pipeline has to be separated from the development pipeline. The production pipeline is a stable version that should not be modified when it is in service, unless minor bug fixes are required. For development purposes, we form a branch with a replica of the pipeline as a development pipeline. For upgrading of the production pipeline, we can merge the development pipeline into the production pipeline. To facilitate such development and upgrading, MLCask provides branching functionality for the pipelines. In a large hospital such as NUH, different data scientist teams and clinicians may develop models of the same pipeline concurrently. The scenario is similar to what has been depicted in Fig. 3 and explained in Section V, where different users are updating different components of the same pipeline at the same time. This could lead to a number of updated pipelines that are difficult to be merged together. As explained in Section V, using a na¨ıve strategy to select the latest components could lead to incompatibility and sub-optimal pipeline issues. To this end, MLCask supports pipeline merging optimization to derive a more effective pipeline.
Modified on	07/02/2022 11:44

Name	MLCask~ Efficient Management of Component Evolution in Collaborative Data Analytics Pipelines
Number of Coding References	17
Number of Codes Coding	4
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	OPTIMIZING MERGE OPERATIONS In this section, we present optimizations to improve the efficiency of the merge operations in MLCask. The nontriviality of the merge operation lies in the huge search space for the optimal pipeline and how to exclude the incompatible pipelines. For a pipeline with Nf components, the upper bound of the number of the possible pipeline candidates is given by i=1 N(S(fi)), where N(S(fi)) denotes the number of elements in set S(fi). Therefore, the number of pipeline Nf
Modified on	07/02/2022 11:44

Name	MLCask~ Efficient Management of Component Evolution in Collaborative Data Analytics Pipelines
Number of Coding References	17
Number of Codes Coding	4
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	Pruning Merge Tree using Component Compatibility Information When the schema of a pipeline component changes, its succeeding components have to be updated accordingly. By leveraging the constraints on component compatibility, we can avoid enumerating the pipelines that are destined to fail in execution. We continue to use the version history as illustrated in Fig. 3 and its corresponding pipeline search tree in Fig. 4 to exemplify the idea and show the compatibility information. The succeeding components of feature extraction can be divided into two sets based on compatibility: • {<CNN, 0.0>, <CNN, 0.1>, <CNN, 0.4>} following <feature_extract, 0.0>; • {<CNN, 0.2>, <CNN, 0.3>} following <feature_extract, 1.0>;
Modified on	07/02/2022 11:44

Name	MLCask~ Efficient Management of Component Evolution in Collaborative Data Analytics Pipelines
Number of Coding References	17
Number of Codes Coding	4
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	SUPPORTING NON-LINEAR VERSION CONTROL We use the pipeline shown in Fig. 2 to illustrate how MLCask achieves branch and merge operations to support non-linear version history. The example pipeline fetches data from a hospital dataset, followed by data cleansing and feature extraction, and eventually feeds the extracted data into a CNN model to predict how likely a specific patient will be readmitted in 30 days.
Modified on	07/02/2022 11:44

Name	MLCask~ Efficient Management of Component Evolution in Collaborative Data Analytics Pipelines
Number of Coding References	17
Number of Codes Coding	4
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	The main contributions of this paper can be summarized as follows: • We identify two key challenges of managing asynchronous activities between agile development of analytics components and retrospective analysis. Understanding these challenges provides the insights for efficiently managing the versioning of ML pipelines. • We present the design of an efficient system MLCask, with the support of non-linear version control semantics in the context of ML pipelines. MLCask can ride upon most of the mainstream ML platforms to manage component evolution in collaborative ML pipelines via branching and merging. • We propose two search tree pruning methods in MLCask to reduce the candidate pipeline search space in order to improve system efficiency under the non-linear version control semantics. We further provide a prioritized pipeline search strategy in MLCask that looks for promising but suboptimal pipelines with a given time constraint. • We have fully implemented MLCask for deployment in a local hospital. Experimental results on diverse realworld ML pipelines demonstrate MLCask achieves better performance than baseline systems, ModelDB [18] and MLflow [22], in terms of storage efficiency and computation reduction.
Modified on	07/02/2022 11:42

Name	MLCask~ Efficient Management of Component Evolution in Collaborative Data Analytics Pipelines
Number of Coding References	17
Number of Codes Coding	4
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	To address the aforementioned challenges, version control semantics [7], [12], [15], [19] need to be introduced to the ML pipeline. Current pipeline management systems either do not explicitly consider the version evolution, or handle versioning by merely archiving different versions into distinctive disk folders so that different versions will not conflict with or overwrite each other. The latter approach not only incurs huge storage and computation overhead, but also fails to describe the logical relationship between different versions. In this paper, we first elaborate on the common challenges in data analytics applications and formulate version control semantics in the context of ML pipeline management. We then present a design of Git-like end-to-end ML life-cycle management system, called MLCask, and its version control support. MLCask facilitates collaborative component updates in ML pipelines, where components refer to the computational units in the pipeline such as data ingestion methods, pre-processing methods, and models. The key idea of MLCask is to keep track of the evolution of pipeline components together with the inputs, execution context, outputs, and the corresponding performance statistics. By introducing the non-linear version control semantics [7], [12], [19] to the context of ML pipelines, MLCask can achieve full historical information traceability with the support of branching and merging. Further, we propose two methods in MLCask to prune the pipeline search tree and reuse materialized intermediate results to reduce the time needed for the metric-driven merge operation. Lastly, to minimize the cost of the merge operation for divergent ML pipeline versions, we devise multiple strategies in MLCask that
Modified on	07/02/2022 11:41

Name	MLCask~ Efficient Management of Component Evolution in Collaborative Data Analytics Pipelines
Number of Coding References	17
Number of Codes Coding	4
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	Version Control for Pipeline Components A semantic version4 in MLCask is represented by an identifier: branch@schema.increment, where branch represents the Git-like branch semantics, schema denotes the output data schema, and increment represents the minor incremental changes that do not affect the output data schema. We use the notation: <feature_extract, master@0.1> to denote a component named feature_extract and its corresponding semantic version. The representation indicates that the component has received one incremental update and there is no output data schema update yet. For components on its master branch, we simplify the representation to the following form: <feature_extract, 0.1>.
Modified on	07/02/2022 11:43

Name	MLCask~ Efficient Management of Component Evolution in Collaborative Data Analytics Pipelines
Number of Coding References	17
Number of Codes Coding	4
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	To address the aforementioned challenges, version control semantics [7], [12], [15], [19] need to be introduced to the ML pipeline. Current pipeline management systems either do not explicitly consider the version evolution, or handle versioning by merely archiving different versions into distinctive disk folders so that different versions will not conflict with or overwrite each other. The latter approach not only incurs huge storage and computation overhead, but also fails to describe the logical relationship between different versions. In this paper, we first elaborate on the common challenges in data analytics applications and formulate version control semantics in the context of ML pipeline management. We then present a design of Git-like end-to-end ML life-cycle management system, called MLCask, and its version control support. MLCask facilitates collaborative component updates in ML pipelines, where components refer to the computational units in the pipeline such as data ingestion methods, pre-processing methods, and models. The key idea of MLCask is to keep track of the evolution of pipeline components together with the inputs, execution context, outputs, and the corresponding performance statistics. By introducing the non-linear version control semantics [7], [12], [19] to the context of ML pipelines, MLCask can achieve full historical information traceability with the support of branching and merging. Further, we propose two methods in MLCask to prune the pipeline search tree and reuse materialized intermediate results to reduce the time needed for the metric-driven merge operation. Lastly, to minimize the cost of the merge operation for divergent ML pipeline versions, we devise multiple strategies in MLCask that
Modified on	07/02/2022 11:41

Name	MLCask~ Efficient Management of Component Evolution in Collaborative Data Analytics Pipelines
Number of Coding References	17
Number of Codes Coding	4
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications\Architecture ML system
Created on	03/02/2022 14:39
Coded Text	(C1) Frequent retraining. Many real-world data analytics applications require frequent retraining since concept drift is a common phenomenon. For instance, in the computer cluster of NUHS1 hospital, there are around 800 to 1200 inpatients at any given time and the number of newly admitted patients each
Modified on	07/02/2022 11:42

Name	MLCask~ Efficient Management of Component Evolution in Collaborative Data Analytics Pipelines
Number of Coding References	17
Number of Codes Coding	4
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications\Architecture ML system
Created on	03/02/2022 14:39
Coded Text	(C2) Asynchronous pipeline component update and merge. As expected for collaborative analytics, concurrent updates of a pipeline introduce both consistency and maintenance issues. First, the asynchronous component update by different users may cause the potential failure of the entire pipeline when two incompatible updated components are combined. Second, we should consider the fundamental difference between software engineering and building ML pipelines: ML pipeline development is metric-driven, rather than featuredriven. For building ML pipelines, data scientists typically pursue pipeline performance, and different branches are used for iterative trials. They often create different branches for iterative trials to improve individual components of the pipeline. In contrast, software engineers merge two branches because the features developed on the merging branches are needed.
Modified on	07/02/2022 11:42

Name	MLCask~ Efficient Management of Component Evolution in Collaborative Data Analytics Pipelines
Number of Coding References	17
Number of Codes Coding	4
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications\Architecture ML system
Created on	03/02/2022 14:39
Coded Text	In a collaborative environment, the changes and updates due to pipeline evolution often cause cumbersome coordination and maintenance work, raising the costs and making it hard to use. Existing solutions, unfortunately, do not address the version evolution problem, especially in a collaborative environment where non-linear version control semantics are necessary to isolate operations made by different user roles. The lack of version control semantics also incurs unnecessary storage consumption and lowers efficiency due to data duplication and repeated data pre-processing, which are avoidable.
Modified on	07/02/2022 11:36

Name	MLCask~ Efficient Management of Component Evolution in Collaborative Data Analytics Pipelines
Number of Coding References	17
Number of Codes Coding	4
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications\Architecture ML system
Created on	03/02/2022 14:39
Coded Text	In many real-world machine learning (ML) applications, new data is continuously fed to the ML pipeline. Consequently, iterative updates and retraining of the analytics components become essential, especially for applications that exhibit significant concept drift behavior where the trained model becomes inaccurate as time passes. Consider healthcare applications [3], [24] as an example in which hospital data is fed to data analytics pipelines [5], [11] on a daily basis for various medical diagnosis predictions. The extracted data schema, pre-processing steps, analytics models are highly
Modified on	07/02/2022 11:39

Name	MLCask~ Efficient Management of Component Evolution in Collaborative Data Analytics Pipelines
Number of Coding References	17
Number of Codes Coding	4
Coverage	11.14%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications\Architecture ML system
Created on	03/02/2022 14:39
Coded Text	volatile [9], [25] due to the evolution of the dataset, leading to a series of challenges. First, to ensure quality satisfaction of the analytics models, the pipeline needs to be retrained frequently to adapt to the changes, which costs a lot of storage and time [1], [15], [20]. Second, the lengthy pipeline and computer cluster environment cause the asynchronous pipeline update problem, because different components may be developed and maintained by different users. Third, the demand for retrospective research on models and data from different time periods further complicates the management of massive pipeline versions.
Modified on	07/02/2022 11:39

Name	MLCask~ Efficient Management of Component Evolution in Collaborative Data Analytics Pipelines
Number of Coding References	17
Number of Codes Coding	4
Coverage	11.14%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56
Coded Text	Versioning for Datasets and Source Code. State-of-the-art systems for managing datasets versioning such as Forkbase [19], OrpheusDB [7], and Decibel [12] support Git-like semantics on datasets to enable collaborative analysis as well as efficient query processing. In terms of versioning code of preprocessing methods and models, the file-based Git is widely used. They store source code in repositories and manage versions of the code based on the text information. However, these methods are not suitable for managing the versioning of the data analytics pipeline. Compared with dataset versioning, pipeline versioning requires not only dataset versioning but also the versioning of the source code. Furthermore, in contrast to Git, pipeline versioning needs to take care of the evolution of the whole pipeline, which comprises the source code, the datasets, and the relationship between pipeline components. Build Automation Tools. In terms of maintaining the relationships between pipeline components, build automation tools such as Maven8, Gradle9 and Ant10 manage the dependency between different software packages to facilitate the project development. In comparison, MLCask has a quite different objective: pipeline versioning organizes various subsystems to form an end-to-end data analytics pipeline instead of compiling a project. Further, pipeline versioning requires explicit data-flow management to enable the saving or reusing of the intermediate outputs for exploration, which is not an objective of the build automation tools. Data Version Control (DVC). DVC11 is a system built upon Git, which supports non-linear version history of pipelines, and also records the performance of the pipelines. Unfortunately, it inherits the merge mechanism from Git, which treats merge operation as combining the latest features. Machine Learning Pipeline Management. In ML pipeline management, MLlib [13] simplifies the development of ML pipelines by introducing the concepts of DataFrame, Transformer, and Estimator. SHERLOCK [17] enables users to store, index, track, and explore different pipelines to support ease of use, while Velox [2] focuses on online management, maintenance, and serving of the ML pipelines. Nevertheless, version control semantics of the pipelines are not supported by the aforementioned methods. The pipeline management system that is most similar to MLCask is proposed in [15]. In this work, versioning is proposed to maintain multiple versions of an end-to-end ML pipeline. It archives different versions of data into distinctive disk folders, which may lead to difficulty in tracing the version
Modified on	07/02/2022 11:45

Name	ML-Defense~ Machine Learning for building Dependable Federated Network System
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	ML-Defense~ Machine Learning for building Dependable Federated Network System Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	MLOps Challenges in Multi-Organization Setup~ Experiences from Two Real-World Cases
Number of Coding References	11
Number of Codes Coding	2
Coverage	14.91%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 10:15
Coded Text	To improve, we need integration mechanisms for ML/AI, analogous to integration patterns in information systems [10] but applicable at the level of AI/ML features, to create multiorganization AI/ML systems. Like with information systems, there are several challenges that need to be tackled, including integration interfaces, scaling, privacy, governance, and so on. In this paper, we focus on integration and scaling of systems that include ML components. The setup we assume is that of continuous deployment [6], where new versions of the system can be rapidly deployed – often referred to DevOps [3], [5] in software development. When also ML components are deployed in a similar fashion, term MLOps [2], [23] is used. As a concrete example of MLOps pipelines, we use Continuous Delivery for Machine Learning (CD4ML
Modified on	07/02/2022 11:27

Name	MLOps Challenges in Multi-Organization Setup~ Experiences from Two Real-World Cases
Number of Coding References	11
Number of Codes Coding	2
Coverage	14.91%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 10:17
Coded Text	
Modified on	07/02/2022 11:29

Name	MLOps Challenges in Multi-Organization Setup~ Experiences from Two Real-World Cases
Number of Coding References	11
Number of Codes Coding	2
Coverage	14.91%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 10:17

Coded Text	A new challenge in software engineering for ML is data related operations. These operations are related to the above to some extent, especially when data sets cannot be moved across data boundaries, but multiple organizations need to access the data. Moreover, data meshes and other forms of integrating data in pieces can complicate designing the more traditional parts of information systems, needed for such integration. In addition, when considering AuroraAI, it seems natural that different solutions might rely on different versions of data sets, for several reasons. For instance, it is possible that extensive data cleaning operations are needed for some applications, meaning that executing such operations frequently is impossible. Similarly, it might be so that the data must be from the same temporal range, and otherwise the operations make no sense. Similar complications are reflected to training ML models based on such data sets, as well as to monitoring how well the models work once they have been deployed. For operationalizing all the above in practice, the same skill gap as for starting to use MLOps within a single organization is valid – indeed the same actions need to be taken. However, this time some of the issues are more difficult to reconcile, because the organizations may have different modes of operation and different organization cultures, as demonstrated in the Oravizio case. Moreover, restrictions, such as those related to privacy or certification, may exist on either side of the boundary, which adds an additional layer of complexity to the design. This has also been identified as a direction for future work, especially from the perspective of governance, auditing, and regulations [22]. In general, to successfully perform multi-organization MLOps, we need patters of integration that help us in the process. Inspiration for these can be found from system integration [10] as well as legality patterns, proposed for open source [9]. In fact, both solutions we have used in the examples of the paper are analogous to patterns of [9] – Oravizio uses the ML model as an Evaluator, and in AuroraAI, User delegation helps to combine data that can only be accessed by the user as a whole. The definition of such patterns remains future work, with some ideas already proposed in [17]. Finally, based on both case studies reported in this paper, it seems that if there is the will, there often is a way
Modified on	07/02/2022 11:33

Name	MLOps Challenges in Multi-Organization Setup~ Experiences from Two Real-World Cases
Number of Coding References	11
Number of Codes Coding	2
Coverage	14.91%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 10:17
Coded Text	Consequently, operations related to data seem to be the most difficult to put into practice. In general, systems like datalakes can be used to integrate data from various sources, but if amounts of data are massive and, in addition, its owners want to protect it, this option is feasible only inside one organization.
Modified on	07/02/2022 11:30

Name	MLOps Challenges in Multi-Organization Setup~ Experiences from Two Real-World Cases
Number of Coding References	11
Number of Codes Coding	2
Coverage	14.91%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 10:17
Coded Text	Finally, the tool is meant to help the doctor and the patient to discuss the risks related to a surgical operation, not to decide whether or not to perform the operation. Instead, the decision is always made by the humans, and the AI only has a supporting role in the process. Hence, the responsibility is carried by humans, not by the AI. Furthermore, in the unlikely event of the system malfunctioning and providing answers that clearly are infeasible, the doctor – an expert in such operations – is able to notice them and fix the situation.
Modified on	07/02/2022 11:31

Name	MLOps Challenges in Multi-Organization Setup~ Experiences from Two Real-World Cases
Number of Coding References	11
Number of Codes Coding	2
Coverage	14.91%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 10:17
Coded Text	Firstly, the model is created, and its quality assurance activities are carried out on the hospital's premises as a shared activity between the two organizations. The mode of operation for this is based on experiments where interesting properties are identified in the dataset, which in general is often the nature of data science projects early on [1]. The rhythm for the operations is defined by these experiments. If desired, the model can be re-created with more precision in given intervals or by some other valid form of meaningful iteration. Each new iteration cycle creates a new version of the model, and it may or may not be handed over to the service provider. Secondly, the service provider is responsible for the devel- opment and the operations of the software that are necessary to use the model as basis for collaboration between the doctors and (potential) patients.
Modified on	07/02/2022 11:31

Name	MLOps Challenges in Multi-Organization Setup~ Experiences from Two Real-World Cases
Number of Coding References	11
Number of Codes Coding	2
Coverage	14.91%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 10:17
Coded Text	For AuroraAI, this has meant that instead of aiming at automata that can provide recommendations for everyone, models are more targeted to individuals, who can use them to determine facts about their well-being. Moreover, based on the models and input from the user, recommendations are given to propose actions to add the observed well-being. Obviously, if an individual citizen chooses to share the results with municipalities, chances are that the individual in question will get a better, more targeted service proposals. However, sharing the results is by no means enforced, meaning that the resulting data set is heterogeneous from the society perspective.
Modified on	07/02/2022 11:33

Name	MLOps Challenges in Multi-Organization Setup~ Experiences from Two Real-World Cases
Number of Coding References	11
Number of Codes Coding	2
Coverage	14.91%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 10:17
Coded Text	Since models are concrete assets in the ML context as well as from operations perspective, they are also something that can be easily shared in AuroraAI. However, these models are only partial, as they are built by different data owners, not based on personal data that only
Modified on	07/02/2022 11:32

Name	MLOps Challenges in Multi-Organization Setup~ Experiences from Two Real-World Cases
Number of Coding References	11
Number of Codes Coding	2
Coverage	14.91%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 10:17
Coded Text	Supporting interoperability at technical, informational and governance levels, such an ecosystem is aligned with the AuroraAI vision, where it is the individuals who combine data, not the society. The use of the digital twin paradigm [8] has also been considered in this context [12], leading to citizen-level use of datasets and recommendations. Unfortunately, such an approach, relying on datasets owned by multiple organizations, does not really provide a data set that would be easily available for ML or even deeper analysis. Firstly, MyData is not automatically shared but is something that only the individuals can release in accordance to their wishes. Secondly it is not obvious which data is true and which false, as individuals themselves provide some data, and, moreover, they can manipulate some data.
Modified on	07/02/2022 11:31

Name	MLOps Challenges in Multi-Organization Setup~ Experiences from Two Real-World Cases
Number of Coding References	11
Number of Codes Coding	2
Coverage	14.91%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 10:17
Coded Text	That said, individual offices often have such systems in place locally, as this is governed by law. Hence, they can monitor what takes place, and, at least to some extent, who accesses what. Opening such monitoring data to individuals with
Modified on	07/02/2022 11:33

Name	MLOps Challenges in Multi-Organization Setup~ Experiences from Two Real-World Cases
Number of Coding References	11
Number of Codes Coding	2
Coverage	14.91%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 10:17

Coded Text	While numerous proposals exist from different vendors, perhaps the most well-known incarnation of MLOps is Continuous Delivery for Machine Learning (CD4ML) [19]. The approach formalized by ThoughtWorks for automating in an end-to-end fashion the lifecycle of machine learning applications. In CD4ML, a cross-functional team produces machine learning applications based on code, data, and models in small and safe increments that can be reproduced and reliably released at any time, in short adaptation cycles. The approach contains three distinct steps: identify and prepare the data for training, experimenting with different models to find the best performing candidate, and deploying and using the selected model in production. The work is split to an ML pipeline that works with the data, and to a deployment pipeline that deploys the result to operations (Figure 1). The above implies that there are three artifacts, in addition to those that are required by DevOps, that need version control in MLOps: (i) different data sets used for training model and their versioning; (ii) model and its versioning; and (iii) monitoring the output of the model to detect bias and other problems.
Modified on	07/02/2022 11:28

Name	Model Provenance Management in MLOps Pipeline
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Model Provenance Management in MLOps Pipeline Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Moving from Composable to Programmable
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Moving from Composable to Programmable Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	MSR4ML~ Reconstructing Artifact Traceability in Machine Learning Repositories
Number of Coding References	11
Number of Codes Coding	3
Coverage	13.10%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 11:05
Coded Text	
Modified on	07/02/2022 11:07

Name	MSR4ML~ Reconstructing Artifact Traceability in Machine Learning Repositories
Number of Coding References	11
Number of Codes Coding	3
Coverage	13.10%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 11:05
Coded Text	Artifact classifier This module is responsible for classifying artifacts into different categories. We can distinguish between four main artifacts of ML project: data, configuration, code and models [1], [5].
Modified on	07/02/2022 11:07

Name	MSR4ML~ Reconstructing Artifact Traceability in Machine Learning Repositories
Number of Coding References	11
Number of Codes Coding	3
Coverage	13.10%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 11:05
Coded Text	Artifact usage identifier The artifact identifier traverses the AST produced by the code parser and identifies all the methods or functions that interact with files. The assumption is that methods interacting with files may reveal links between code and data files.
Modified on	07/02/2022 11:07
Name	MSR4ML~ Reconstructing Artifact Traceability in Machine Learning Repositories
Number of Coding References	11
Number of Codes Coding	3
Coverage	13.10%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 11:05
Coded Text	Commit tracker The commit tracker (Figure 5) is responsible for tracking the commits associated with each artifact in the project, by querying Git for the relevant commit data. While the basic logic simply allows the interaction with a Git repository, this can be extended with plug-ins to interact with other platforms, like GitHub or Bitbucket, or include other third party Git libraries.
Modified on	07/02/2022 11:08

Name	MSR4ML~ Reconstructing Artifact Traceability in Machine Learning Repositories
Number of Coding References	11
Number of Codes Coding	3
Coverage	13.10%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 11:05
Coded Text	In this work, we propose a framework for automatic identification and traceability of links between data, code and ML model through Mining Software Repositories (MSR) techniques. Our tool combines static code analysis and mining commit data to identify ML, code and data artifacts, reconstruct links between them and retrieve commits that affect each end of the link. The objective is to increase productivity and the developers' awareness of their project through the recovered traceability.
Modified on	07/02/2022 11:05

Name	MSR4ML~ Reconstructing Artifact Traceability in Machine Learning Repositories
Number of Coding References	11
Number of Codes Coding	3
Coverage	13.10%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 11:05
Coded Text	MSR4ML (MSR for ML) is a framework for automatic identification and tracing of artifact usage in Git-based ML projects. It explores the code and the repository of a ML project to extract relevant information about artifact usage and retrieve links between them. It aims to provide a tool for reconstructing traceability in existing Git-based ML projects.
Modified on	07/02/2022 11:06

Name	MSR4ML~ Reconstructing Artifact Traceability in Machine Learning Repositories
Number of Coding References	11
Number of Codes Coding	3
Coverage	13.10%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 11:05
Coded Text	Source code parser The source code parser is responsible for parsing code files and obtain their Abstract Syntax Tree (AST) representation for further analysis. The output of the parser is an extended AST node representing the contents of a code file. The AST allows us to traverse the code's labelled nodes and find specific elements, including method invocations, variable declarations and accesses, string literals and others. The resulting AST must be complete, so that if reversed engineered, it would produce the exact original code.
Modified on	07/02/2022 11:07

Name	MSR4ML~ Reconstructing Artifact Traceability in Machine Learning Repositories
Number of Coding References	11
Number of Codes Coding	3
Coverage	13.10%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 11:05
Coded Text	The primary objective of this work is to initiate a discussion about the peculiarities of ML applications as software projects and emphasize the need to increase the awareness of the diverse development teams about their artifacts. In this paper, we propose to leverage static code analysis and mining software repository (MSR) techniques to recover links between code, data, and ML models and improve traceability in Git-based
Modified on	07/02/2022 11:06

Name	MSR4ML~ Reconstructing Artifact Traceability in Machine Learning Repositories
Number of Coding References	11
Number of Codes Coding	3
Coverage	13.10%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 11:05
Coded Text	Traceability: Our traceability process takes advantage of Git features and model-artifact links to monitor the evolution of a ML project. It can be used for many purposes such as model metadata extraction, model analysis, continuous integration, and others. Consider the following example of its application for model analysis. A developer may ask: ”What changes caused the model to perform worse?” The corresponding query can be adapted as: ”Retrieve all commits affecting the model and its artifacts between the current version of the model and the previous one.” Our framework will follow these steps to get the information: 1) Using the model filename, extract the exact time t of the previous commit modifying the model; 2) Retrieve all the artifacts that are linked with the model; 3) Extract all the commits modifying these artifacts from t until now; 4) Return the commits, classifying them according to the priority of the link between the artifact and the model.
Modified on	07/02/2022 11:07

Name	MSR4ML~ Reconstructing Artifact Traceability in Machine Learning Repositories
Number of Coding References	11
Number of Codes Coding	3
Coverage	13.10%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 11:04
Coded Text	The increasing popularity of Machine Learning (ML) is generating challenges also for developers. The multitude of programming languages, libraries and available resources allow them to easily build their own models or algorithms. However, ML models are tightly connected to their data implying a different development process from other types of software. Software projects often rely on version control platforms, such as GitHub, but these platforms have not yet been extended to support ML projects. There is poor support for data versioning and no link between ML and software artifacts. Thus, traceability and model evolution can become challenging for developers. While some specific ML platforms exist, they still require considerable manual specification of ML artifacts and links between them.
Modified on	07/02/2022 11:04

Name	MSR4ML~ Reconstructing Artifact Traceability in Machine Learning Repositories
Number of Coding References	11
Number of Codes Coding	3
Coverage	13.10%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56
Coded Text	ML traceability and reproducibility is a trending topic in the research community [13]. Aravantinos proposed a descriptive traceability model for deep learning applications [14]. By using software engineering methods, the author proposes a set of activities and artifacts for deep learning development management. The proposed model also integrates model requirements and architecture. This was a valuable source of inspiration during the implementation of our framework. Our contribution can be used as an extension to this descriptive model. For example, artifacts and their attributes as proposed in the model can be used to organize and manage artifact types used by our classification module. ML traceability requires full management of the project's metadata. Tsay et al. proposed AIMMX, a tool for automatic extraction of model's metadata from repositories [15]. Using MSR techniques and machine learning, AIMMX extracts relevant information from ML repositories such as the dataset name and the AI framework used by the model. However, contrary to our framework, the tool does not retrieve the actual data files, nor the link with code files. It is mainly focused on identification of model's metadata for knowledge purposes. Meurice et al. [16] use a similar method as the artifact identification module to assess specific data artifact usage in the code. They use static analysis of code files to identify database usage in Java projects, as we do to extract artifact usage in ML projects: specify method names, extract arguments from method invocation node and infer their values. When looking at the existing ML management platforms, we found Data Version Control (DVC) [17] to be a powerful tool proposing traceability in Git-based ML projects. It extends Git to add capabilities such as data, hyper-parameters, metrics and model versioning. However, it needs a manual initial configuration phase where ML artifacts are provided. Also, it supports only the links between input files, their corresponding code and output file. It does not provide traceability for coupled code files. This limits its capacity for evolution analysis since the retrieval of specific modifications affecting a final model and its artifacts is limited. Our framework could be used to preprocess the software repository and prepare it for integration with DVC.
Modified on	22/07/2022 16:48

Name	Multimodal AutoML for Image, Text and Tabular Data
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Multimodal AutoML for Image, Text and Tabular Data Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Non-Functional Requirements for Machine Learning~ Challenges and New Directions
Number of Coding References	5
Number of Codes Coding	3
Coverage	4.61%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Software Mantainability Aspects
Created on	04/02/2022 13:27
Coded Text	As a concrete example of an ML solution and its associated qualities, consider an airport which may screen passengers against images of people of interest; here a precise match is desirable, as the cost of misidentification is relatively high, yet the processing time may be relatively slow (e.g., 20 seconds), given the time one takes to get through security. These desired quality requirements should be captured and considered through the solution lifetime. In order to understand what ML solutions may meet our quality requirements, we can attempt a relatively straightforward application of existing frameworks for NFR modeling, such as the NFR Framework [7] or iStar [11]. We do so
Modified on	07/02/2022 10:55

Name	Non-Functional Requirements for Machine Learning~ Challenges and New Directions
Number of Coding References	5
Number of Codes Coding	3
Coverage	4.61%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:22
Coded Text	Accuracy & Performance. Most ML work reports on algorithm accuracy (often precision and recall), i.e., how “correct” the output is compared to reality. Further work looks more broadly at algorithm performance (e.g., [24]), including comparisons of performance in specific contexts (e.g., [25]). Fairness. Recent work has focused on technical solutions to make ML algorithms more fair, finding that the removal of sensitive features (e.g., race, gender) is not sufficient to ensure fair results, and considering the trade-off between fairness and other NFRs [2]. Work in this area has attempted to find mathematical or formal
Modified on	07/02/2022 10:56

Name	Non-Functional Requirements for Machine Learning~ Challenges and New Directions
Number of Coding References	5
Number of Codes Coding	3
Coverage	4.61%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:22
Coded Text	In particular, the nature of ML means that the meaning of many NFRs for ML solutions differs compared to regular software, and these NFRs are often not well understood (e.g., what is fairness? [10]). What does it mean for an ML-enabled system to be maintainable? Are NFRs such as compatibility and modularity still relevant? Some NFRs may have reduced importance for ML solutions compared to typical software. On the other hand, NFRs such as fairness [2] and transparency [3] have become critical from an ML perspective, whereas previous NFR work has not typically emphasized these dimensions. Further, as-yet-unexplored NFRs such as “retrainability” may also become relevant.
Modified on	07/02/2022 10:54

Name	Non-Functional Requirements for Machine Learning~ Challenges and New Directions
Number of Coding References	5
Number of Codes Coding	3
Coverage	4.61%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:22
Coded Text	Machine Learning (ML) provides approaches which use big data to enable algorithms to “learn”, producing outputs which would be difficult to obtain otherwise. Despite the advances allowed by ML, much recent attention has been paid to certain qualities of ML solutions, particularly fairness and transparency, but also qualities such as privacy, security, and testability. From a requirements engineering (RE) perspective, such qualities are also known as non-functional requirements (NFRs). In RE, the meaning of certain NFRs, how to refine those NFRs, and how to use NFRs for design and runtime decision making over traditional software is relatively well established and understood. However, in a context where the solution involves ML, much of our knowledge about NFRs no longer applies. First, the types of NFRs we are concerned with undergo a shift: NFRs like fairness and transparency become prominent, whereas other NFRs such as modularity may become less relevant. The meanings and interpretations of NFRs in an ML context (e.g., maintainability, interoperability, and usability) must be rethought, including how these qualities are decomposed into sub-qualities. Tradeoffs between NFRs in an ML context must be re-examined. Beyond the changing landscape of NFRs, we can ask if our known approaches to understanding, formalizing, modeling, and reasoning over NFRs at design and runtime must also be adjusted, or can be applied as-is to this new area? Given these questions, this work outlines challenges and a proposed research agenda for the exploration of NFRs for ML-based solutions.
Modified on	07/02/2022 10:52

Name	Non-Functional Requirements for Machine Learning~ Challenges and New Directions
Number of Coding References	5
Number of Codes Coding	3
Coverage	4.61%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Research gap in Maintaining ML systems
Created on	07/02/2022 10:53
Coded Text	Machine Learning (ML) provides approaches which use big data to enable algorithms to “learn”, producing outputs which would be difficult to obtain otherwise. Despite the advances allowed by ML, much recent attention has been paid to certain qualities of ML solutions, particularly fairness and transparency, but also qualities such as privacy, security, and testability. From a requirements engineering (RE) perspective, such qualities are also known as non-functional requirements (NFRs). In RE, the meaning of certain NFRs, how to refine those NFRs, and how to use NFRs for design and runtime decision making over traditional software is relatively well established and understood. However, in a context where the solution involves ML, much of our knowledge about NFRs no longer applies. First, the types of NFRs we are concerned with undergo a shift: NFRs like fairness and transparency become prominent, whereas other NFRs such as modularity may become less relevant. The meanings and interpretations of NFRs in an ML context (e.g., maintainability, interoperability, and usability) must be rethought, including how these qualities are decomposed into sub-qualities. Tradeoffs between NFRs in an ML context must be re-examined. Beyond the changing landscape of NFRs, we can ask if our known approaches to understanding, formalizing, modeling, and reasoning over NFRs at design and runtime must also be adjusted, or can be applied as-is to this new area? Given these questions, this work outlines challenges and a proposed research agenda for the exploration of NFRs for ML-based solutions.
Modified on	07/02/2022 10:53

Name	On misbehaviour and fault tolerance in machine learning systems
Number of Coding References	23
Number of Codes Coding	2
Coverage	6.48%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	As such, in this paper, the goal is to gather additional knowledge on fault tolerance solutions and beyond, and the practical applicability and reasoning of the solutions – which are used and considered useful. We reached out to experienced software architects familiar with ML through their work. In this way, we aim to shed light on which design solutions are seen as useful by experts, which are not, and which need additional studying, thus answering the lack of research on the functionality of deployed ML models identified by Zhang et al. [5].
Modified on	07/02/2022 10:43

Name	On misbehaviour and fault tolerance in machine learning systems
Number of Coding References	23
Number of Codes Coding	2
Coverage	6.48%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	Fall-over options Over the course of the interviews, fall-over procedures were mentioned by the respondents. Essentially, a fall-over means what to do when an error is detected. The recovery blocks of the previous subsection fall into this category as well: when an error is detected, the input is handed over to another model which acts as a fall-over component.
Modified on	07/02/2022 10:48

Name	On misbehaviour and fault tolerance in machine learning systems
Number of Coding References	23
Number of Codes Coding	2
Coverage	6.48%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	Fault-tolerance solution proposals Input checker (used by Jonsson et al. [16]) is a component that aims to prohibit such inputs from entering the ML model that could activate the ML model's faults. Thus, the faults are tolerated by limiting the potential situations in which they could cause errors.
Modified on	07/02/2022 10:45

Name	On misbehaviour and fault tolerance in machine learning systems
Number of Coding References	23
Number of Codes Coding	2
Coverage	6.48%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	Input checker Input checkers were rarely being used in practice. However, there is use for input checkers, when certain conditions are met. First of all, hard limits on inputs were seen – at best – as an efficient way to prevent poor quality data from entering the model. For example, broken data or data beneath or above some threshold can be filtered out. It may be that the model cannot handle null values, or its results may be unreliable if the user – for example – has not watched enough videos for a recommendation.
Modified on	07/02/2022 10:47
Name	On misbehaviour and fault tolerance in machine learning systems
Number of Coding References	23
Number of Codes Coding	2
Coverage	6.48%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	Input distribution observing Input distribution observing was not one of the original study propositions was but, however, mentioned by every respondent. The statistics of the inputs are measured over time, and deviations in the statistics either alert the developers, or potentially lead to some predefined actions being taken.
Modified on	07/02/2022 10:48

Name	On misbehaviour and fault tolerance in machine learning systems
Number of Coding References	23
Number of Codes Coding	2
Coverage	6.48%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	Misbehaviour is usually the result of faulty implementation, misuse of the model's results, or a poor or buggy model.
Modified on	07/02/2022 10:46

Name	On misbehaviour and fault tolerance in machine learning systems
Number of Coding References	23
Number of Codes Coding	2
Coverage	6.48%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	ML testing into offline and online testing. Offline testing is basically ML model validation [15], whereas online testing includes the initial testing after model deployment, and the measures taken to ensure correct functionality beyond initial tests, such as monitoring and other fault tolerant patterns. However, the papers yielded by their search presented mostly offline testing, and very little online testing.
Modified on	07/02/2022 10:45

Name	On misbehaviour and fault tolerance in machine learning systems
Number of Coding References	23
Number of Codes Coding	2
Coverage	6.48%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	Model observers Measuring the resource consumption of the ML model was mostly disregarded as a tool for fault tolerance for a ML system, but was considered more as a development tool to indicate non-optimal solutions when building an ML model.
Modified on	07/02/2022 10:48

Name	On misbehaviour and fault tolerance in machine learning systems
Number of Coding References	23
Number of Codes Coding	2
Coverage	6.48%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	On misbehaviour ofML systems (RQ1) The mentioned kinds of misbehaviour were unexpected input-output pairs, poor quality of incoming data, and decay of the ML model over time. The first could be considered the simplest kind of erroneous behaviour: with some inputs, the
Modified on	07/02/2022 10:46

Name	On misbehaviour and fault tolerance in machine learning systems
Number of Coding References	23
Number of Codes Coding	2
Coverage	6.48%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	On the role offault tolerance in ML software (RQ2) The need for and role of fault tolerance was deemed to be contextual and varying. As mentioned earlier in Section
Modified on	07/02/2022 10:46

Name	On misbehaviour and fault tolerance in machine learning systems
Number of Coding References	23
Number of Codes Coding	2
Coverage	6.48%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	Output checker (used by Prado et al. [17] and Li et al. [18], also known as acceptance test [11]) is a component which detects errors by assessing ML model's outputs and prevents errors from propagating further into other parts of the system.
Modified on	07/02/2022 10:45

Name	On misbehaviour and fault tolerance in machine learning systems
Number of Coding References	23
Number of Codes Coding	2
Coverage	6.48%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	Output checker The respondents considered hard limits on outputs more use- ful than their counterparts for inputs. Again, business rules or easily confirmable erroneous outputs with direct consequences to users are what set the rules for outputs. For example, business executives might not even approve an autonomous
Modified on	07/02/2022 10:47

Name	On misbehaviour and fault tolerance in machine learning systems
Number of Coding References	23
Number of Codes Coding	2
Coverage	6.48%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	Output distribution observing Our study proposal of comparing outputs to historical data was not triumphant when it concerned single outputs. Instead, monitoring the distribution of outputs in a manner similar to inputs in Section 5.3.2 is something that the respondents mentioned frequently.
Modified on	07/02/2022 10:48

Name	On misbehaviour and fault tolerance in machine learning systems
Number of Coding References	23
Number of Codes Coding	2
Coverage	6.48%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	Patterns used as fault tolerance (RQ3) In this section, we present what the respondents thought about the fault tolerance solutions as presented in the study
Modified on	07/02/2022 10:47

Name	On misbehaviour and fault tolerance in machine learning systems
Number of Coding References	23
Number of Codes Coding	2
Coverage	6.48%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	Redundant models Having multiple divergent models as recovery blocks to hand the inputs over to was seen as somewhat useful as a fall-over approach in case the main model not give any outputs, or if it was possible to detect erroneous outputs.
Modified on	07/02/2022 10:48

Name	On misbehaviour and fault tolerance in machine learning systems
Number of Coding References	23
Number of Codes Coding	2
Coverage	6.48%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	Solution proposals selection The patterns chosen are either mentioned in earlier research in the context of ML, presented in materials for traditional software, or are a modification of some of these solutions which we
Modified on	07/02/2022 10:45

Name	On misbehaviour and fault tolerance in machine learning systems
Number of Coding References	23
Number of Codes Coding	2
Coverage	6.48%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	
Modified on	07/02/2022 10:49

Name	On misbehaviour and fault tolerance in machine learning systems
Number of Coding References	23
Number of Codes Coding	2
Coverage	6.48%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	Architectural designs have been suggested to protect the systems from hardware failures and malicious attacks (e.g. [6]), but there is little emphasis on architectural software design to answer the inherent unpredictability and uncertainty of the utilized ML itself. One step towards achieving dependability is fault tolerance. Traditionally, software faults have been seen as the results of design errors [7]. However, due to their statistical, data-driven nature, ML systems can be seen as inherently faulty not by design, but by paradigm. Thus, unpredictable errors will emerge from deployed ML systems that cannot be captured by traditional fault-tolerance models. The question remains of how to build ML systems that detect these errors and prevent them from propagating.
Modified on	07/02/2022 10:42
Name	On misbehaviour and fault tolerance in machine learning systems
Number of Coding References	23
Number of Codes Coding	2
Coverage	6.48%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	Dependability, faults, and ML systems System dependability means a system's trustworthiness [2]. Dependability is assessed by evaluating a system's reliability, availability, and maintainability. Sometimes additional quality characteristics, such as safety and integrity, are applied [10]. In other words, a dependable system – at the very least – delivers correct service consistently, does not suffer from long periods of down-time, and is easily corrected and altered. Threats to dependability originate from failures, errors, and faults [10]. Failures are deviations from the desired service. Failures result from propagating errors, i.e., incorrect functioning of the system. Errors are caused by faults that are defects in system's components (software or hardware), activated by given inputs in a given state.
Modified on	07/02/2022 10:44

Name	On misbehaviour and fault tolerance in machine learning systems
Number of Coding References	23
Number of Codes Coding	2
Coverage	6.48%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	Key findings are: • ML system can provide poor results if the inputs are of poor quality, the input-output-pairs do not match, or the input distribution drifts. This can be caused by a buggy model, faulty deployment, changes in user base, or misuse of the models results. • Interest in fault tolerance is rising but its overall role and frameworks for it are still developing. • Some patterns for fault tolerance can be – and already are – used to tackle the problems caused by the ML model in the system, despite the field still developing.
Modified on	07/02/2022 10:48

Name	On misbehaviour and fault tolerance in machine learning systems
Number of Coding References	23
Number of Codes Coding	2
Coverage	6.48%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	The results show that there is much to desire in the dependability of ML systems. Some patterns for fault tolerance are used in practice, but the developers and buyers often lack knowledge and frameworks to apply them. Thus, the role of fault tolerance is – at least today – very limited and vague in practice. This relative immaturity is not limited to fault tolerance, however, but also other phases of managing the life-cycle of ML systems. To our knowledge, this is the first attempt to gather information about fault tolerance for ML systems in one place, thus forming the basis for further research. Practitioners can use the gathered information to design more dependable ML systems.
Modified on	07/02/2022 10:44

Name	On misbehaviour and fault tolerance in machine learning systems
Number of Coding References	23
Number of Codes Coding	2
Coverage	6.48%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	Two means of diminishing threats are fault prevention and fault tolerance.
Modified on	07/02/2022 10:44

Name	On misbehaviour and fault tolerance in machine learning systems
Number of Coding References	23
Number of Codes Coding	2
Coverage	6.48%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	We studied software designs that aim at introducing fault tolerance in ML systems so that possible problems in ML components of the systems can be avoided. The research was conducted as a case study, and its data was collected through five semi-structured interviews with experienced software architects.
Modified on	07/02/2022 10:42

Name	On testing machine learning programs
Number of Coding References	27
Number of Codes Coding	6
Coverage	6.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	Dead experimental code paths which happens when code is written for rapid prototyping to gain quick turnaround times by performing additional experiments simply by tweaks and experimental code paths within the main production code.
Modified on	07/02/2022 10:21

Name	On testing machine learning programs
Number of Coding References	27
Number of Codes Coding	6
Coverage	6.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	03/02/2022 10:06
Coded Text	Conceptual issues. One key assumption behind the training process of supervised ML models is that the training dataset, the validation dataset, and the testing dataset, which are sampled from manually labeled data, are representative samples of the underlying problem. Following the concept of Empirical Risk Minimization (ERM), the optimizer allows finding the fitted model that minimizes the empirical risk; which is the loss computed over the training data assuming that it is a representative sample of the target distribution. The empirical risk can correctly approximates the true risk only if the training data distribution is a good approximation of the true data distribution (which is often out of reach in real-world scenarios). The size of the training dataset has an impact on the approximation goodness of the true risk, i.e.
Modified on	07/02/2022 10:22

Name	On testing machine learning programs
Number of Coding References	27
Number of Codes Coding	6
Coverage	6.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	a)Use of the centered formula. Instead of relying on the traditional gradient formula, Karpathy recommends using the centered formula from Equation 1 which is more precise. The taylor expansion of the numerator
Modified on	07/02/2022 10:26

Name	On testing machine learning programs
Number of Coding References	27
Number of Codes Coding	6
Coverage	6.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Approaches that aim to detect conceptual errors in ML models Approaches in this category assume that the models are implemented into programs without errors and focus on providing mechanisms to detect potential errors in the calibration of the models. These approaches can be divided in two groups: black-box and white-box approaches [9].
Modified on	07/02/2022 10:24

Name	On testing machine learning programs
Number of Coding References	27
Number of Codes Coding	6
Coverage	6.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Approaches that aim to detect errors in ML code implementations Given the stochastic nature of most ML algorithms and the absence of oracles, most existing testing techniques are inadequate for ML code implementations. As a consequence, the ML community have resorted to numerical testing, property-based testing
Modified on	07/02/2022 10:25

Name	On testing machine learning programs
Number of Coding References	27
Number of Codes Coding	6
Coverage	6.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	b) Use of relative error for the comparison. As mentioned above, developers perform gradient checking by computing the difference between the numerical gradient f⍰ n and the analytic gradient f⍰ a. This difference can be seen as an absolute error and the aim of the gradient checking
Modified on	07/02/2022 10:26

Name	On testing machine learning programs
Number of Coding References	27
Number of Codes Coding	6
Coverage	6.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Black-box testing approaches for ML models. The common denominator to black-box testing approaches is the generation of adversarial data set that is used to test the ML models. These approaches leverage statistical analysis techniques to devise a multidimensional random process that can generate data with the same statistical characteristics as the input data of the model.
Modified on	07/02/2022 10:24

Name	On testing machine learning programs
Number of Coding References	27
Number of Codes Coding	6
Coverage	6.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Coverage-Guided Fuzzing. Odena and Goodfellow [38] developed a coverage-guided fuzzing framework specialized for testing neural networks. Coverage-guided fuzzing has been used in traditional software testing to find critical errors. For ML code, the fuzzing process consists of handling an input corpus that evolves through the execution of tests by applying random mutation operations on its contained data and keeping only interesting instances that allow triggering new program behavior.
Modified on	07/02/2022 10:27

Name	On testing machine learning programs
Number of Coding References	27
Number of Codes Coding	6
Coverage	6.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	d) Stick around active range of floating point. To train complex statistical models, one needs large amounts of data. So, it is common to opt for mini-batch stochastic gradient descent and to normalize the loss function over the batch. However, if the back-propagated gradient is very small, additional divisions by data inputs count will yield extremely smaller vales, which in turn can lead to numerical issues.
Modified on	07/02/2022 10:26
Name	On testing machine learning programs
Number of Coding References	27
Number of Codes Coding	6
Coverage	6.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	In this paper, we survey existing testing practices that have been proposed for ML programs, explaining the context in which they can be applied and their expected outcome. We also, identify gaps in the literature related to the testing of ML programs and suggest future research directions for the scientific community. This paper makes the following contributions: • We present and explain challenges related to the testing of ML programs that use differentiable models. • We provide a comprehensive review of current software testing practices for ML programs. • We identify gaps in the literature related to the testing of ML programs and provide future research directions for the scientific community.
Modified on	07/02/2022 10:17

Name	On testing machine learning programs
Number of Coding References	27
Number of Codes Coding	6
Coverage	6.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Metamorphic testing. Murphy et al. [32] introduced metamorphic testing to ML in 2008. They defined several Metamorphic Relationships (MRs) that can be classified into six categories (i.e., additive, multiplicative, permutative, invertive, inclusive, and exclusive).
Modified on	07/02/2022 10:26

Name	On testing machine learning programs
Number of Coding References	27
Number of Codes Coding	6
Coverage	6.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Mutation testing. Ma et al.[36] proposed DeepMutation that adapts mutation testing [37] to DNN-based systems with the aim of evaluating the test data quality in terms of its capacity to detect faults in the programs. Mutation testing consists in injecting artificial faults (i.e., mutants) in a program under test and generating test cases to detect them.
Modified on	07/02/2022 10:26

Name	On testing machine learning programs
Number of Coding References	27
Number of Codes Coding	6
Coverage	6.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Numerical-based testing: Finite-difference techniques. Most machine learning algorithms are formulated as optimization problems that can be solved using gradientbased optimizers, such as gradient descent or L-BFGS (i.e., Limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm). The correctness of the objective function gradient that are computed with respect to the model parameters, is crucial.
Modified on	07/02/2022 10:25

Name	On testing machine learning programs
Number of Coding References	27
Number of Codes Coding	6
Coverage	6.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Property-based testing. Property-based testing is a technique that consists in inferring the properties of a computation using the theory and formulating invariants that should be satisfied by the code.
Modified on	07/02/2022 10:26

Name	On testing machine learning programs
Number of Coding References	27
Number of Codes Coding	6
Coverage	6.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	This paper reviews current existing testing practices for ML programs. First, we identify and explain challenges that should be addressed when testing ML programs. Next, we report existing solutions found in the literature for testing ML programs. Finally, we identify gaps in the literature related to the testing of ML programs and make recommendations of future research directions for the scientific community. We hope that this comprehensive review of software testing practices will help ML engineers identify the right approach to improve the reliability of their ML-based systems. We also hope that the research community will act on our proposed research directions to advance the state of the art of testing for ML programs.
Modified on	07/02/2022 10:15

Name	On testing machine learning programs
Number of Coding References	27
Number of Codes Coding	6
Coverage	6.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	White-box testing approaches for ML models. Pei et al. proposed DeepXplore [15], the first white-box
Modified on	07/02/2022 10:24

Name	On testing machine learning programs
Number of Coding References	27
Number of Codes Coding	6
Coverage	6.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	Conceptual issues. Once data is gathered, cleaning data tasks are required to ensure that the data is consistent, free from redundancy and given a reliable starting point for statistical learning. Common cleaning tasks include: (1) removing invalid or undefined values (i.e., Not-a-Number, Not-Available), duplicate rows, and outliers that seems to be too different from the mean value); and (2) unifying the variables' representations to avoid multiple data formats and mixed numerical scales. This can be done by data transformations such as normalization, min-max scaling, and data format conversion. This pre-processing step allows to ensure a high quality of raw data,
Modified on	07/02/2022 10:18

Name	On testing machine learning programs
Number of Coding References	27
Number of Codes Coding	6
Coverage	6.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	Dead experimental code paths which happens when code is written for rapid prototyping to gain quick turnaround times by performing additional experiments simply by tweaks and experimental code paths within the main production code.
Modified on	07/02/2022 10:20

Name	On testing machine learning programs
Number of Coding References	27
Number of Codes Coding	6
Coverage	6.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	Implementation issues. To process data as described above, ML engineers implement data pipelines containing components for data transformations, validation, enrichment, summarization, and–or any other necessary treatment. Each pipeline component is separated from the others, and takes in a defined input, and returns a defined output that will be served as input data to the next component in the pipeline. Data
Modified on	07/02/2022 10:19

Name	On testing machine learning programs
Number of Coding References	27
Number of Codes Coding	6
Coverage	6.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	The identified patterns represent the core logic of the model. Changes in data (i.e., the input signals) are likely to have a direct impact on these patterns and hence on the behavior of the model and its corresponding predictions. Because of this strong dependence on data, ML models are considered to be data-sensitive or data-dependent algorithms. A poor selection of features can impact a ML system negatively. Sculley et al. [3] report that unnecessary dependencies to features that contribute with little or no value to the model quality can generate vulnerabilities and noises in a ML system. Examples of such features are : Epsilon Feature, which are features that have no significant contribution to the performance of the model, Legacy Feature, which are features that lost their added information value on model accuracy improvement when other more rich features are included in the model, or Bundled Features, which are groups of features that are integrated to a ML system simultaneously without a proper testing of the contribution of each individual feature.
Modified on	07/02/2022 10:19

Name	On testing machine learning programs
Number of Coding References	27
Number of Codes Coding	6
Coverage	6.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	Traditionally, software systems are constructed deductively, by writing down the rules that govern the behavior of the system as program code. However, with ML, these rules are inferred from training data (i.e.,, they are generated inductively). This paradigm shift in application development makes it difficult to reason about the behavior of software systems with ML components, resulting in systems that are intrinsically challenging to test and verify, given that they do not have (complete) specifications or even source code corresponding to some of their critical behaviors. In fact some ML programs rely on proprietary third-party libraries like
Modified on	07/02/2022 10:17
Name	On testing machine learning programs
Number of Coding References	27
Number of Codes Coding	6
Coverage	6.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	03/02/2022 10:24
Coded Text	Model Engineering: Challenges and Issues Once ML engineers have collected and processed the data, they proceed to finding the appropriate statistical learning model that could fit the available data in order to build its own logic and solve the given problem. A wide range of statistical models can be acquired and–or extended to suit different classification and regression purposes. There are simple models that make initial assumptions
Modified on	07/02/2022 10:20

Name	On testing machine learning programs
Number of Coding References	27
Number of Codes Coding	6
Coverage	6.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Research gap in Maintaining ML systems
Created on	07/02/2022 10:23
Coded Text	Approaches that aim to detect conceptual and implementation errors in data The approaches proposed in the literature to test the quality of data addresses both conceptual and implementation issues, therefore, we discuss these two aspects together in this section. The most common technique used to test the quality of data is the analysis-driven data cleaning that consists of applying analytical queries
Modified on	07/02/2022 10:23

Name	On testing machine learning programs
Number of Coding References	27
Number of Codes Coding	6
Coverage	6.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Research gap in Maintaining ML systems
Created on	07/02/2022 10:23
Coded Text	Approaches that aim to detect conceptual and implementation errors in ML models As discussed in Section 2.2 and illustrated on Figure 1, errors in ML models can be due to conceptual mistakes when creating the model or implementation errors when writing the code corresponding to the model.
Modified on	07/02/2022 10:24

Name	On testing machine learning programs
Number of Coding References	27
Number of Codes Coding	6
Coverage	6.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Research gap in Maintaining ML systems
Created on	07/02/2022 10:23
Coded Text	Approaches that aim to detect conceptual errors in ML models Approaches in this category assume that the models are implemented into programs without errors and focus on providing mechanisms to detect potential errors in the calibration of the models. These approaches can be divided in two groups: black-box and white-box approaches [9]. Black-box approaches are testing approaches that do not need access to the internal implementation details of the model under test.
Modified on	07/02/2022 10:24

Name	On testing machine learning programs
Number of Coding References	27
Number of Codes Coding	6
Coverage	6.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Research gap in Maintaining ML systems
Created on	07/02/2022 10:23
Coded Text	Black-box testing approaches for ML models. The common denominator to black-box testing approaches is the generation of adversarial data set that is used to test the ML models. These approaches leverage statistical analysis techniques to devise a multidimensional random process that can generate data with the same statistical characteristics as the input data of the model.
Modified on	07/02/2022 10:24

Name	On testing machine learning programs
Number of Coding References	27
Number of Codes Coding	6
Coverage	6.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Research gap in Maintaining ML systems
Created on	07/02/2022 10:23
Coded Text	Research Trends in ML Application Testing
Modified on	07/02/2022 10:23

Name	On the Co-evolution of ML Pipelines and Source Code - Empirical Study of DVC Projects
Number of Coding References	16
Number of Codes Coding	5
Coverage	4.15%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	As such, a new breed of data and model versioning tools have appeared to support data engineers and scientists [3]. Popular tools comprise DVC [4], MLFlow [5], Pachyderm [6], ModelDB [7] and Quilt Data [8]. They typically combine the ability to specify data and/or model pipelines, with advanced versioning support for data/models, and the ability to define and manage model experiments.
Modified on	04/02/2022 22:59

Name	On the Co-evolution of ML Pipelines and Source Code - Empirical Study of DVC Projects
Number of Coding References	16
Number of Codes Coding	5
Coverage	4.15%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	03/02/2022 10:06
Coded Text	As such, a new breed of data and model versioning tools have appeared to support data engineers and scientists [3]. Popular tools comprise DVC [4], MLFlow [5], Pachyderm [6], ModelDB [7] and Quilt Data [8]. They typically combine the ability to specify data and/or model pipelines, with advanced versioning support for data/models, and the ability to define and manage model experiments.
Modified on	04/02/2022 22:59
Name	On the Co-evolution of ML Pipelines and Source Code - Empirical Study of DVC Projects
Number of Coding References	16
Number of Codes Coding	5
Coverage	4.15%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	04/02/2022 23:00
Coded Text	~ Despite ML versioning being a young practice in open source repositories, 71.4% of the studied projects use at least two ofthe main DVCfeatures, i.e., data versioning and pipelines. More than half of the DVC files within projects past the experimentation stage are frequently changed, suggesting non-negligible maintenance effort for practitioners.
Modified on	04/02/2022 23:04

Name	On the Co-evolution of ML Pipelines and Source Code - Empirical Study of DVC Projects
Number of Coding References	16
Number of Codes Coding	5
Coverage	4.15%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	04/02/2022 23:00
Coded Text	Although there is low commit-level coupling amongst the DVC files ofa project, most coupling observed with dvcutilities and software artifacts are automated by DVC. On the contrary, DVC files and software artifacts such as tests and data files are rarely changed together at the commit-level.
Modified on	04/02/2022 23:04

Name	On the Co-evolution of ML Pipelines and Source Code - Empirical Study of DVC Projects
Number of Coding References	16
Number of Codes Coding	5
Coverage	4.15%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	04/02/2022 23:00
Coded Text	As such, a new breed of data and model versioning tools have appeared to support data engineers and scientists [3]. Popular tools comprise DVC [4], MLFlow [5], Pachyderm [6], ModelDB [7] and Quilt Data [8]. They typically combine the ability to specify data and/or model pipelines, with advanced versioning support for data/models, and the ability to define and manage model experiments.
Modified on	04/02/2022 23:00

Name	On the Co-evolution of ML Pipelines and Source Code - Empirical Study of DVC Projects
Number of Coding References	16
Number of Codes Coding	5
Coverage	4.15%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	04/02/2022 23:00
Coded Text	Coupling between DVC and software artifacts are much stronger than would be expected by chance, with one out of four PRs changing source code, and one out of two PRs changing tests, requiring changes to pipeline files.
Modified on	04/02/2022 23:05

Name	On the Co-evolution of ML Pipelines and Source Code - Empirical Study of DVC Projects
Number of Coding References	16
Number of Codes Coding	5
Coverage	4.15%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	04/02/2022 23:00
Coded Text	Implications to ML versioning tool developers/companies.
Modified on	04/02/2022 23:06

Name	On the Co-evolution of ML Pipelines and Source Code - Empirical Study of DVC Projects
Number of Coding References	16
Number of Codes Coding	5
Coverage	4.15%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	04/02/2022 23:00
Coded Text	Implications to Researchers.
Modified on	04/02/2022 23:06

Name	On the Co-evolution of ML Pipelines and Source Code - Empirical Study of DVC Projects
Number of Coding References	16
Number of Codes Coding	5
Coverage	4.15%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	04/02/2022 23:00
Coded Text	VII. Implications of our findings Implications to ML application developers.
Modified on	04/02/2022 23:05

Name	On the Co-evolution of ML Pipelines and Source Code - Empirical Study of DVC Projects
Number of Coding References	16
Number of Codes Coding	5
Coverage	4.15%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications\Architecture ML system
Created on	03/02/2022 14:39
Coded Text	The growing popularity of machine learn- ing (ML) applications has led to the introduction of software engineering tools such as Data Versioning Control (DVC), MLFlow and Pachyderm that enable versioning ML data, models, pipelines and model evaluation metrics. Since these versioned ML artifacts need to be synchronized not only with each other, but also with the source and test code of the software applications into which the models are integrated, prior findings on co-evolution and coupling between software artifacts might need to be revisited.
Modified on	04/02/2022 22:57

Name	On the Co-evolution of ML Pipelines and Source Code - Empirical Study of DVC Projects
Number of Coding References	16
Number of Codes Coding	5
Coverage	4.15%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	04/02/2022 22:57
Coded Text	
Modified on	04/02/2022 23:02

Name	On the Co-evolution of ML Pipelines and Source Code - Empirical Study of DVC Projects
Number of Coding References	16
Number of Codes Coding	5
Coverage	4.15%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	04/02/2022 22:57
Coded Text	Association Rules. To measure the coupling between DVC files and other project files, we use association rules, similar to earlier papers in this field [10], [11]. Such an association rule is of the form A⇒B, describing the possible coupling of changes to file type A (e.g., “source code”) implying changes to file type B (e.g., “DVC data file”). We use the conventional [21] metrics of “Support” (Supp), “Confidence” (Conf) and “Interest” (Lift) to measure the importance of an association rule. Supp(A) indicates the frequency of appearance of A, while Conf(A⇒B) indicates the percentage of times a change of A will happen together (“is coupled”) with a change of B.
Modified on	04/02/2022 23:03

Name	On the Co-evolution of ML Pipelines and Source Code - Empirical Study of DVC Projects
Number of Coding References	16
Number of Codes Coding	5
Coverage	4.15%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	04/02/2022 22:57
Coded Text	Coupling between DVC and software artifacts are much stronger than would be expected by chance, with one out of four PRs changing source code, and one out of two PRs changing tests, requiring changes to pipeline files.
Modified on	04/02/2022 23:05

Name	On the Co-evolution of ML Pipelines and Source Code - Empirical Study of DVC Projects
Number of Coding References	16
Number of Codes Coding	5
Coverage	4.15%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	04/02/2022 22:57
Coded Text	Pipeline Complexity Analysis In order to estimate the overhead that pipeline descrip- tions represent for data engineers/scientists and developers, we use two measures of pipeline complexity, i.e., McCabe (graph structure complexity of pipelines) and Halstead (effort to understand the textual form of the .dvc pipeline specification files).
Modified on	04/02/2022 23:03

Name	On the Co-evolution of ML Pipelines and Source Code - Empirical Study of DVC Projects
Number of Coding References	16
Number of Codes Coding	5
Coverage	4.15%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	04/02/2022 22:57
Coded Text	The growing popularity of machine learn- ing (ML) applications has led to the introduction of software engineering tools such as Data Versioning Control (DVC), MLFlow and Pachyderm that enable versioning ML data, models, pipelines and model evaluation metrics. Since these versioned ML artifacts need to be synchronized not only with each other, but also with the source and test code of the software applications into which the models are integrated, prior findings on co-evolution and coupling between software artifacts might need to be revisited.
Modified on	04/02/2022 22:57

Name	On the Co-evolution of ML Pipelines and Source Code - Empirical Study of DVC Projects
Number of Coding References	16
Number of Codes Coding	5
Coverage	4.15%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	04/02/2022 22:57
Coded Text	this new generation of tools, this paper aims to empirically study the prevalence of ML pipelines in open source projects, as well as the amount of maintenance effort involved. Previous studies on non-ML projects have shown that frequent changes to source code might require corresponding changes to other software artifacts such as build files [10] and infrastructure-as -code files (IaC) [11] (or vice versa), causing overhead to developers. In the case of ML projects, changes to data and/or model pipelines might induce similar overhead due to the conceptual coupling between data, model and release pipelines
Modified on	04/02/2022 23:01

Name	On the distributed software architecture of a data analysis workflow~ A case study
Number of Coding References	6
Number of Codes Coding	1
Coverage	8.61%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	
Modified on	04/02/2022 22:50

Name	On the distributed software architecture of a data analysis workflow~ A case study
Number of Coding References	6
Number of Codes Coding	1
Coverage	8.61%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	
Modified on	04/02/2022 22:50

Name	On the distributed software architecture of a data analysis workflow~ A case study
Number of Coding References	6
Number of Codes Coding	1
Coverage	8.61%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	
Modified on	04/02/2022 22:50

Name	On the distributed software architecture of a data analysis workflow~ A case study
Number of Coding References	6
Number of Codes Coding	1
Coverage	8.61%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	Hybrid distributed computing software architectures gain great importance in data analysis workflows as the number of available underlying machine learning libraries and data storage systems increase.We argue that there is a need for novel approaches for software architecture designs thatcanenablemachinelearning data analysisworkflows to run on top of different subsystem libraries. To address this need, we propose a hybrid distributed software architecture in this manuscript. The proposed architecture managesmachine learning models for both supervised and unsupervised machine learning data analysis workflows. To show the usability of the proposed architecture, we implement a prototype for the banking sector as a case study. The prototype application includes two data analysis workflows: a workflow for predicting the loan usage tendency of customers, and a workflow for clustering the customers based on the usage patterns of banking loans. The prototype is tested on a large scale banking dataset. Performance tests were carried out to investigate the performance in terms of both responsiveness and scalability of the system. The results obtained reveal the usability of the proposed architecture.
Modified on	04/02/2022 22:44
Name	On the distributed software architecture of a data analysis workflow~ A case study
Number of Coding References	6
Number of Codes Coding	1
Coverage	8.61%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	The software architecture has multiple components as shown in Figure 2. These include Data preprocessing Module, Model Training Man- agement Module, Credit Usage Tendency Prediction Model Manager, Customer Segmentation Manager, Monitoring, and Reporting Module. The semantics for the programming interface of the proposed software architecture is as follows. We use the Predictive Model Markup Language to define and express the machine learning models. Also, we utilize a RESTful web service programming interface to provide various functions within the proposed architecture: (a) API for credit usage tendency prediction, (b) API for clustering customers based on loan usage activities, (c) API for data preprocessing, (d) API for managing machine learning models, (e) API for analytics functionalities. The user will utilize the proposed API to filter the raw transaction datasets and create machine learning ready datasets. Here, various data mining functions such as missing data imputation, outlier removal and normalization will be applied. Then, the machine learning models will be created and trained by utilizing the proposed model management API.
Modified on	04/02/2022 22:49

Name	On the distributed software architecture of a data analysis workflow~ A case study
Number of Coding References	6
Number of Codes Coding	1
Coverage	8.61%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	Upon examination the literature, we observe different distributed software architectures designed for managing data or metadata.16-18 Exam- ples of these architectures are designed in for different domains such as provenance management,16 social networks,17 and cloud computing.18 Thereexists also somework in workflow-based distributed data analysis studies.AnXML-basedworkflowmechanismcandramatically improve the orchestration process for data analysis.19 Amodular structure to orchestrate data analysis benefits better distribution.20 In this study, we focus on a hybrid and generic distributed software architecture to manage machine learning models to be used for classification and clustering in banking sector. Our earlier work on distributed software architecture for finding the next best action in the banking sector was published in a conference paper.21 Moreover, our early work on distributed workflows was presented at another conference.22 There exists a number of studies focusing on distributed software architecture that are designed to manage and process large scale meta- data in different domains such as social media,23,24 weather forecasting,25 cloud monitoring systems,18 and information systems.26 These studies mainly focus on designing and implementing scalable and high-performance systems that work with negligible processing overheads. In our study, we are also interested in designing and implementing a distributed software architecture. Different from previous work, our proposed framework is designed inway that it can employ different machine learning business workflows particularly for banking sector, while providing scalability, high performance,negligible processingoverheads. Thereare studies that focusonstreamingmachine learning algorithmson big data.27 In this study, we studymachine learning algorithms such as xgboost,28 lightgbm,29 kmeans,30 which are designed for batch-data processing on the big data processing platforms. There are studies that compare and contrasts various different distributed computing approaches including cluster computing, grid computing and cloud computing.31 Different from these approaches, in this study, we focus on micro-service based distributed computing. Kuwil et al. utilizes a critical distance function in the clustering algorithms.32 In this study, to evaluate the performance of the clusteringworkflow, we also use clustering algorithms. However, in this study, we utilize Euclidean distance function for clustering algorithms.
Modified on	04/02/2022 22:49

Name	On the experiences of adopting automated data validation in an industrial machine learning project
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.61%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	06/01/2022 16:31
Coded Text	
Modified on	04/02/2022 22:37

Name	On the experiences of adopting automated data validation in an industrial machine learning project
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.61%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	06/01/2022 16:31
Coded Text	
Modified on	04/02/2022 22:40

Name	On the experiences of adopting automated data validation in an industrial machine learning project
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.61%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	06/01/2022 16:31
Coded Text	1) Defining data quality tests: a data validation process for ML projects requires having an overview of the level of data (feature, dataset, cross-dataset, data stream) at which
Modified on	04/02/2022 22:38

Name	On the experiences of adopting automated data validation in an industrial machine learning project
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.61%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	06/01/2022 16:31
Coded Text	2) Providing actionable feedback: communicating the out- put results of data validation in terms of warnings and validation report requires a careful design decision of what and
Modified on	04/02/2022 22:39

Name	On the experiences of adopting automated data validation in an industrial machine learning project
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.61%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	06/01/2022 16:31
Coded Text	3) Treating data errors with similar rigor as code: similar to software bugs, data errors should be documented, tracked and resolved. Like software unit tests that try to test atomic components in codebase, data validation tests allow designers to quantify the performance of ML models adhering to some specific properties found in ML training datasets. Therefore, structuring data validation tests around the properties of data that the ML model expects to acquire serves as one such approach. In our study, collaborative work, which is important in ML projects,
Modified on	04/02/2022 22:39

Name	On the experiences of adopting automated data validation in an industrial machine learning project
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.61%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	06/01/2022 16:31
Coded Text	B. Benefits (RQ2) The benefits of adopting data validation process and tool for ML projects include: 1) minimization of manual effort in data preparation; 2) early identification of data errors; 3) a testing approach to ML enabled software systems.
Modified on	04/02/2022 22:39

Name	On the experiences of adopting automated data validation in an industrial machine learning project
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.61%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	06/01/2022 16:31
Coded Text	Deequ is a tool developed by Amazon Research for automating data quality verification. Deequ allows its users to define ‘unit tests’ for data and combines common quality constraints with user-defined validation code [3]. To perform data validation, the tool relies on declarative user-defined checks on the dataset, for example, isComplete and isUnique checks. The declarative user-defined checks are converted into computations of metrics on data, e.g. different statistical analysis, that can be used to evaluate constraints. After executing data quality verification, the tool reports constraints that succeeded and failed, including information of the computed metric. Although Deequ provides overall data quality report, the tool does not fetch individual records that did not succeed the validations. At Google, the TensorFlow data validation tool [2] is used to validate trillions of training and serving examples per day. To perform data validation, the tool relies on a data schema
Modified on	04/02/2022 22:36

Name	On the experiences of adopting automated data validation in an industrial machine learning project
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.61%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	06/01/2022 16:31
Coded Text	Finally, Data Sentinel [5] is a data validation platform developed at LinkedIn. To perform data validation, users use a well-structured configuration file to specify data checks that are desired for specific features. This simplifies the need to write and maintain data checking code. For a given dataset, Data Sentinel computes statistical summaries of the specified features and evaluates the assertions. Eventually, the summaries and validation results are recorded into a dataset profile and validation report. Overall, studies do not provide experiences of adopting a data validation process and tool by development different teams. The tools presented are also developed by dedicated teams in large companies with several years of experience in deploying to production several ML projects. The few studies that share experiences show slow and poor early adoption with several development iterations [5]. For companies that are in the early stages of deploying ML components to production and from the embedded domain, learning from these experiences is important to help systematize the adoption with minimum resources. This is because the data validation process and tools consume huge amounts of engineering resources and maintenance [5].
Modified on	04/02/2022 22:37

Name	On the experiences of adopting automated data validation in an industrial machine learning project
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.61%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	06/01/2022 16:31
Coded Text	IV. RESULTS This section discusses experiences of adopting a data validation process and tool for ML projects by data science teams at a large software-intensive company in telecommunication domain. The experiences are shaped in form of Best Practices, Benefits and Barriers of adopting data validation in ML projects. A. Best Practices (RQ1) Best practices of adopting a data validation process and tool for ML projects are classified in three groups: 1) defining data quality tests, 2) providing actionable feedback, and 3) treating data errors with similar rigor as code. Results
Modified on	04/02/2022 22:38

Name	On the experiences of adopting automated data validation in an industrial machine learning project
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.61%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	06/01/2022 16:31
Coded Text	V. DATA VALIDATION FRAMEWORK (DVF) Based on the experiences, we propose a data validation framework (DVF) shown in Figure 2 that systematizes the adoption of data validation in ML projects. We group important aspects in the DVF into: A) validation process, B) validation artefacts, C) data validation types, D) data validation tool setup, and E) feedback and mitigation strategy.
Modified on	04/02/2022 22:40

Name	On the experiences of adopting automated data validation in an industrial machine learning project
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.61%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	06/01/2022 16:31
Coded Text	While several problems in existing data validation tools can be identified, including implementation errors and decoupling from data cleaning capabilities [13], much focus is on the implementations of these different tools [2, 11, 3, 5]. There is limited reporting on the experiences of adopting the data validation process. The experiences are especially useful for teams that are in the early stages of deploying to production ML-enabled software systems. Adopting the data validation process and tool demands huge engineering resources for development and maintenance [5]. Furthermore, there are no well-established guidelines for establishing a data validation
Modified on	04/02/2022 22:34

Name	On the experiences of adopting automated data validation in an industrial machine learning project
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.61%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	06/01/2022 16:25
Coded Text	A tool called Data Linter (adopting the concept of code lint in SE) is used to automatically inspect training data and suggest ways in which features can be transformed into suitable data representation [11]. The assumption is that data can be valid but not in a representation that the ML model can best learn from, e.g. a timestamp encoded as a string. Three types of data lints that can be detected by the tool are miscoding lints (e.g. number as string), lints for outliers and scaling (e.g. tailed distribution detectors) and packaging error lints (e.g. duplicate values). Technically, the data linter tool inspects training dataset's summary statistics, examines individual examples and names given to the data features. One main limitation of the data linter tool is that it does not allow users to configure and select a set of specific lint detectors to run. As a result, the latter affect tool performance especially for large and medium scale dataset [11].The tool does not provide support for data transformation, rather the user has to manually perform the transformations. This is in addition to the lack of proper documentation and discontinued support of the data linter tool 2.
Modified on	04/02/2022 22:36

Name	On the experiences of adopting automated data validation in an industrial machine learning project
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.61%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	06/01/2022 16:25
Coded Text	C. Barriers (RQ3) The barriers of adopting data validation include 1) limited flexibility ofdata validation tool e.g. in terms ofease ofadding new tests, and 2) limited support for the existing technology stack of ML system development process while also ensuring low learning curve.
Modified on	04/02/2022 22:39

Name	On the experiences of adopting automated data validation in an industrial machine learning project
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.61%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	06/01/2022 16:25
Coded Text	Data errors are a common challenge in machine learning (ML) projects and generally cause significant performance degradation in ML-enabled software systems. To ensure early detection of erroneous data and avoid training ML models using bad data, research and industrial practice suggest incorporating a data validation process and tool in ML system development process. Aim: The study investigates the adoption of a data validation process and tool in industrial ML projects. The data validation process demands significant engineering resources for tool development and maintenance. Thus, it is important to identify the best practices for their adoption especially by development teams that are in the early phases of deploying ML-enabled software systems.
Modified on	04/02/2022 22:31
Name	On the experiences of adopting automated data validation in an industrial machine learning project
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.61%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	06/01/2022 16:25
Coded Text	Data errors are common and can be difficult to detect when developing and operating ML-enabled software systems [2, 3, 4]. For companies, data errors can result in significant losses in business value. For example, LinkedIn observed financial losses and had to put huge efforts to detect data errors in their job recommendations platform [5]. Poor visibility of complex data dependencies, errors in application code, drifts in sensor data, gaps in data due to network connection problems are among the causes of data errors [6, 7, 8]. Understanding the different types of data errors and their effects on ML projects is important because literature shows that unnecessary data cleaning can be wasteful and harmful to the training of ML models [9]. To handle data errors in ML projects, research and industrial practice suggest integration of data validation tools into the development process of ML-enabled systems1 instead of only relying on data scientists to manually check the quality of the data [10, 2, 3, 11, 12]. Important data quality dimensions of consideration are with respect to accuracy, completeness, consistency, timeliness [3, 13]. The data validation tools are particularly useful when dealing continuously with large scale data [2, 11, 3, 5]. The data validation process is also considered an approach to testing ML-enabled software systems [14].
Modified on	04/02/2022 22:34

Name	On the experiences of adopting automated data validation in an industrial machine learning project
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.61%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	06/01/2022 16:25
Coded Text	Data validation process in ML projects In most commercial ML systems, deployed ML models are continuously retrained in order to adapt to environmental changes [2, 15]. When retraining the ML models, new training data collected at inference time can have different distribution due to various reasons, like bugs in application code [2, 16]. Differences in data distribution at training and inference, also called training-serving skew, is one form of data errors in ML projects. When the erroneous data is not detected, ML models are retrained on problematic data and can result to performance degradation of an ML system [2, 16]. Furthermore, it is rare that training datasets collected from many different sources at different time periods would always have the same exact structure and distribution [16]. Data validation in ML projects is the process of ensuring the high quality of data that is fed into the ML algorithm(s). The aim is to continuously check and monitor the data in order to assess its quality and identify underlying issues in data quality [2, 3]. Recently, studies have demonstrated that
Modified on	04/02/2022 22:35

Name	On the experiences of adopting automated data validation in an industrial machine learning project
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.61%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	06/01/2022 16:25
Coded Text	Data validation tools in ML projects Ehrlinger et. al. [13] conducted a state-of-the-art survey of data quality systems (both commercial and open-source), and investigated their measurement and monitoring functionalities in order to determine how data quality is measured and monitored. While their survey did not include other tools identified by this study (discussed in next paragraphs), several limitations are reported, including implementation errors and narrow coverage of data quality metrics for important quality dimensions [13]. In addition, their study did not report the actual use of the data quality tools in industrial ML projects [13]. We identify and present studies that discuss the use of data validation tools in industrial ML projects.
Modified on	04/02/2022 22:35

Name	On the experiences of adopting automated data validation in an industrial machine learning project
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.61%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	06/01/2022 16:25
Coded Text	Overall, studies do not provide experiences of adopting a data validation process and tool by development different teams. The tools presented are also developed by dedicated teams in large companies with several years of experience in deploying to production several ML projects. The few studies that share experiences show slow and poor early adoption with several development iterations [5]. For companies that are in the early stages of deploying ML components to production and from the embedded domain, learning from these experiences is important to help systematize the adoption with minimum resources. This is because the data validation process and tools consume huge amounts of engineering resources and maintenance [5].
Modified on	04/02/2022 22:37
Name	On-Device Learning Systems for Edge Intelligence~ A Software and Hardware Synergy Perspective
Number of Coding References	14
Number of Codes Coding	3
Coverage	7.23%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	04/02/2022 22:14
Coded Text	
Modified on	04/02/2022 22:21

Name	On-Device Learning Systems for Edge Intelligence~ A Software and Hardware Synergy Perspective
Number of Coding References	14
Number of Codes Coding	3
Coverage	7.23%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	04/02/2022 22:14
Coded Text	Although many in-cloud learning approaches have been proposed, they may not well match the emerging trend that enables edge-intelligent features running on personal devices by using user local data only [7], [8]. In this scenario, the main drawbacks of in-cloud learning can be summarized into three aspects. The first is the privacy and security issue. As the intermediate training results and model parameters need to be transmitted through the network and stored in the cloud, the user data containing sensitive information can be easily eavesdropped and user privacy will be exposed to the cloud carrier [9]. The second problem is that the in-cloud learning cannot provide personalized model for the user as the conventional training procedure aims to provide a unified model by aggregating the results from plenty of workers, i.e., distributed training in a data-parallel manner [10]. Finally, although it is possible to conduct the application processing in the cloud and fetch the final result to the user to meet the personalized requirement, the data transmission may yield an unacceptable network transmission time, especially in the bandwidth-limited environment. This high latency will hamper the real-time model update and lead to a low processing throughput [11].
Modified on	04/02/2022 22:17

Name	On-Device Learning Systems for Edge Intelligence~ A Software and Hardware Synergy Perspective
Number of Coding References	14
Number of Codes Coding	3
Coverage	7.23%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	04/02/2022 22:14
Coded Text	However, this in-cloud computing scheme cannot satisfy the demands of emerging edge intelligence scenarios, including providing personalized models, protecting user privacy, adapting to real-time tasks, and saving resource cost. In order to conquer the limitations of conventional in-cloud computing, there comes the rise of on-device learning, which makes the end-to-end ML procedure totally on user devices, without unnecessary involvement of the cloud. In spite of the promising advantages of on-device learning, implementing a highperformance on-device learning system still faces with many severe challenges, such as insufficient user training data, backward propagation (BP) blocking, and limited peak processing speed.
Modified on	04/02/2022 22:14

Name	On-Device Learning Systems for Edge Intelligence~ A Software and Hardware Synergy Perspective
Number of Coding References	14
Number of Codes Coding	3
Coverage	7.23%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	04/02/2022 22:20
Coded Text	
Modified on	04/02/2022 22:21

Name	On-Device Learning Systems for Edge Intelligence~ A Software and Hardware Synergy Perspective
Number of Coding References	14
Number of Codes Coding	3
Coverage	7.23%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	04/02/2022 22:20
Coded Text	
Modified on	04/02/2022 22:24

Name	On-Device Learning Systems for Edge Intelligence~ A Software and Hardware Synergy Perspective
Number of Coding References	14
Number of Codes Coding	3
Coverage	7.23%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	04/02/2022 22:20
Coded Text	2) Objectives: Building a high-performance on-device learning system requires a clear understanding of why we need on-device learning and what are the advantages on-device learning can bring. Briefly, the objectives of on-device learning can be summarized as follows. Resource Saving: Due to the resource-constrained envi- ronment of the edge devices, a high-performance on-device learning system needs to reduce the resource cost as much as possible, including the hardware-level computation overhead, communication-level network traffic, and energy-level consumption. Overall, resource saving is one of the most essential demands to deploy on-device learning [49]. Personalized Model: As on-device learning applications are executed totally based on the user local data, the corresponding learning frameworks cannot be designed as usual distributed ML systems in the cloud that aim to provide a general model working for an ensemble of tasks. Instead, on-device learning should provide customized models, which are tailed to meet the user's preference, i.e., different users own different models based on their needs [11]. Privacy Protection: As the training procedure only happens on the edge devices and the final model contains the sensitive information of user preference, the data privacy should be carefully protected [74]. All the training data should be stored and accessed under the user's control, without any unnecessary data sharing with other machines. Online Learning: As the user data are generated on the device continuously, the systems should be able to handle the learning procedure online and incrementally retrain the model according to the latest user data, so as to make the customized model up to date [75]. Summary: The above four properties closely rely on the co-design of learning algorithms and system implementation, where several challenges need to be addressed. We will discuss them in the next. 3) Challenges: Although on-device learning shows promis- ing advantages to deploy modern ML applications in the edge environment, implementing the relevant frameworks in the realistic scenario is not easy. In order to guide the future research of the on-device learning and corresponding system design, we summarize the most significant challenges as follows. Insufficient User Data: As the data on the edge devices are usually specific to each user and cannot exchange with others due to the privacy concern, the available training data are not sufficient enough to conduct the model training as traditional distributed ML does. This condition is similar to the few-shot learning [16] with relatively few user data. Generally, it is hard to ensure a model with a good generalization based on smallscale training data set. Therefore, on-device learning cannot simply follow conventional training methods that learn from scratch. This property requires the systems to learn customized
Modified on	04/02/2022 22:26

Name	On-Device Learning Systems for Edge Intelligence~ A Software and Hardware Synergy Perspective
Number of Coding References	14
Number of Codes Coding	3
Coverage	7.23%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	04/02/2022 22:20
Coded Text	features from the small-scale data set. We will mainly introduce the techniques of knowledge distillation (Section III-B) and model fine-tuning (Section III-C) to tackle this challenge. Backward Propagation Blocking: BP is the key step to cal- culate the gradients and update the model parameter. However, when training a deep model on device, conventional chain rulebased gradient calculation may be stuck in some layers and blocks the BP to the next layers. For example, as to the ondevice learning using the Core ML framework [76] on iPhone, only the FC and CONV layers can participate in the gradient calculation, while the weights based on the operations of batch normalization, embeddings, bias, and scales cannot be updated. Besides, models based on RNN, LSTM, and GRU are also not well supported. Therefore, the systems need to cover more layers and model properties when deploying ondevice learning on mobile phones. This requirement is closely related to the hardware implementation (Section V) on different edge platforms. We will cover the aspects of embedded memory, computational primitives, low-level instructions, and communication scheme to present a comprehensive discussion. Limited Peak Speed: Different from the servers in the cloud environment, edge devices, specially the mobile phones, are not built for running at top speed for hours continuously to finish a task. However, the model training procedure requires a huge amount of iterations until the convergence, which will easily generate much heat and invoke the CPU cooling process to protect the expensive hardware. The most common way to restrict the hardware temperature is to control the CPU's clock in a lower frequency, which will definitely slow down the processing speed. Besides, running at peek speed for a long time will exhaust the battery energy of mobile devices [10]. As a result, conducting the learning applications on devices requires more efficient management of the limited resources, where compressing model size and simplifying arithmetic operations are two key steps. We will mainly discuss the topics of parameter pruning for model compression (Section III-A), loss regularization for network sparsification (Section IV-A), and data quantization for processing acceleration (Section IV-B) to reduce the computational overhead. Summary: The above-mentioned challenges are the most critical issues to implement on-device learning systems. In the rest of this survey, we will discuss the related technologies to address these challenges.
Modified on	04/02/2022 22:26

Name	On-Device Learning Systems for Edge Intelligence~ A Software and Hardware Synergy Perspective
Number of Coding References	14
Number of Codes Coding	3
Coverage	7.23%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	04/02/2022 22:20
Coded Text	model update may be blocked as not all the neurons or layers are updatable in the on-device environment [18]. Moreover, the peak processing speed is often restricted by the operating system because the devices are not built for running at full speed continuously in a long time, i.e., the generated heat will slow down the processors and exhaust the limited battery.
Modified on	04/02/2022 22:21

Name	On-Device Learning Systems for Edge Intelligence~ A Software and Hardware Synergy Perspective
Number of Coding References	14
Number of Codes Coding	3
Coverage	7.23%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	04/02/2022 22:20
Coded Text	The basic objective of on-device learning is to swiftly produce the personalized model with good quality in the resource-constrained environment, e.g., the Tensorflow Lite [14] and PyTorch Mobile [15]. However, achieving this target requires a co-design of learn- ing algorithms and system implementation, where several challenges need to be addressed. First, an edge device usually owns few user data [16], which are not sufficient enough and may lead to model overfitting [17]. Besides, the BP for
Modified on	04/02/2022 22:20

Name	On-Device Learning Systems for Edge Intelligence~ A Software and Hardware Synergy Perspective
Number of Coding References	14
Number of Codes Coding	3
Coverage	7.23%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:15
Coded Text	
Modified on	04/02/2022 22:22

Name	On-Device Learning Systems for Edge Intelligence~ A Software and Hardware Synergy Perspective
Number of Coding References	14
Number of Codes Coding	3
Coverage	7.23%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:15
Coded Text	
Modified on	04/02/2022 22:23

Name	On-Device Learning Systems for Edge Intelligence~ A Software and Hardware Synergy Perspective
Number of Coding References	14
Number of Codes Coding	3
Coverage	7.23%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:15
Coded Text	
Modified on	04/02/2022 22:28

Name	On-Device Learning Systems for Edge Intelligence~ A Software and Hardware Synergy Perspective
Number of Coding References	14
Number of Codes Coding	3
Coverage	7.23%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:15
Coded Text	On-Device Learning As modern ML applications often require a huge amount of computational resources, conventional implementation of the learning approaches usually relies on the power of cloudbased machines deployed in the distributed manner. However, with the rapid development of the hardware processing speed and memory capacity, it is possible to handle some small-scale learning applications by the on-device hardware itself, e.g., the Apple Face ID [69] and Microsoft Flower Recognition [70]. Here, we will introduce the definition of on-device learning, the objective of these applications, and the crucial challenges in practice.
Modified on	04/02/2022 22:26

Name	On-Device Learning Systems for Edge Intelligence~ A Software and Hardware Synergy Perspective
Number of Coding References	14
Number of Codes Coding	3
Coverage	7.23%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:15
Coded Text	This survey presents a software and hardware synergy of on-device learning techniques, covering the scope of model-level neural network design, algorithm-level training optimization, and hardware-level instruction acceleration. We hope this survey could bring fruitful discussions and inspire the researchers to further promote the field of edge intelligence.
Modified on	04/02/2022 22:15

Name	OneLabeler~ A Flexible System for Building Data Labeling Tools
Number of Coding References	6
Number of Codes Coding	2
Coverage	1.66%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	10/02/2022 12:05
Coded Text	building data labeling tools. OneLabeler is designed with reusability and flexibility in mind. It enables visual programming to compose and configure software modules. Developers can build a data labeling tool by creating a workflow graph using built-in implementations of the conceptual modules. In a created labeling tool, interface modules and algorithm modules can be instantiated as human, machine, or mixed computation procedures. The constraints on composing modules are embedded in a static program checker to check the workflow graph and verify the feasibility ofthe created
Modified on	16/02/2023 11:04

Name	OneLabeler~ A Flexible System for Building Data Labeling Tools
Number of Coding References	6
Number of Codes Coding	2
Coverage	1.66%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	10/02/2022 12:05
Coded Text	Labeled datasets are essential for supervised machine learning. Various data labeling tools have been built to collect labels in different usage scenarios. However, developing labeling tools is timeconsuming, costly, and expertise-demanding on software development. In this paper, we propose a conceptual framework for data labeling and OneLabeler based on the conceptual framework to support easy building of labeling tools for diverse usage scenarios. The framework consists of common modules and states in labeling tools summarized through coding of existing tools. OneLabeler supports configuration and composition of common software modules through visual programming to build data labeling tools. A module can be a human, machine, or mixed computation procedure in data labeling. We demonstrate the expressiveness and utility of the system through ten example labeling tools built with OneLabeler. A user study with developers provides evidence that OneLabeler supports efficient building of diverse data labeling tools.
Modified on	16/02/2023 11:04

Name	OneLabeler~ A Flexible System for Building Data Labeling Tools
Number of Coding References	6
Number of Codes Coding	2
Coverage	1.66%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	10/02/2022 12:05
Coded Text	labeling tool. A labeling tool built with OneLabeler can be exported as an installer of the tool for sharing. To demonstrate the expressiveness of the proposed framework and system, we present a case study of ten tools built with OneLabeler. These tools cover various usage scenarios, including a set of typical labeling tools for different data types and labeling tasks, a classification tool for a customized webpage data type, a machineaided multi-task text labeling tool, a mixed-initiative image classification tool, and prototyping an interactive machine learning system for chart image reverse engineering. To examine OneLabeler's usability, we conduct a user study in which developers are asked to accomplish four tasks on building data labeling tools with OneLabeler. The study results indicate that OneLabeler is easy to learn and use, enabling developers to efficiently build different data labeling tools. This paper has the following major contributions: • We propose a conceptual framework that illuminates common conceptual modules and composition constraints in data labeling. • We develop the OneLabeler system based on the framework to support easy building of diverse data labeling tools. • We conduct an extensive case study to demonstrate the capability of OneLabeler in creating diverse data labeling tools, as well as a user study to validate the usability of OneLabeler.
Modified on	16/02/2023 11:05

Name	OneLabeler~ A Flexible System for Building Data Labeling Tools
Number of Coding References	6
Number of Codes Coding	2
Coverage	1.66%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	10/02/2022 12:05
Coded Text	Meanwhile, labeling tools for different labeling tasks share commonalities. To alleviate the burden of building labeling tools, we advocate a modular composable design that extracts the commonalities among different labeling tools to enable easy customization and extension. In this paper, we present a conceptual data labeling framework with a modular composable design. We have identified eight types of common modules through inductive coding of modules in existing data labeling tools. The framework consists of common conceptual modules and constraints in composing these modules. In the framework, a data labeling tool is modeled as a graph denoting its workflow. In the graph, modules are nodes, and the execution order of the modules is encoded with directed edges. Based on the framework, we develop OneLabeler, a system for
Modified on	16/02/2023 11:04

Name	OneLabeler~ A Flexible System for Building Data Labeling Tools
Number of Coding References	6
Number of Codes Coding	2
Coverage	1.66%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	03/02/2022 10:07
Coded Text	Labeled datasets are essential for supervised machine learning. Various data labeling tools have been built to collect labels in different usage scenarios. However, developing labeling tools is timeconsuming, costly, and expertise-demanding on software development. In this paper, we propose a conceptual framework for data labeling and OneLabeler based on the conceptual framework to support easy building of labeling tools for diverse usage scenarios. The framework consists of common modules and states in labeling tools summarized through coding of existing tools. OneLabeler supports configuration and composition of common software modules through visual programming to build data labeling tools. A module can be a human, machine, or mixed computation procedure in data labeling. We demonstrate the expressiveness and utility of the system through ten example labeling tools built with OneLabeler. A user study with developers provides evidence that OneLabeler supports efficient building of diverse data labeling tools.
Modified on	16/02/2023 11:03

Name	OneLabeler~ A Flexible System for Building Data Labeling Tools
Number of Coding References	6
Number of Codes Coding	2
Coverage	1.66%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	03/02/2022 10:07
Coded Text	ONELABELER Based on the requirements described in Section 3, we propose the OneLabeler system for building labeling tools. OneLabeler enables developers to visually program (R4) data labeling tools by composing software modules into a workflow (R2). A created labeling tool can be exported as an installer. The conceptual modules identified in Section 4 inform the API design for data labeling modules in OneLabeler (R1). A developer can reuse a collection of built-in modules and templates (R3) or customize on demand. OneLabeler’s visual programming environment supports static checking (R5) of user-created workflows to assist debugging and guide towards
Modified on	16/02/2023 11:05

Name	OneLabeler~ A Flexible System for Building Data Labeling Tools Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Overton~ A Data System for Monitoring and Improving Machine-Learned Products
Number of Coding References	7
Number of Codes Coding	4
Coverage	12.17%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	(1) Code-free Deep Learning In Overton-based systems, engineers focus exclusively on fine-grained monitoring of their application quality and improving supervision–not tweaking deep learning models. An Overton engineer does
Modified on	24/02/2022 09:56

Name	Overton~ A Data System for Monitoring and Improving Machine-Learned Products
Number of Coding References	7
Number of Codes Coding	4
Coverage	12.17%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	Overton provides the engineer with abstractions that allow them to build, maintain, and monitor their application by manipulating data files–not custom code. Inspired by relational systems, supervision (data) is managed separately from the model (schema). Akin to traditional logical independence, Overton's schema provides model independence: serving code does not change even when inputs, parameters, or resources of the model change. The schema changes very infrequently–many production services have not updated their schema in over a year. Overton takes as input a schema whose design goal is to support rich applications from modeling to automatic deployment. In more detail, the schema has two elements: (1) data payloads similar to a relational schema, which describe the input data, and (2) model tasks, which describe the tasks that need to be accomplished. The schema defines the input, output, and coarse-grained data flow of a deep learning model. Informally, the schema defines what the model computes but not how the model computes it: Overton does not prescribe architectural details of the underlying model (e.g., Overton is free to embed sentences using an LSTM or a Transformer) or hyperparameters, like hidden state size. Additionally, sources of supervision are described as data–not in the schema–so they are free to rapidly evolve. As shown in Figure 1, given a schema and a data file, Overton is responsible to instantiate and train a model, combine supervision, select the model's hyperparameters, and produce a production-ready binary. Overton compiles the schema into a (parameterized) TensorFlow or PyTorch program, and performs an architecture and hyperparameter search. A benefit of this compilation approach is that Overton can use standard toolkits to monitor training (TensorBoard equivalents) and to meet service-level agreements (Profilers). The models and metadata are written to an S3-like data store that is accessible from the production infrastructure. This has enabled model retraining and deployment to be nearly automatic, allowing teams to ship products more quickly.
Modified on	24/02/2022 09:56

Name	Overton~ A Data System for Monitoring and Improving Machine-Learned Products
Number of Coding References	7
Number of Codes Coding	4
Coverage	12.17%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	technique led to state-of-the-art results on natural language benchmarks including GLUE and SuperGLUE [31].3 (2) Multitask Learning Overton was built to natively support multitask learning [2,24,26] so that all model tasks are concurrently predicted. A key benefit is that Overton can accept supervision at whatever granularity (for whatever task) is available. Overton models often perform ancillary tasks like part-of-speech tagging or typing. Intuitively, if a representation has captured the semantics of a query, then it should reliably perform these ancillary tasks. Typically, ancillary tasks are also chosen either to be inexpensive to supervise. Ancillary task also allow developers to gain confidence in the model's predictions and have proved to be helpful for aids for debugging errors. (3) Weak Supervision Applications have access to supervision of varying quality and combining this contradictory and incomplete supervision is a major challenge. Overton uses techniques from Snorkel [23] and Google's Snorkel DryBell [12], which have studied how to combine supervision in theory and in software. Here, we describe two novel observations from building production applications: (1) we describe the shift to applications which are constructed almost entirely with weakly supervised data due to cost, privacy, and cold-start issues, and (2) we observe that weak supervision may obviate the need for popular methods like transfer learning from massive pretrained models, e.g., BERT [8]–on some production workloads, which suggests that a deeper trade-off study may be illuminating. In summary, Overton represents a first-of-its kind machine-learning lifecycle management system
Modified on	24/02/2022 09:56

Name	Overton~ A Data System for Monitoring and Improving Machine-Learned Products
Number of Coding References	7
Number of Codes Coding	4
Coverage	12.17%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	We briefly cover some of the design decisions in Overton. Design for Weakly Supervised Code As described, weakly supervised machine learning is often the dominant source of supervision in many machine learning products. Overton uses ideas from Snorkel [23] and Google's Snorkel Drybell [12] to model the quality of the supervision. The design is simple: lineage is tracked for each source of information. There are production systems with no traditional supervised training data (but they do have such data for validation). This is important in privacy-conscious applications. Modeling to Deployment In many production teams, a deployment team is distinct from the modeling team, and the deployment team tunes models for production. However, we noticed quality regressions as deployment teams have an incomplete view of the potential modeling tradeoffs. Thus, Overton was built to construct a deployable production model. The runtime performance of the model is potentially suboptimal, but it is well within production SLAs. By encompassing more of the process, Overton has allowed faster model turn-around times. Use Standard Tools for the ML Workflow Overton compiles the schema into (many versions of) TensorFlow, CoreML, or PyTorch. Whenever possible, Overton uses a standard toolchain. Using standard tools, Overton supports distributed training, hyperparameter tuning, and building servable models. One unanticipated benefit of having both backends was that different resources are often available more conveniently on different platforms. For example, to experiment with pretrained models, the Huggingface repository [14] allows quick experimentation–but only in PyTorch. The TensorFlow production tools are unmatched. The PyTorch execution mode also allows REPL and in-Jupyternotebook debugging, which engineers use to repurpose elements, e.g., query similarity features. Even if a team uses a single runtime, different runtime services will inevitably use different versions of that runtime, and Overton insulates the modeling teams from the underlying changes in production serving infrastructure. Model Independence and Zero-code Deep Learning A major design choice at the outset of the project was that domain engineers should not be forced to write traditional deep learning modeling code. Two years ago, this was a contentious decision as the zeitgeist was that new models were frequently published, and this choice would hamstring the developers. However, as the pace of new model building blocks has slowed, domain engineers no longer feel the need to fine-tune individual components at the level of TensorFlow. Ludwig9 has taken this approach and garnered adoption. Although developed separately, Overton's schema looks very similar to Ludwig's programs and from conversations with the developers, shared similar motivations. Ludwig, however, focused on the one-off model building process not the management of the model lifecycle. Overton itself only supports text processing, but we are prototyping image, video, and multimodal applications. Engineers are Comfortable with Automatic Hyperparameter Tuning Hyperparameter tuning is conceptually important as it allows Overton to avoid specifying parameters in the schema for the model builder. Engineers are comfortable with automatic tuning, and first versions of all Overton systems are tuned using standard approaches. Of course, engineers may override the search: Overton is used to produce servable models, and so due to SLAs, production models often pin certain key parameters to avoid tail performance regressions. 9https://uber.github.io/ludwig/ 7 Resourcing Error Reduction Amount of Weak Supervision High 65% (2.9×) Medium 82% (5.6×) Medium 72% (3.6×) Low 40% (1.7×) 80% 96% 98% 99% Figure 3: For products at various resource levels, percentage (and factor) fewer errors of Overton system makes compared to previous system, and the percentage of weak supervision of all supervision. Make it easy to manage ancillary data products Overton is also used to produce back-end data products (e.g., updated word or multitask embeddings) and multiple versions of the same model. Inspired by HuggingFace [14], Overton tries to make it easy to drop in new pretrained embeddings as they arrive: they are simply loaded as payloads. Teams use multiple models to train a "large" and a "small" model on the same data. The large model is often used to

	populate caches and do error analysis, while the small model must meet SLA requirements. Overton makes it easy to keep these two models synchronized. Additionally, some data products can be expensive to produce (on the order of ten days), which means they are refreshed less frequently than the overall product. Overton does not have support for model versioning, which is likely a design oversight.
Modified on	09/07/2022 20:05
Name	Overton~ A Data System for Monitoring and Improving Machine-Learned Products
Number of Coding References	7
Number of Codes Coding	4
Coverage	12.17%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	04/02/2022 22:03
Coded Text	In summary, Overton represents a first-of-its kind machine-learning lifecycle management system that has a focus on monitoring and improving application quality. A key idea is to separate the model and data, which is enabled by a code-free approach to deep learning. Overton repurposes ideas from the database community and the machine learning community to help engineers in supporting the lifecycle of machine learning toolkits. This design is informed and refined from use in production systems for over a year in multiple machine-learned products.
Modified on	04/02/2022 22:03
Name	Overton~ A Data System for Monitoring and Improving Machine-Learned Products
Number of Coding References	7
Number of Codes Coding	4
Coverage	12.17%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 16:13
Coded Text	In the life cycle of many production machine-learning applications, maintaining and improving deployed models is the dominant factor in their total cost and effectiveness–much greater than the cost of de novo model construction. Yet, there is little tooling for model life-cycle support. For such applications, a key task for supporting engineers is to improve and maintain the quality in the face of changes to the input distribution and new production features. This work describes a new style of data management system called Overton that provides abstractions to support the model life cycle by helping build models, manage supervision, and monitor application quality.1 Overton is used in both near-real-time and backend production applications. However, for concreteness, our running example is a product that answers factoid queries, such as “how tall is the president of the united states?” In our experience, the engineers who maintain such machine learning products face several challenges on which they spend the bulk of their time.
Modified on	24/02/2022 09:54

Name	Overton~ A Data System for Monitoring and Improving Machine-Learned Products
Number of Coding References	7
Number of Codes Coding	4
Coverage	12.17%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	04/02/2022 21:53
Coded Text	Fine-grained Quality Monitoring While overall improvements to quality scores are important, often the weekto-week battle is improving fine-grained quality for important subsets of the input data. An individual subset may be rare but are nonetheless important, e.g., 0.1% of queries may correspond to a product feature that appears in an advertisement and so has an outsized importance. Traditional machine learning approaches effectively optimize
Modified on	04/02/2022 21:53
Name	PHOTONAI-A Python API for rapid machine learning model development
Number of Coding References	6
Number of Codes Coding	3
Coverage	6.57%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	31/01/2022 14:42
Coded Text	In the field of(deep) neural networks, libraries such as Tensorflow, Theano, Caffe and PyTorch [1–4] offer domain-specific implementations for nodes, layers, optimizers, as well as evaluation and utility functions. On top of that, higher level Application Programming Interfaces (APIs) such as Keras and fastai [5, 6] offer expressive syntaxes for accelerated and enhanced development ofdeep neural network architectures. In the same manner, the scikit-learn [7] toolbox, has evolved as one ofthe major resources ofthe field, covering a very broad range ofregression, classification, clustering, and preprocessing algorithms. It has established the de-facto standard interface for data processing and learning algorithms, and, in addition, offers a wide range of utility functions, such as cross-validation schemes and model evaluation metrics. Next to these general frameworks, other libraries in the software landscape offer functional- ities to address more specialized problems. Prominent examples are the imbalanced-learn toolbox [8], which provides numerous over- or under-sampling methods, or modality-specific libraries such as nilearn and nibabel [9, 10] which offer utility functions for accessing and preparing neuroimaging data. On top of that, the software landscape is complemented by several hyperparameter optimi- zation packages, each implementing a different strategy to find the most effective hyperparameter combination. Next to Bayesian approaches, such as Scikit-optimize or SMAC [11, 12], there are packages implementing evolutionary strategies [13] or packages approximating gradient descent within the hyperparameter space [14, 15]. Each package requires specific syntax and unique hyperparameter space definitions. Finally, there are approaches uniting all these components into algorithms that automati- cally derive the best model architecture and hyperparameter settings for a given dataset. Libraries such as auto-sklearn, TPOT, AutoWeka, Auto-keras, AutoML, Auto-Gluon and others optimize a specific set ofdata-processing methods, learning algorithms and their respective hyperparameters [16–22]. While very intriguing, these libraries aim at full automation—neglecting the need for customization and foregoing the opportunity to incorporate high-level domain knowledge in the model architecture search. Especially the complex and often highdimensional data structure native to medical and biological research requires the integration and application ofmodality-specific processing and often entails the development ofnovel algorithms.
Modified on	04/02/2022 14:35

Name	PHOTONAI-A Python API for rapid machine learning model development
Number of Coding References	6
Number of Codes Coding	3
Coverage	6.57%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Software Mantainability Aspects
Created on	04/02/2022 14:30
Coded Text	
Modified on	04/02/2022 14:36

Name	PHOTONAI-A Python API for rapid machine learning model development
Number of Coding References	6
Number of Codes Coding	3
Coverage	6.57%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Software Mantainability Aspects
Created on	04/02/2022 14:30
Coded Text	PHOTONAI is a high-level Python API designed to simplify and accelerate machine learning model development. It functions as a unifying framework allowing the user to easily access and combine algorithms from different toolboxes into custom algorithm sequences. It is especially designed to support the iterative model development process and automates the repetitive training, hyperparameter optimization and evaluation tasks. Importantly, the workflow ensures unbiased performance estimates while still allowing the user to fully customize the machine learning analysis. PHOTONAI extends existing solutions with a novel pipeline implementation supporting more complex data streams, feature combinations, and algorithm selection. Metrics and results can be conveniently visualized using the PHOTONAI Explorer and predictive models are shareable in a standardized format for further external validation or application.
Modified on	04/02/2022 14:30

Name	PHOTONAI-A Python API for rapid machine learning model development
Number of Coding References	6
Number of Codes Coding	3
Coverage	6.57%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Software Mantainability Aspects
Created on	04/02/2022 14:30
Coded Text	To address these issues, we propose PHOTONAI as a high-level Python API that acts as a mediator between different toolboxes. Established solutions are conveniently accessible or can be easily added. It combines an automated supervised machine learning workflow with the concept ofcustom machine learning pipelines. Thereby it is able to considerably accelerate design iterations and simplify the evaluation ofnovel analysis pipelines. In essence, PHOTONAI's major contributions are: Increased accessibility. By pre-registering data processing methods, learning algorithms, hyperparameter optimization strategies, performance metrics, and other functionalities, the user can effortlessly access established machine learning implementations via simple keywords. In addition, by relying on the established scikit-learn object API [23], users can easily integrate any third-party or custom algorithm implementation. Extended pipeline functionality. A simple to use class structure allows the user to arrange selected algorithms into single or parallel pipeline sequences. Extending the pipeline concept ofscikit-learn [7], we add novel functionality such as flexible positioning oflearning algorithms, target vector manipulations, callback functions, specialized caching, parallel datastreams, Or-Operations, and other features as described below. Automation. PHOTONAI can automatically train, (hyperparameter-) optimize and eval- uate any custom pipeline. Importantly, the user designs the training and testing procedure by selecting (nested) cross-validation schemes, hyperparameter optimization strategies, and performance metrics from a range ofpre-integrated or custom-built options. Thereby, development time is significantly decreased and conceptual errors such as information leakage between training, validation, and test set are avoided. Training information, baseline performances, hyperparameter optimization progress, and test performance evaluations are persisted and can be visualized via an interactive, browser-based graphical interface (PHOTONAI Explorer) to facilitate model insight. Model sharing. A standardized format for saving, loading, and distributing optimized and trained pipeline architectures enables model sharing and external model validation even for non-expert users.
Modified on	04/02/2022 14:37

Name	PHOTONAI-A Python API for rapid machine learning model development
Number of Coding References	6
Number of Codes Coding	3
Coverage	6.57%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:02
Coded Text	Currently, iterative model development approaches across different toolboxes as well as design and optimization ofcustom algorithm sequences are barely supported. For a start, scikit-learn has introduced the concept ofpipelines, which successively apply a list ofprocessing methods (referred to as transformers) and a final learning algorithm (called estimator) to the data. The pipeline directs the data from one algorithm to another and can be trained and evaluated in (simple) cross-validation schemes, thereby significantly reducing programmatic overhead. Scikit-learn's consistent usage ofstandard interfaces enables the pipeline to be subject to scikitlearn's inherent hyperparameter optimization strategies based on random- and grid-search. While being a simple and effective tool, several limitations still remain. For one, hyperparameter optimization requires a nested cross-validation scheme, which is not inherently enforced. Second, a standardized solution for easy integration ofcustom or third-party algorithms is not considered. In addition, several repetitive tasks, such as metric calculations, logging, and visualization lack automation and still need to be handled manually. Finally, the pipeline can not handle adjustments to the target vector, thereby excluding algorithms for e.g. data augmentation or handling class imbalance
Modified on	04/02/2022 14:36

Name	PHOTONAI-A Python API for rapid machine learning model development
Number of Coding References	6
Number of Codes Coding	3
Coverage	6.57%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:02
Coded Text	PHOTONAI is a high-level Python API designed to simplify and accelerate machine learning model development. It functions as a unifying framework allowing the user to easily access and combine algorithms from different toolboxes into custom algorithm sequences. It is especially designed to support the iterative model development process and automates the repetitive training, hyperparameter optimization and evaluation tasks. Importantly, the workflow ensures unbiased performance estimates while still allowing the user to fully customize the machine learning analysis. PHOTONAI extends existing solutions with a novel pipeline implementation supporting more complex data streams, feature combinations, and algorithm selection. Metrics and results can be conveniently visualized using the PHOTONAI Explorer and predictive models are shareable in a standardized format for further external validation or application.
Modified on	04/02/2022 14:30

Name	Practitioners’ insights on machine-learning software engineering design patterns~ a preliminary study
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Practitioners’ insights on machine-learning software engineering design patterns~ a preliminary study ~ IEEE Conference Publication ~ IEEE Xplore
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Pre-trained models~ Past, present and future
Number of Coding References	8
Number of Codes Coding	2
Coverage	7.00%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	Considering this issue, over the same period of developing deep neural networks, massive efforts have been devoted to manually constructing high-quality datasets for AI tasks (Deng et al., 2009; Lin et al., 2014; Bojar et al., 2014), making it possible to learn effective neural models for specific tasks that are superior to conventional non-neural models. However, it is expensive and time-consuming to manually annotate large-scale data. For example, utilizing crowdsourcing to segment images costs about $6.4 per image (Liu et al., 2020b). Some complex tasks that require expert annotations may charge much more to build their datasets. Several tasks such as visual recognition (Deng et al., 2009) and machine translation (Bojar et al., 2014) have datasets containing millions of samples, yet it is impossible to build such large-scale datasets for all AI tasks. More generally, the dataset of a specific AI task usually has a limited size. Hence, for a long time until now, it has been a key research issue: how to train effective deep neural models for specific tasks with limited human-annotated data. One milestone for this issue is the introduction of transfer learning (Thrun and Pratt, 1998; Pan and Yang, 2009). Instead of training a model from scratch with large amounts of data, human beings can learn to solve new problems with very few samples. This amazing learning process is motivated by the fact that human beings can use previously learned knowledge to handle new problems. Inspired by this, transfer learning formalizes a two-phase learning framework: a pre-training phase to capture knowledge from one or more source tasks, and a fine-tuning stage to transfer the captured knowledge to target tasks. Owing to the wealth of knowledge obtained in the pre-training phase, the fine-tuning phase can enable models to well handle target tasks with limited samples. Transfer learning provides a feasible method for alleviating the challenge of data hungry, and it has soon been widely applied to the field of computer vision (CV). A series of CNNs (Krizhevsky et al., 2012; Simonyan and Zisserman, 2015; Szegedy et al., 2015; He et al., 2016) are pre-trained on the human-annotated visual recognition dataset ImageNet (Deng et al., 2009). Benefiting from the strong visual knowledge distributed in ImageNet, fine-tuning these pre-trained CNNs with a small amount of task-specific data can perform well on downstream tasks. This triggers the first wave of exploring pre-trained models (PTMs) in the era of deep learning. In this wave, PTMs are used for almost all CV tasks such as image classification (He et al., 2016), object detection (Sermanet et al., 2014; Ren et al., 2016), image segmentation (Long et al., 2015), and image captioning (Oriol Vinyals et al., 2015). The natural language processing (NLP) community was also aware of the potential of PTMs and started to develop PTMs for NLP tasks (Qiu et al., 2020). To take full advantage of large-scale unlabeled corpora to provide versatile linguistic knowledge for NLP tasks, the NLP community adopts self-supervised learning (Liu et al., 2020b) to develop PTMs. The motivation of self-supervised learning is to leverage intrinsic correlations in the text as supervision signals instead of human supervision. For example, given the sentence “Beijing is the capital of China”, we mask the last word in the sentence, and then require models to predict the masked position with the word “China”. Through self-supervised learning, tremendous amounts of unlabeled textual data can be utilized to capture versatile linguistic knowledge without labor-intensive workload. This self-supervised setting in essence follows the well-known language model learning (Bengio et al., 2003). For a long time, the problem of vanishing or exploding gradients (Bengio et al., 1994) is the pain point of using deep neural networks for NLP tasks. Therefore, when the CV community advances the research of deep PTMs, the early exploration of the NLP community focuses on pre-training shallow networks to capture semantic meanings of words, 226 like Word2Vec (Mikolov et al., 2013a,b,c) and GloVe (Pennington et al., 2014). Although these pre-trained word embeddings play an important role in various NLP tasks, they still face a major limitation to represent polysemous words in different contexts, as each word is represented by only one dense vector. A famous example in NLP is that the word “bank” has entirely different meanings in the sentences “open a bank account” and “on a bank of the river”. This motivates pre-training RNNs to provide contextualized word embeddings (Melamud et al., 2016; Peters et al., 2018; Howard and Ruder, 2018), yet the performance of these models is still limited by their model size and depth. With the development of deep neural networks in the NLP commu-

	nity, the introduction of Transformers (Vaswani et al., 2017) makes it feasible to train very deep neural models for NLP tasks. With Transformers as architectures and language model learning as objectives, deep PTMs GPT (Radford and Narasimhan, 2018) and BERT (Devlin et al., 2019) are proposed for NLP tasks in 2018. From GPT and BERT, we can find that when the size of PTMs becomes larger, large-scale PTMs with hundreds of millions of parameters can capture polysemous disambiguation, lexical and syntactic structures, as well as factual knowledge from the text. By fine-tuning large-scale PTMs with quite a few samples, rich linguistic knowledge of PTMs brings awesome performance on downstream NLP tasks. As shown in Fig. 1(a) and Fig. 1(b), large-scale PTMs well perform on both language understanding and language generation tasks in the past several years and even achieve better results than human performance. As shown in Fig. 2(a), all these efforts and achievements in the NLP community let large-scale PTMs become the focus of AI research, after the last wave that PTMs allow for huge advances in the CV community. Up to now, various efforts have been devoted to exploring large-scale PTMs, either for NLP (Radford et al., 2019; Liu et al., 2020d; Raffel et al., 2020; Lewis et al., 2020a), or for CV (Lu et al., 2019; Li et al., 2019; Tan and Bansal, 2019). Fine-tuning large-scale PTMs for specific AI tasks instead of learning models from scratch has also become a consensus (Qiu et al., 2020). As shown in Fig. 2(b), with the increasing computational power boosted by the wide use of distributed computing devices and strategies, we can further advance the parameter scale of PTMs from million-level to billion-level (Brown et al., 2020; Lepikhin et al., 2021; Zeng et al., 2021; Zhang et al., 2020c, 2021a) and even trillion-level (Fedus et al., 2021). And the emergence of GPT-3 (Brown et al., 2020), which has hundreds of billions of parameters, enables us to take a glimpse of the latent power distributed in massive model parameters, especially the great abilities of few-shot learning like human beings (shown in Fig. 3). The existing large-scale PTMs have improved the model performance on various AI tasks and even subverted our current perception of the performance of deep learning models. However, several fundamental issues about PTMs still remain: it is still not clear for us the nature hidden in huge amounts of model parameters, and huge computational cost of training these behemoths also prevents us from further exploration. At this moment, these PTMs have pushed our AI researchers to a crossroad, with a number of open directions to go.
Modified on	28/02/2023 12:36

Name	Pre-trained models~ Past, present and future
Number of Coding References	8
Number of Codes Coding	2
Coverage	7.00%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	6.3. Model compression Another important approach to improve the efficiency of PTMs is model compression. In this setting, large models are compressed to small ones to meet the demand for faster inference and deployment on resource-constrained devices. Parameter Sharing. PTMs can be compressed with sharing param- eters across similar units. ALBERT (Lan et al., 2019) uses factorized embedding parameterization and cross-layer parameter sharing to reduce the parameters of PTMs. Using same weights across all Transformer layers, ALBERT achieves a significant parameter reduction based on the BERT model, and meanwhile has the same or even better performance. This indicates over-parameterized. Model Pruning. To take more advantage of the over-parameterized feature of current PTMs, another method to reduce model parameters is model pruning, which cuts off some useless parts in PTMs to achieve accelerating while maintaining the performance. In (Fan et al., 2019), Transformer layers are selectively dropped during training, resulting in a more shallow model during inference. In (Michel et al., 2019; Voita et al., 2019), and (Zhang et al., 2021b), researchers study the redundancy of the attention heads in Transformers and find that only a small part of them is enough for good performance. Most of these heads can be that PTMs can be extremely X. Han et al. AI Open 2 (2021) 225–250 removed with little impact on the accuracy. Other trials such as CompressingBERT (Gordon et al., 2020) try to prune the weights of attention layers and linear layers to reduce the number of parameters in PTMs, while maintaining the comparable performance to the original model. Knowledge Distillation. Although ALBERT saves the memory usage of PTMs, its inference time is not significantly decreased since features still need to go through its layers with the same number as the original model. Knowledge distillation aims at training a small model to reproduce the behavior of a large teacher model. The memory usage and the time overhead are both decreased when using a small distilled model for inference. There are some typical works employing knowledge distillation for PTMs, such as DistillBERT (Sanh et al., 2019), TinyBERT (Jiao et al., 2019), BERT-PKD (Sun et al., 2019b) and MiniLM (Wang et al., 2020d). In these works, a small student model is trained to mimic the output probability, the hidden states, and the attention matrices of a large teacher model during both the pre-training and fine-tuning stages. With knowledge distillation, the modelegy in the teacher model is transferred into the student model, which can lead to increasing performance compared to training a student model alone. However, the knowledge distillation methods mentioned above require the data used for pre-training the teacher model, which is usually not released in consideration of the data copyright and privacy. Moreover, the teacher model needs to forward over the entire pre-training data to produce logits or intermediate representations for knowledge distillation, causing an even longer training time. Model Quantization. To get a more compressed model, model quantization is also a useful technique, which has been widely explored in some CNN-based models (Stock et al., 2020; Polino et al., 2018). Model quantization refers to the compression of higher-precision floating-point parameters to lower-precision floating-point ones. Conventional PTMs are usually represented in 32 bits or 16 bits, while models after quantization can be in 8 bits or even 1 or 2 bits. For recent Transformer-based models, 8-bit quantization has been proved to be effective for model compression in Q8BERT (Zafrir et al., 2019), with little impact on the model performance. Despite this, training 1 or 2 Bits models remains challenging due to the significant decrease in model capacity. To alleviate the performance degradation, other methods to preserve the accuracy can also be employed. Q-BERT (Shen et al., 2020a) uses mixed-bits quantization in which the parameters with higher Hessian spectrum require higher precision while those parameters with lower Hessian spectrum need lower precision. TernaryBERT (Zhang et al., 2020b) applies knowledge distillation in quantization, forcing low-bit models to imitate full-precision models. Both Q-BERT and TernaryBERT result in ultra low-bit models. However, low-bit representation is a highly hardware-related technique, which means quantization often requires specific hardware and can not generalize to other devices.

Modified on	28/02/2023 12:41
Name	Pre-trained models~ Past, present and future
Number of Coding References	8
Number of Codes Coding	2
Coverage	7.00%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	be regarded as a branch of unsupervised learning because they both apply unlabeled data. However, unsupervised learning mainly focuses on detecting data patterns (e.g., clustering, community discovery, and anomaly detection), while self-supervised learning is still in the paradigm of supervised settings (e.g., classification and generation) (Liu et al., 2020b). The development of self-supervised learning makes it possible to perform pre-training on large-scale unsupervised data. Compared to supervised pre-training working as the cornerstone of CV in the deep learning era, self-supervised pre-training allows for huge advances in the field of NLP. Although some supervised pre-training methods like CoVE have achieved promising results on NLP tasks, it is nearly impossible to annotate a textual dataset as large as ImageNet, considering annotating textual data is far more complex than annotating images. Hence, applying self-supervised learning to utilize unlabeled data becomes the best choice to pre-train models for NLP tasks. The recent stunning breakthroughs in PTMs are mainly towards NLP tasks, more specifically pre-trained language models.
Modified on	28/02/2023 12:38

Name	Pre-trained models~ Past, present and future
Number of Coding References	8
Number of Codes Coding	2
Coverage	7.00%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Improving computational efficiency As introduced in Section 1, a major trend of PTMs is that the number of parameters is getting larger and larger. Increasing the size of a neural network typically improves accuracy, but it also increases the memory and computational requirements for training the model. In this section, we will introduce how to improve computational efficiency from the following three aspects: system-level optimization, efficient learning algorithms, and model compression strategies. 6.1. System-level optimization An effective and practical way to reduce computational requirements is system-level optimization towards computational efficiency and memory usage. System-level optimization methods are often modelagnostic and do not change underlying learning algorithms. Therefore, they are widely used in training large-scale PTMs. Generally, these methods can be divided into single-device optimization methods and multi-device optimization ones. Single-Device Optimization. Current large-scale PTMs usually cost 237 a lot of memory for pre-training. This is mainly due to the redundant representation of floating-point numbers. Modern deep learning systems are mainly based on a single-precision floating-point format (FP32). However, the weights of models usually fall in a limited range, and using a half-precision floating-point format (FP16) can accomplish most of the computation with little precision loss (Gupta et al., 2015). However, in some cases, training models in FP16 may fail because of the floating-point truncation and overflow. To tackle this problem, mixed-precision training methods (Micikevicius et al., 2018) have been proposed, which preserve some critical weights in FP32 to avoid the floating-point overflow and use dynamic loss scaling operations to get rid of the floating-point truncation. Sufficient experiments have shown that mixed-precision training methods are more stable than directly training models in FP16. Although mixed-precision training methods can significantly reduce the training time and memory usage, they still face some challenges. When model parameters are not initialized well, mixed-precision methods may still cause unstable training. All these challenges still require to be further explored. Besides the redundant representation of floating-point numbers, the activation states saved for computing gradients are also redundant. For example, in Transformer-based models, apart from the weights of attention layers and linear layers, computational devices also store the hidden states of each layer for the efficiency of the chain rule used in the gradient back-propagation. As compared with model parameters, these hidden states can consume even much more memory. To handle redundant activation states, gradient check pointing methods (Rasley et al., 2020) have been used to save memory by storing only a part of the activation states after forward pass. The discarded activation states are recomputed during the backward steps if necessary. When pre-training recent large-scale PTMs, the memory consump- tion can be too large to fit in a single GPU. Therefore, some works (Huang et al., 2020a) attempt to store model parameters and activation states with the CPU memory rather than the GPU memory, since the CPU memory is usually much larger. As shown in Fig. 10, some works such as ZeRO-Offload (Ren et al., 2021) design delicate strategies to schedule the swap between the CPU memory and the GPU memory so that memory swap and device computation can be overlapped as much as possible. Multi-Device Optimization. Recently, distributed training is commonly used in pre-training, where multiple GPUs distributed in many computational nodes are used together to train a single model. Data parallelism (Li et al., 2020d) is a simple and effective approach to accelerate training a model. As shown in Fig. 11, when we use data parallelism, a large batch is partitioned to different nodes and thus forward pass can be parallelized. At backward pass, the gradients on different nodes should be aggregated with all-reduce operations to ensure the consistency of parameter optimization, which may introduce additional communication overhead. When pre-training models with billions to trillions of parameters, traditional data parallelism brings challenges of fitting whole model parameters into a single GPU, even with half-precision or mixedprecision training. Although this problem can be solved by using a GPU with larger memory, the expenses can be hard to afford, limiting the use of PTM by ordinary

	researchers. Model parallelism is an effective way to tackle this problem (Shazeer et al., 2018). As shown in Fig. 11, when conducting model parallelism, model parameters can be distributed to multiple nodes. The communication operations between these nodes like reduce-scatter and all-gather guarantee the correctness of forward pass and backward pass. Megatron-LM (Shoeybi et al., 2019) adopts model parallelism to Transformer-based PTMs. It splits self-attention heads as well as feed-forward layers into different GPUs, reducing the memory burden of a single GPU. Mesh-Tensorflow (Shazeer et al., 2018) also enables users to split tensors along any tensor dimensions, which can bring more customized options for model parallelism. Although model parallelism enables different computational nodes
Modified on	28/02/2023 12:39
Name	Pre-trained models~ Past, present and future
Number of Coding References	8
Number of Codes Coding	2
Coverage	7.00%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Large-scale pre-trained models (PTMs) such as BERT and GPT have recently achieved great success and become a milestone in the field of artificial intelligence (AI). Owing to sophisticated pre-training objectives and huge model parameters, large-scale PTMs can effectively capture knowledge from massive labeled and unlabeled data. By storing knowledge into huge parameters and fine-tuning on specific tasks, the rich knowledge implicitly encoded in huge parameters can benefit a variety of downstream tasks, which has been extensively demonstrated via experimental verification and empirical analysis. It is now the consensus of the AI community to adopt PTMs as backbone for downstream tasks rather than learning models from scratch. In this paper, we take a deep look into the history of pre-training, especially its special relation with transfer learning and self-supervised learning, to reveal the crucial position of PTMs in the AI development spectrum. Further, we comprehensively review the latest breakthroughs of PTMs. These breakthroughs are driven by the surge of computational power and the increasing availability of data, towards four important directions: designing effective architectures, utilizing rich contexts, improving computational efficiency, and conducting interpretation and theoretical analysis. Finally, we discuss a series of open problems and research directions of PTMs, and hope our view can inspire and advance the future study of PTMs.
Modified on	28/02/2023 12:35

Name	Pre-trained models~ Past, present and future
Number of Coding References	8
Number of Codes Coding	2
Coverage	7.00%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Self-supervised learning and self-supervised pre-training As shown in Fig. 4, transfer learning can be categorized under four sub-settings, inductive transfer learning (Lawrence and Platt, 2004; Mihalkova et al., 2007; Evgeniou and Pontil, 2004), transductive transfer learning (Shimodaira, 2000; Zadrozny, 2004; Daume III and Marcu, 2006), self-taught learning (Raina et al., 2007; Dai et al., 2008),4 and unsupervised transfer learning (Wang et al., 2008). Among these four settings, the inductive and transductive settings are the core of research, as these two settings aim to transfer knowledge from supervised source tasks to target tasks. Although supervised learning is always one of the core issues of machine learning research, the scale of unlabeled data is much larger than that of manually labeled data. Recently, more and more researchers have noticed the importance of large-scale unlabeled data and are committed to extracting information from unlabeled data. Self-supervised learning has been proposed to extract knowledge from large-scale unlabeled data by leveraging input data itself as supervision. Self-supervised learning and unsupervised learning have many similarities in their settings. To a certain extent, self-supervised learning can
Modified on	28/02/2023 12:37

Name	Pre-trained models~ Past, present and future
Number of Coding References	8
Number of Codes Coding	2
Coverage	7.00%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	that we can pre-train large-scale PTMs with a lower cost solution. Efficient Training Methods. Conventional pre-training tasks can be sample-inefficient. For example, for MLM which is widely used to pretrain recent PTMs, models are required to predict masked tokens according to contexts. The masked tokens are usually a subset (typically 15%) of input tokens, i.e., models can only learn from a small set of input tokens. To tackle this problem, ELECTRA (Clark et al., 2020) applies the replaced token detection task. This task forces models to distinguish whether an input token is replaced by a generator. This task can leverage more supervision information from each sample since all input tokens need to be distinguished. ELECTRA takes much fewer pre-training steps when it reaches similar performance to those MLM models. Furthermore, traditional MLM randomly masks tokens in a document to predict. Since the difficulty of predicting different tokens varies a lot, the random masking strategy makes the training process aimless and inefficient. Therefore, some works selectively mask tokens based on their importance (Gu et al., 2020) or gradients (Chen et al., 2020b) in back-propagation to speed up model training. Apart from the pre-training tasks, the current pre-training dynamics are also sub-optimal. Recent large-scale PTMs usually require a large batch size. But in an early work (Goyal et al., 2017), researchers find that naively increasing the batch size may cause difficulty in optimization. Therefore, they propose a warmup strategy that linearly increases the learning rate at the beginning of training. This strategy is commonly used in recent large-scale PTMs. Another feature of recent PTMs is that they are usually composed of multiple stacks of a base structure like Transformers. The conventional training paradigm optimizes each layer simultaneously using the same hyper-parameters. However, some recent works study Transformer-based models and claim that different layers can share similar self-attention patterns. Therefore, a shallow model can firstly be trained and then duplicated to construct a deep model (Gong et al., 2019). Some layers can also be dropped during training to reduce the complexity of back-propagation and weight update (Zhang and He, 2020). In addition, You et al. (2017) and You et al. (2020) find that adaptively using different learning rates at different layers can also speed up convergence when the batch size is large. Efficient Model Architectures. Besides efficient pre-training methods, more variants of model architectures can also reduce the computational complexity to improve the efficiency of training PTMs. For most Transformer-based PTMs, as their input sequence goes longer, their efficiency is limited by the computation of attention weights due to its quadratic time and space complexity of the sequence length. Therefore, many works attempt to reduce the complexity of Transformers. Some works (Peng et al., 2021; Choromanski et al., 2021; Wang et al., 2020c; Katharopoulos et al., 2020) design low-rank kernels to theoretically approximate the original attention weights and result in linear complexity. Some works (Child et al., 2019) introduce sparsity into attention mechanisms by limiting the view of each token to a fixed size
Modified on	28/02/2023 12:40

Name	Pre-trained models~ Past, present and future
Number of Coding References	8
Number of Codes Coding	2
Coverage	7.00%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	to store different parts of model parameters, it has to insert collective communication primitives during both forward pass and backward pass, which can not be overlapped by device computation. On the contrary, the all-reduce collective communication operation in data parallelism usually can be overlapped by the backward computation. As a result, data parallelism is preferred as long as it can conquer the excessive requirement of memory capacity. In the standard implementation of data parallelism, optimizer states are usually copied along different nodes to guarantee synchronized optimization across data parallelism units. This redundancy leads to the additional overhead of GPU memory, especially when models are trained in a mixed-precision manner because the optimizer needs to store 32-bit master states of these models to ensure accuracy. To eliminate the redundancy brought by optimizer states and parameters, ZeRO optimizer (Rajbhandari et al., 2020) methods equally partition and distribute optimizer states to each node of data parallelism, such that each node only updates the optimizer states corresponding to its partition. At the end of a training step, all optimizer states are gathered across data parallelism nodes. The above-mentioned model parallelism techniques mainly focus on partitioning and parallelizing matrix operations across different nodes. 238 As shown in Fig. 12, another effective method for model parallelism is pipeline parallelism, which partitions a deep neural network into multiple layers and then puts different layers onto different nodes. After the computation of each node, the output is sent to the next node where the next layer computation takes place. Since pipeline parallelism only needs to communicate the intermediate activation states between nodes performing adjacent stages of the pipeline, the communication cost is relatively small. Existing pipeline methods include GPipe (Huang et al., 2019b) which can send smaller parts of samples within a mini-batch to different nodes, and Tera Pipe (Li et al., 2021) which can apply token-level pipeline mechanisms for Transformer-based models to make each token in a sequence be processed by different nodes. Both of these pipeline methods speed up the large-scale PTMs. However, they should be stopped at the end of each batch until the gradient back-propagation is complete, which can lead to pipeline bubbles. 6.2. Efficient pre-training Besides some system-level optimization methods, various efforts have been devoted to exploring more efficient pre-training methods, so
Modified on	28/02/2023 12:40

Name	Pre-trained models~ Past, present and future Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Quality Assurance for AI-Based Systems~ Overview and Challenges (Introduction to Interactive Session)
Number of Coding References	9
Number of Codes Coding	2
Coverage	9.96%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	
Modified on	04/02/2022 14:00

Name	Quality Assurance for AI-Based Systems~ Overview and Challenges (Introduction to Interactive Session)
Number of Coding References	9
Number of Codes Coding	2
Coverage	9.96%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	For instance, additional quality properties of AI components and AI-based systems have to be taken into account. Zhang et al. [5] consider the following quality properties: – Correctness refers to the probability that an AI component gets things right. – Model relevance measures how well an AI component fits the data. – Robustness refers to the resilience of an AI component towards perturbations. – Security measures the resilience against potential harm, danger or loss made via manipulating or illegally accessing AI components. – Data privacy refers to the ability of an AI component to preserve private data information.
Modified on	04/02/2022 14:01

Name	Quality Assurance for AI-Based Systems~ Overview and Challenges (Introduction to Interactive Session)
Number of Coding References	9
Number of Codes Coding	2
Coverage	9.96%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	we defined the three dimensions artifact type (i.e., data, model, framework, and system), process (from isolated to continuous), and quality characteristics (with respect to software quality, quality-in-use, and data quality). Furthermore, we elaborated on the key challenges of (1) understandability and interpretability of AI models, (2) lack of specifications and defined requirements, (3) need for validation data and test input generation, (4) defining expected outcomes as test oracles, (5) accuracy and correctness measures, (6) non-functional properties of AI-based systems, (7) self-adaptive and self-learning characteristics, and (8) dynamic and frequently changing environments. In order to properly address the challenges raised in this paper and to enable high quality AI-based systems, first and foremost, exchange of knowledge and ideas between the SE and the AI community is needed. One channel of exchange is education or training through dedicated courses [29]ormedia[30]. Another one are dedicated venues for exchange and discussion of challenges on quality assurance for AI-based systems
Modified on	04/02/2022 14:02

Name	Quality Assurance for AI-Based Systems~ Overview and Challenges (Introduction to Interactive Session)
Number of Coding References	9
Number of Codes Coding	2
Coverage	9.96%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	
Modified on	04/02/2022 14:00

Name	Quality Assurance for AI-Based Systems~ Overview and Challenges (Introduction to Interactive Session)
Number of Coding References	9
Number of Codes Coding	2
Coverage	9.96%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	Efficiency measures the construction or prediction speed of an AI component. – Fairness ensures that decisions made by AI components are in the right way and for the right reason to avoid problems in human rights, discrimination, law, and other ethical issues. – Interpretability refers to the degree to which an observer can understand the cause of a decision made by an AI component.
Modified on	04/02/2022 14:01

Name	Quality Assurance for AI-Based Systems~ Overview and Challenges (Introduction to Interactive Session)
Number of Coding References	9
Number of Codes Coding	2
Coverage	9.96%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	For instance, additional quality properties of AI components and AI-based systems have to be taken into account. Zhang et al. [5] consider the following quality properties: – Correctness refers to the probability that an AI component gets things right. – Model relevance measures how well an AI component fits the data. – Robustness refers to the resilience of an AI component towards perturbations. – Security measures the resilience against potential harm, danger or loss made via manipulating or illegally accessing AI components. – Data privacy refers to the ability of an AI component to preserve private data information.
Modified on	04/02/2022 14:01

Name	Quality Assurance for AI-Based Systems~ Overview and Challenges (Introduction to Interactive Session)
Number of Coding References	9
Number of Codes Coding	2
Coverage	9.96%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	In addition to outlining important concepts and terms in the previous section, this section elaborates on the following key challenges encountered in the development of approaches for quality assurance and testing of AI-based systems. – Understandability and interpretability of AI models – Lack of specifications and defined requirements – Need for validation data and test input generation – Defining expected outcomes as test oracles – Accuracy and correctness measures – Non-functional properties of AI-based systems – Self-adaptive and self-learning characteristics – Dynamic and frequently changing environments.
Modified on	04/02/2022 14:01

Name	Quality Assurance for AI-Based Systems~ Overview and Challenges (Introduction to Interactive Session)
Number of Coding References	9
Number of Codes Coding	2
Coverage	9.96%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	The knowledge and background of different communities are brought together for developing AI-based systems. While this leads to new and innovative approaches, exciting breakthroughs, as well as a significant advancement in what can be achieved with modern AI-based systems, it also fuels the babel of terms, concepts, perceptions, and underlying assumptions and principles. For instance, the term “regression” in ML refers to regression models or regression analysis, whereas in SE it refers to regression testing. Speaking about “testing”, this term is defined as the activity of executing the system to reveal defects in SE but refers to the evaluation of performance characteristics (e.g., accuracy) of a trained model with a holdout validation dataset in ML. The consequences are increasing confusion and potentially conflicting solutions for how to approach quality assurance for AI-based systems and how to tackle the associated challenges. While this paper starts from a software engineering point of view, its goal is to incorporate and discuss also many other perspectives, which eventually aggregate into a multi-dimensional big picture of quality assurance for AI-based systems.
Modified on	04/02/2022 14:00

Name	Quality Assurance for AI-Based Systems~ Overview and Challenges (Introduction to Interactive Session)
Number of Coding References	9
Number of Codes Coding	2
Coverage	9.96%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	With the pervasive use and the dependence on AIbased systems, the quality of these systems becomes essential for their practical usage. However, quality assurance for AI-based systems is an emerging area that has not been well explored and requires collaboration between the SE and AI research communities. This paper discusses terminology and challenges on quality assurance for AI-based systems to set a baseline for that purpose. Therefore, we define basic concepts and characterize AI-based systems along the three dimensions of artifact type, process, and quality characteristics. Furthermore, we elaborate on the key challenges of (1) understandability and interpretability of AI models, (2) lack of specifications and defined requirements, (3) need for validation data and test input generation, (4) defining expected outcomes as test oracles, (5) accuracy and correctness measures, (6) non-functional properties of AI-based systems, (7) self-adaptive and self-learning characteristics, and (8) dynamic and frequently changing environments.
Modified on	04/02/2022 13:58

Name	Quality Management of Machine Learning Systems
Number of Coding References	8
Number of Codes Coding	2
Coverage	21.34%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	
Modified on	04/02/2022 13:52

Name	Quality Management of Machine Learning Systems
Number of Coding References	8
Number of Codes Coding	2
Coverage	21.34%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	3.1 Where are the bugs?
Modified on	04/02/2022 13:56

Name	Quality Management of Machine Learning Systems
Number of Coding References	8
Number of Codes Coding	2
Coverage	21.34%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	3.2 Quality Improvement Tasks for ML systems This section describes the suggested tasks to find defects in the artifacts described in Section 3.1 and resolve them. These are traditional activities modified to reflect the inclusion of the ML component in the application. Due to space limitations, reference to any specific technique or tool is meant to provide an example, rather than an exhaustive list. Quality improvement tasks that address the unique aspects of assessing 'Trust' in ML systems are described in Section 3.3.
Modified on	04/02/2022 13:56

Name	Quality Management of Machine Learning Systems
Number of Coding References	8
Number of Codes Coding	2
Coverage	21.34%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	Manual Inspection With the support of tools [17] manual inspection is still an effective way to find defects in requirements, architecture, algorithms and code. Techniques, such as pair programming [6] have proven very useful in practice. Static Analysis Application of static analysis to find defects in software programs is a very mature field, dating back to many decades [18]. There have been recent examples of applying this technique to machine learning code [19]. Many development environments support basic syntax checking, when the code is being written. White Box Testing Traditional white box testing [20] leverages the knowledge of the program structure to execute the program in ways to achieve the desired coverage e.g. branch coverage, statement coverage, etc. to locate defects. 8 P. Santhanam Similarly, a data scientist can use the detailed knowledge of a neural network behavior in the model building process to apply various coverage criteria to the network to find the defects in the model. This has led to concepts such as neuron coverage [21], novel test criteria that are tailored to structural features of DNNs and their semantics [22], the condition-decision relationships between adjacent layers and the combinations of values of neurons in the same layer [23], mutation techniques on source code, data and models [24] and combinatorial test design consisting of neuron pairs in the layers and the neuron-activation configurations[25]. These techniques demonstrate various ways to expose incorrect behavior of the network while being mindful of the computational cost of test generation itself. Black Box Testing Traditional black box testing (or functional testing) [20] focuses on detecting defects in the expected external behavior of the software component by carefully manipulating the input space. For ML models, it is important that test data represents the business requirements in terms of data values and distributions and was not used during the model creation process. Key goal of black box testing is to evaluate if the model generalizes adequately for previously unseen data or suggest a model rework, if not suitable. These considerations also apply for system integration tests. Data Assessment & Testing There are several techniques and tools to check the quality of the modeling data during development. Breck et al.[26] present a highly scalable data validation system, designed to detect data anomalies (e.g. unexpected patterns, schema-free data, etc.) in the machine learning pipelines. Barash et al.[27] use combinatorial design methodology to define the space of business requirements and map it to the ML solution data, and use the notion of data slices to identify uncovered requirements, under-performing slices, or suggest the need for additional training data. This is also an example of using data slicing for black box testing. Application Monitoring Application monitoring during operations is a critical activity in ML applications since the model performance can change over time due to previously unseen pattern in the operational data or emergent behavior not expected in the model building process. Breck et al [26] also describe techniques to detect feature skew by doing a key-join between corresponding batches of training and operational data followed by a feature wise comparison. Distribution skew between training data and serving data is detected by distance measures. Raz et al.[28] discuss a novel approach, solely based on a classifier suggested labels and its confidence in them, for alerting on data distribution or feature space changes that are likely to cause data drift. This has two distinct benefits viz. no model input data is required and does not require labeling of data in production. In addition to the detecting any degradation of model performance, there need to be processes in place to correct the behavior Quality Management of Machine Learning Systems 9 as and when it occurs. There are examples of commercial offerings to perform this task [29]. Debugging Debugging is often the most under-appreciated activity in software development that takes considerable skill and effort in reality. As noted in [7], debugging typically happens during three different stages in software life cycle, and the level of granularity of the analysis required for locating the defect differs in these three. First stage is during the model building process by the data scientist who has access to the details of the model. Here, there are two classes of errors that need debugging. (a) raw errors resulting in the execution of the model code in the development environment during the process

of model creation. The development frameworks can provide support for debugging this class of problems. (b) Model executes successfully, but the overall performance of the model output is not adequate for the chosen task or if the model output does not meet the expectation for specific input instances, it is necessary to find the reason for these behaviors. There could be many causes, including bad choice of the algorithm, inadequate tuning of the parameters, quality and quantity of the modeling data, etc. [30, 31]. Some care is also needed in providing model debugging information to the user to avoid exposure of system details, susceptible for adversarial attacks. The second stage for debugging is during the later black box testing activities

in development when an unexpected behavior is encountered. A certain amount of debugging of the test execution is necessary to conclude that the AI model is the cause of the unexpected behavior. Once that is confirmed, debugging of the model follows the same process as described above. Third stage is during operations, when the application is being put to real use. Any unexpected behavior here can be the result of changes in the computing environment relative to development or due to new patterns in the operational data not previously seen the modeling data. Techniques discussed in [26, 28,29] can be used to address the model drift problems.

3.3 AI Trust Assessment

Due to the black box nature of the ML models and their behavior being decided by modeling data, trust in model outputs has become an important consideration in business applications. This section deals with four specific aspects trust.

Explainability In many business critical applications, the outputs of the black box ML models also require explanations to meet the business objectives. There are many motivations for explanations [32] and it is important to know the need so that the appropriate approach can be used. There are examples of open source packages for implementing explainability [33] in business applications.

Bias/Fairness Due to the potential sensitivity of the outputs of the ML models to biases inherent in the modeling data, there is a critical question of the fairness

10 P. Santhanam

of the algorithms [34] in extracting the model from the data. There are examples of open source packages for understanding and mitigating biases [35] in business applications.

Robustness The owner of a ML application needs a strategy for defending against adversarial attacks. Xu et al.[36] provide a comprehensive summary of the adversarial attacks against ML models built using images, graphs and text and the countermeasures available. Reference [37] describes an open-source software library, designed to help researchers and developers in creating novel defense techniques, and in deploying practical defenses of real-world AI systems.

Transparency Given the abundance of AI components (i.e. algorithms, services, libraries, frameworks) available from open source and commercial offerings, it makes sense for a company to reuse the available software component in its application. However, due to the concerns about the trust in the available component, the consumer of the component needs some detailed information about the component to manage the risk. This need for transparency requires additional assessment. Key pieces of such information are captured in a FactSheet [38], which provides the technical and process background of the AI asset to the consumer.

3.4 Quality Metrics

Due to the unique attributes of AI based systems (discussed in Section 2), there is a critical need to reevaluate the metrics for their quality management. This section discusses three aspects that highlight the need.

Defect management The lack of a clear definition of software defect in ML applications discussed in section 2.2 is a major problem in quality management. While the defects in the other artifacts can be captured unambiguously, the perceived errors in the model outputs are subject to the statistical uncertainties. As a result, until a detailed debugging is performed to diagnose the reason for the unexpected behavior one cannot be certain that this is a bug. Hence the defect management tools have to allow this possibility with potentially more time assigned for the necessary investigation that may point to an inadequate training data set.

Model Evaluation Model evaluation is an important part of the ML application development. There are many metrics [39] that can be used and depending on the specific application domain. They need to be chosen and used carefully. In addition to these usual machine learning metrics, additional metrics specific to the trust topics discussed in section 3.3 (explainability, bias, robustness and transparency) are also necessary to support the business objectives and manage technical & business risk. There is also recent work [40] to measure applicationlevel key performance indicators to provide feedback to the AI model life cycle.

Quality Management of Machine Learning Systems 11

Model Uncertainty In addition to the usual ML metrics [39], typically at the end of a DNN pipeline is a softmax activation (usually a sigmoid function) that estimates a confidence level for each output, expressed as a probability measure between 0 and 1. In reality, a high confidence level does not necessarily mean low uncertainty [41] and hence it is not reliable for decision support. This is because DNN models do not have a way of calculating uncertainty by themselves. Gal and Ghahramani [41] have proposed a new method to estimate uncertainty in DNN outputs that approximates Bayesian models, while not

Modified on	requiring a high computing cost. Lakshminarayanan et al.[42] have demonstrated an alternate approach that is scalable and can be easily implemented. Any mission critical ML system has to include such uncertainty measures. 24/01/2023 11:56
Name	Quality Management of Machine Learning Systems
Number of Coding References	8
Number of Codes Coding	2
Coverage	21.34%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	The goal of this paper is to provide an overview of such a framework built upon tools and methodology available today and identify gaps for new software engineering research. The focus of this paper is on AI systems implemented using machine learning. A popular ML technique is the use of Deep Neural Networks (DNN). This paper uses AI and ML interchangeably.
Modified on	04/02/2022 13:51
Name	Quality Management of Machine Learning Systems
Number of Coding References	8
Number of Codes Coding	2
Coverage	21.34%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	
Modified on	04/02/2022 13:52

Name	Quality Management of Machine Learning Systems
Number of Coding References	8
Number of Codes Coding	2
Coverage	21.34%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	In spite of an explosive growth in the raw AI technology and in consumer facing applications on the internet, its adoption in business applications has conspicuously lagged behind. For business/missioncritical systems, serious concerns about reliability and maintainability of AI applications remain. Due to the statistical nature of the output, software 'defects' are not well defined. Consequently, many traditional quality management techniques such as program debugging, static code analysis, functional testing, etc. have to be reevaluated. Beyond the correctness of an AI model, many other new quality attributes, such as fairness, robustness, explainability, transparency, etc. become important in delivering an AI system.
Modified on	04/02/2022 13:45

Name	Quality Management of Machine Learning Systems
Number of Coding References	8
Number of Codes Coding	2
Coverage	21.34%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	This supports the assertion that moving AI from a proof-ofconcept to real business solution is not a trivial exercise. Some common reasons cited for this result are: – Insufficient alignment of business goals and processes to the AI technology (akin to the challenges of introducing information technology in the 1990's). – Lack of data strategy (i.e. "There is no AI without IA (Information Architecture)") – Shortage of skilled people who can combine domain knowledge and the relevant AI technology. – Unique concerns about AI (e.g. model transparency, explainability, fairness/bias, reliability, safety, maintenance, etc.) – Need for better engineering infrastructure for data and model provenance. As the application of AI moves to business/mission critical tasks with more severe consequences, the need for a rigorous quality management framework becomes critical. It is bound to be very different from the practices and processes that have been in place for IT projects over many decades.
Modified on	04/02/2022 13:49

Name	Recent Challenges on Edge AI with Its Application~ A Brief Introduction
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Recent Challenges on Edge AI with Its Application~ A Brief Introduction Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Regulating Accuracy-Efficiency Trade-Offs in Distributed Machine Learning Systems
Number of Coding References	7
Number of Codes Coding	1
Coverage	16.93%
Folder Location	Codes\\Tradeoff between Maintainability and Scalability
Parent Name	Codes\\Tradeoff between Maintainability and Scalability\\effect and influence between Maintainbility and scalability
Created on	01/06/2022 13:41
Coded Text	3.3 Resource-Constrained Machine Learning The prior examples discuss the cost ofrunning computations. Specifically, they discuss how differently-sized batches of data (Section 3.1) and how differing degrees ofnumerical precision (Section 3.2) directly impact how long it takes a computer to execute a computation. Even though these examples concern a computer's behavior, we have not yet considered how hardware specifications of the computer running the algorithm might also impact that behavior. Surely this is important, as different computers have different computing capabilities due to varying hardware; a NASA supercomputer has more computational resources than a personal laptop. Recent years have seen an increase in the variety of computational devices available and a corresponding increase in the variety of computations we wish to run on them. For example, Internet of Things (IoT) devices and sensors, such as Google Home or Amazon Echo, perform inference. They serve up answers to spoken language questions; however, they also have limited on-board capabilities to perform computations locally. These limitations take several forms. For example, such devices might not have a lot of power to process data quickly or might lack storage capacity for large amounts of data. Often, these devices can communicate with more sophis- ticated computers over the Internet, offloading computation or storage to those computers. However, this communication exposes another trade-off between accuracy and efficiency; it takes time to send the data to a remote computer, perform some computation, and then return a response to the device [11]. That computation may be more accurate, but achieving that accuracy comes with a cost in speed. Conversely, doing the computation locally on the device would be faster; however, due to the device's more limited computational resources, it will not necessarily be very accurate.
Modified on	28/02/2023 12:01

Name	Regulating Accuracy-Efficiency Trade-Offs in Distributed Machine Learning Systems
Number of Coding References	7
Number of Codes Coding	1
Coverage	16.93%
Folder Location	Codes\\Tradeoff between Maintainability and Scalability
Parent Name	Codes\\Tradeoff between Maintainability and Scalability\\effect and influence between Maintainbility and scalability
Created on	01/06/2022 13:41
Coded Text	3.5 Asynchronous ML Finally, we examine the trade-off in machine learning in asynchronous settings. The examples we have discussed so far are synchronous: there is one computer process9 that does all of the computation, one step at a time. In contrast, it is possible to run computations asynchronously, in which different computer processes or threads10 perform computations side-by-side. This facilitates dividing computationally intensive tasks into parts, such that different portions can happen in parallel and then can be combined to compute the final result. In other words, the parallelization from asynchrony can lead to speed-ups in ML since multiple parts of the learning problem can be computed at once. However, depending on how the parallel results are combined, it can also lead to decreases in accuracy. That is, if different processes end up working on the overlapping parts of the overarching computation, the process that finishes its computation second can overwrite the value computed by the one that finished first, causing inaccuracies in the results [5, 26, 58, 68]. This can be avoided by forcing the different processes to coordinate their updates, such that they do not overwrite each other. However, such coordination takes time; it enables more accuracy, but decreases efficiency. In some cases, this overwriting is worth the speed-ups it enables; it is still possible to compute good quality learning estimates [25, 76].11 So far, our discussion does not take into consideration how the accuracy-efficiency trade-off behaves for ML in realworld, deployed systems—systems that often consist ofmultiple computers that communicate and work together to solve large, complex problems. Such systems often communicate asynchronously: instead of one computer doing multiple sub-computations at the same time (Section 3.5), there are multiple computers operating in parallel on the same problem. In the next section, we discuss how such real-world distributed ML systems raise unique concerns with regard to accuracy and efficiency. 9A computer can run multiple processes at once. Each process is an instance ofa running program—this is why one can run both an Internet browser and a text editor at the same time. In other words, processes allow for parallel tasks to run on one computer [6]. 10A thread is a further mechanism for parallelization on a computer, which operates below the level of a process. That is, a process can have multiple threads running at the same time. For example, this is what allows a text editor (which is running in a process) to simultaneously enable displaying both typing and syntax-error highlighting in real-time. Each of these functions happens in its own thread of computation. 11Asynchrony is complementary to other examples in this section. For example, it can be used in combination with minibatching, low-precision, and in MCMC to implement other types of accuracy-efficiency trade-offs. 7 4 Making Sense of Additional Trade-Offs in Real-World ML Systems Our overarching aim is to understand the particular tensions between accuracy and efficiency for distributed machine learning systems, and how these tensions differ from those we discussed regarding machine learning algorithms in Section 3. To make these distinctions clear, we first clarify some key ideas from distributed computing in Section 4.1.12 From this basis, we can then layer on more complexity in Section 4.2. We weave in our understanding of the accuracyefficiency trade-off for ML algorithms from Section 3 and observe how the different tensions interact with each other. Considered together, we demonstrate how machine learning and distributed systems trade-offs present especially challenging problems for real-time, high-impact systems like autonomous vehicles. These real-time domains inform our policy discussion in Section 5. 4.1 Accuracy and Efficiency in Distributed Systems In contrast with a single, solitary computer, a distributed system is a network of computers that communicate with each other. Via this communication, the computers can work together to solve problems. Each computer in the network has its own data and performs its own computations, and it shares data and computation results with other computers in the network when necessary. For example, if a computer needs data from another computer in order to execute a computation, it can request the data from that computer. Because the computers are in distributed locations and need to communicate, there are important considerations with regard to how efficiently information can be shared between them. That is, when a

computer contacts another in the system to request its data, it takes time to complete the request and receive the data—in direct opposition to efficiency. There are also issues of accuracy between computers in the system. Each computer has its own data—its own snapshot of what it knows to be the state of the overarching system. However, that information is not complete; it is just a subset and can possibly contradict the information that other computers in the system have. Simply put, the computers can be inconsistent with each other. In other words, in distributed systems we can frame the trade-off between accuracy and efficiency as a tension between consistency and latency—the speed with which the system updates. There is a trade-off between all of the computers in the system having the same understanding of the data in the system and the time it takes to propagate that understanding throughout the system [2, 15]. Due to this tradeoff, in distributed systems that update their data frequently it is actually quite difficult to quickly build a consistent, holistic

12We touch on this topic only briefly, since our main focus is the behavior of such systems in the context ofmachine learning For more detailed treatment, see Cooper [21] and Cooper and Levy [22].

understanding of the environment across different computers in the network. This is because consistency is a moving target; each computer processes information locally faster than it can share it with the entire network. Given that it takes time to communicate, it is hard for computers to stay completely up to date with each other. Nevertheless, for the sake of efficiency, individual comput-

ers in the system often need to make decisions in the presence of inconsistency. Otherwise, because of the tension between consistency and latency, waiting for complete consistency across computers before a computer could make local changes would bring the entire system to a standstill. Instead, particular distributed system implementations need to answer the question of how much inconsistency and slowness they can each tolerate, which is often application-dependent. To understand this spectrum, we will consider a few ex-

amples of distributed systems that implement the trade-off differently [27, 43]. First, consider a social media website, which has computers hosting its data distributed all over the world. A user visiting the site from a personal device tends to access the geographically closest computer server hosting the site; different users across the world therefore access different computer servers. Such a system favors efficiency (i.e., low latency) over the different computer servers being consistent with each other. It is more important to return the website to each user quickly than it is to make sure that every user is accessing the website with exactly the same data. This is why on some social media sites it is possible to see out-of-order comments on a feed; the site is making a best effort to resolve its current state, which entails aggregating information from across the system. It attempts to build a consistent picture, but limits how much time it spends doing so—sacrificing consistency—so that it can remain fast [27, 60, 89]. The system implements this choice via its communication strategy. Rather than contacting every computer in the system to construct a coordinated, consistent picture (which would take a lot of time) a particular computer only communicates with a subset. It trades off the accuracy it would get from communicating with every computer for the efficiency of communicating with fewer computers [42, 54]. In contrast to an efficiency-favoring social media site,

blockchain technology is a distributed system for storing a transaction ledger that favors consistency at the cost of being slow [65]. In short, it is a distributed system where each computer has its own copy of the entire ledger. When a computer wants to add a transaction to the system, it has to broadcast that information to every computer in the network. All of the computers need to agree on the validity of a transaction before it can be included. As a result, the

system proceeds in lockstep, only when there is coordinated agreement.13 These different implementation choices reflect different

design goals. The cloud was designed for e-commerce applications, in which supplying (even potentially inaccurate) responses quickly to the user is critical for user engagement [9, 17]. For blockchain systems, consistency is paramount; it is crucial that all of the computers agree with each other about the state of the ledger, because it is this agreement that facilitates its reliability as a transaction record. While these two examples seem to imply that there is

an all-or-nothing choice in the trade-off between consistency and latency in distributed systems, this is not the case. Like accuracy and efficiency more generally in approximate computing (Section 2.2), the trade-off between consistency and latency is a spectrum [2, 94]. It is possible to quantify consistency and to measure and monitor its maintenance throughout a distributed system [60, 81]. Developers can reason about the degree of inconsistency their particular system can tolerate safely, and can detect and tune the system's implementation accordingly to also enforce an upper bound on latency [10, 30, 37, 73, 86, 95].

4.2 Bringing it All Together: Trade-Offs in Distributed Machine Learning Systems

Given this background on how accuracy and efficiency are in tension with each other in distributed systems in general (Section 4.1) and our earlier discussion ofaccuracy-efficiency trade-offs in ML (Section 3), we can now specifically consider real-time (i.e., latency-critical) distributed ML systems. As an example, consider a distributed system of autonomous vehicles. Numerous vehicles are potentially networked together and with other devices, such as smart traffic lights. Moreover, while each vehicle moves throughout the environment with its own local notion of the state of the environment, information that other vehicles possess could also prove useful. For example, if an accident is up ahead, a vehicle closer to the crash can communicate that information to the vehicles behind it, which in turn can apply pressure to their brakes and potentially prevent a pile-up. In such real-time transportation domains,

Modified on	accuracy and efficiency14 are both critical. Some ML inference applications may be error tolerant, but in high-stakes domains this may not always be the case; it is unclear how much inaccuracy will be tolerable while still ensuring safety [12].15 28/02/2023 12:02
Name	Regulating Accuracy-Efficiency Trade-Offs in Distributed Machine Learning Systems
Number of Coding References	7
Number of Codes Coding	1
Coverage	16.93%
Folder Location	Codes\\Tradeoff between Maintainability and Scalability
Parent Name	Codes\\Tradeoff between Maintainability and Scalability\\effect and influence between Maintainbility and scalability
Created on	01/06/2022 13:41
Coded Text	A Call to Action: Enabling the Regulation of the Accuracy-Efficiency Trade-Off We have taken considerable space to clarify a variety of accuracy-efficiency trade-offs—from how they generally impact the field ofcomputing to how they describe the range of possible behaviors for distributed machine learning systems. More specifically, it is necessary and urgent to expose the accuracy-efficiency trade-off because it is a potential lever for regulation. Though various manifestations of the tradeoff are well-acknowledged in technical communities, they have not, to date, been legible to policymakers. We argue that policymakers must understand the implications of the accuracy-efficiency trade-off in order to responsibly regulate emerging technologies—that it is necessary and urgent to expose the trade-off as a potential lever for regulation. As we have documented in Sections 2.2-4.2, this trade-off is not binary; it is a spectrum and can be treated like a tunable dial set appropriately to the context. Our hope is that exposing this dial will provide a certain degree of technical transparency to lawmakers, such that high-stakes systems do not get deployed without sufficient public oversight [21, 22]. Contemporary policy debates about high-stakes, time-sensitive machine learning applications—in domains
Modified on	28/02/2023 12:02

Name	Regulating Accuracy-Efficiency Trade-Offs in Distributed Machine Learning Systems
Number of Coding References	7
Number of Codes Coding	1
Coverage	16.93%
Folder Location	Codes\\Tradeoff between Maintainability and Scalability
Parent Name	Codes\\Tradeoff between Maintainability and Scalability\\effect and influence between Maintainbility and scalability
Created on	01/06/2022 13:41
Coded Text	Accuracy-Efficiency Considerations in Machine Learning Algorithms Several influential papers on artificial intelligence (AI) from the 1980s and 1990s also demonstrate the potentially high impact of appropriately dealing with accuracy-efficiency trade-offs [13, 47]. In particular, in a classic paper, Horvitz poses the question ofhow autonomous agents can effectively perform computations under tight computational resource constraints [47]. He discusses how approximations or heuristics can lead to more efficient resource utilization—at the 4There have been similar findings in other cities such as Detroit [56]. 5We do not argue in favor of this "need for speed" in law enforcement. Advocates for police reform in the US have argued for years that such a "need" is in fact constructed to benefit police in cases of misconduct [74]. 6While this observation speaks to the trade-off between accuracy and efficiency, we do not intend for this to be taken as an endorsement for using risk assessment tools in criminal law domains. We instead apply this example narrowly to explicate accuracy-efficiency considerations, without commenting on the normative implications of what accuracy means in this context or the desirability of the use of such tools. 4 cost of potentially less-correct computation. He frames this as a "time-precision tradeoff,"7 in order to indicate how there is an inherent tension between the utility of a correct computation and how fast that computation is completed, in the context of evaluating reasoning under uncertainty for autonomous agents. This trade-off persists beyond classical AI to contempo- rary work in statistical ML, as ML's probabilistic nature has important implications for the relationship between accuracy and efficiency in ML models. Trained ML models perform inference that is not always correct, often tolerating a certain degree of inaccuracy. Being resilient to errors is necessary for producing robust models. This notion of error resilience (or inaccuracy tolerance) varies for different types of ML algorithms. Regardless of particular differences, there is a general tension between correctness and performance. The correctness of a ML algorithm can be understood as whether or not the algorithm converged to the distribution we set out to learn, i.e., Did we learn the right model? Its performance indicates whether convergence to the distribution—whether correct or incorrect—happened in a timely manner, i.e., How fast did we learn the model? As with other approximate computing problems, ML can relax its demands on accuracy in order to achieve increases in efficiency. In fact, this relaxation is a requirement in many learning domains. Without it, inference computations can be so inefficient to perform that they become intractable. We describe five such cases below. 3.1 Data Subsampling Performance directly relates to the size of the task on which we perform learning. Intuitively, if a learning algorithm is slow on small tasks—that is, tasks with small datasets— then that algorithm will be slow, if not computationally intractable, on much larger ones. More concretely, this relationship between runtime and task size often exists due to coupling between the computation done by the learning procedure's optimization algorithm and the task's dataset size. For example, when computing the gradient needed to determine which direction the learning algorithm should step for its next iteration, it is often necessary to sum over every data point in the dataset. As we show in Figure 2 with the Gradient Descent (GD) algorithm, for larger datasets this summation becomes increasingly costly. A very common approach for improving efficiency is to use a subsample or minibatch of the dataset, rather than the whole dataset, when performing calculations. In the case 7While "precision" and "accuracy" are different, there is a relationship between them. For our purposes, it useful to think of the degree of precision as a mechanism for controlling how much accuracy is possible to achieve when performing a computation. For example, using fewer bits (i.e, low bit precision) to represent numbers can drastically effect the degree ofaccuracy ofcalculations done with those numbers, since this is effectively the same as doing computation on (potentially very highly) rounded numbers (Section
Modified on	28/02/2023 12:01

Name	Regulating Accuracy-Efficiency Trade-Offs in Distributed Machine Learning Systems
Number of Coding References	7
Number of Codes Coding	1
Coverage	16.93%
Folder Location	Codes\\Tradeoff between Maintainability and Scalability
Parent Name	Codes\\Tradeoff between Maintainability and Scalability\\effect and influence between Maintainbility and scalability
Created on	01/06/2022 13:41
Coded Text	more time (i.e., sacrifices efficiency) but can capture a more accurate range of results. Much work in machine learning explores using low- pre- 10 3 10 4 10 5 103 Number of data points Figure 2. The runtime for GD (a full-batch method using the whole dataset to compute gradients) is coupled with dataset size: as dataset size increases, so does runtime per iteration of the algorithm. In contrast, a subsampled, minibatch method like SGD (which here uses only 1 data point to compute the gradient) is decoupled from the dataset size: it maintains a relatively constant runtime per iteration. of computing gradients, instead of using a "full batch" (i.e., the whole dataset) we use a randomly sampled subset of the data points, which entails spending less time on computation. Stochastic Gradient Descent (SGD) is an example of an algorithm that takes this approach. Using a minibatch can often have minimal impact on the overall accuracy of the learned model. A particular iteration of the algorithm will have less accuracy when computing the gradient (Figure 2); but, when run for lots of iterations, the final result can still be statistically correct. In expectation, we can learn the same distribution as if we had been using the whole dataset in each iteration; we can often theoretically guarantee robustness [14, 51]. Moreover, the decision to subsample is not all-or-nothing; it is a spectrum. It is possible to vary the minibatch size the algorithm uses. Larger minibatches—especially those that approach the size of the full dataset—require more time but are also more accurate per iteration. Conversely, smaller batch sizes make each iteration faster and more scalable to larger datasets, but in doing so sacrifice accuracy per iteration. 3.2 Low-Precision Computation Another common approach involves using low-precision representations of the numerical values on which the computer performs computations. This method, sometimes called quantization, is similar to the idea of floating-point precision— how much accuracy the computer can capture based on how many bits it uses to represent numbers—that we discussed in Section 2.2. Computing with more precise floating-point numbers is more computationally expensive; it tends to take 5 104 cision numbers to achieve faster results. This work relaxes requirements on the accuracy of the trained model in order to achieve these speed-ups [4, 24, 26, 38, 39, 41]. As with the minibatching example in Section 3.1, this sacrifice in accuracy does not necessarily require sacrificing overall correctness if in expectation the algorithm can still theoretically guarantee learning the right distribution. There is also a spectrum at play here; similar to varying the minibatch size to tune the trade-off between accuracy and efficiency, it is possible to vary the number of bits of precision. More bits yield higher accuracy and slowdowns, while fewer bits require less time per computation and thus potentially sacrifice some correctness. Depending on a particular application's tolerance to error, this sacrifice in accuracy can be worth the speed-ups it creates [75].
Modified on	28/02/2023 12:01

Name	Regulating Accuracy-Efficiency Trade-Offs in Distributed Machine Learning Systems
Number of Coding References	7
Number of Codes Coding	1
Coverage	16.93%
Folder Location	Codes\\Tradeoff between Maintainability and Scalability
Parent Name	Codes\\Tradeoff between Maintainability and Scalability\\effect and influence between Maintainbility and scalability
Created on	01/06/2022 13:41
Coded Text	Prior work considers how computer vision models can be learned and stored on a mobile device like a smartphone.8 For such resource-constrained devices, different applications have different needs in terms of how to trade-off between how accurately and how quickly a computation is performed. Some prior work has explored these application-specific needs, providing an interface for flexibly implementing different points along the accuracy-efficiency trade-off spectrum. For example, MobileNets contains manually-tunable parameters that allow the model developer to strike the right balance for particular learning problems [48]. Depending on the application domain, the developer can tune a larger model that uses more resources (i.e., a model that is slower but more accurate) or one that is smaller and uses fewer resources (i.e., a model that is faster but less accurate).
Modified on	28/02/2023 12:01

Name	Regulating Accuracy-Efficiency Trade-Offs in Distributed Machine Learning Systems
Number of Coding References	7
Number of Codes Coding	1
Coverage	16.93%
Folder Location	Codes\\Tradeoff between Maintainability and Scalability
Parent Name	Codes\\Tradeoff between Maintainability and Scalability\\effect and influence between Maintainbility and scalability
Created on	01/06/2022 13:41
Coded Text	The way such systems will need to treat efficiency is similar. They will need to make decisions quickly and, much like the non-computing examples in Section 2.1, there is an inherent trade-off between waiting to make a completely informed decision and making a decision fast enough for it to be useful [2, 15]. What is different here is the degree of efficiency needed—in some cases, inference decisions will be necessary at subsecond speeds. In short, it is not entirely clear what the right design goal is for real-time systems like autonomous vehicles and how the trade-off should be implemented for them [28]. Given the dynamic nature of the environment, the particular trade-off implementation may depend on context. Some environments will be more efficiency-critical: it would be catastrophic for a car to take an extra half-second to be certain that there is a pedestrian directly in front of it. In other cases, having an accurate sense of the environment may be more important than allowing the cars to operate quickly. For example, when detecting a deep pothole up ahead, it could be safer for a car to slow down to decide its course of action—to accurately determine if the hole is shallow enough for the car to continue on its course or if the hole is deep enough to warrant veering off the road to avoid it. Distributed ML systems raise different accuracy-efficiency questions than either distributed systems that do not involve ML, or ML systems that are not distributed. With regard to the former, the kinds of coordination and consistency issues that distributed systems can tolerate while maintaining correctness are different in nature than what newer ML systems can tolerate (particularly around issues like numerical error and staleness) [9, 27, 95]. With regard to the latter, as we saw in Section 3, since ML models (necessarily imperfectly) approximate representations of the world, it is possible for ML models to operate on data that are not completely accurate and still yield results that are correct enough—that fall within the same bounds of imperfection that we deem tolerable when operating on accurate data. We can extend such data inaccuracies beyond things like subsampling to include data staleness inherent in asynchronous distributed settings. Allowing for such staleness comes with the benefit of increasing efficiency, as the system would not need to wait to synchronize—to completely resolve staleness issues before proceeding with its computation. Similar to the single computer case, their overall output still can be correct even when operating on numerically imprecise or stale data in a distributed setting; however, existing work in this field does not necessarily guarantee such output must be correct [5, 31, 38, 58, 68, 77, 99]. Instead, prior work has examined this phenomenon at a high level by looking at the correctness and the performance ofend-to-end ML systems, rather than directly evaluating the underlying accuracy-efficiency trade-off. This work focuses on empirical results for tuning the staleness ofthe underlying 9 data storage layer. Tuning has generally either been manual— curated to the particular problem domain—or absent, leaving the user to pick from a few predefined settings that enforce high accuracy, ignore accuracy altogether for efficiency, or attempt some middle-ground, "in-between" approach [3, 46, 53, 57, 71]. Attempts at more flexible trade-offs have entailed very domain- or algorithm- specific solutions [59, 70, 91]. While it is possible to implement any of these different points in the trade-off, current large-scale systems for distributed learning and inference tend to opt for efficiency. They focus on minimizing communication between computers in the system in order to be efficient enough to scale to larger problems. Some of these systems can achieve orders of magnitude in performance improvements by dropping updates without simultaneously destroying correctness [68, 87]; however, it is not clear these approaches will work for realtime distributed ML systems that are safety-critical, such as autonomous vehicles. It will not always be feasible for these systems to lose updates. Existing approaches to mitigate such losses in ML systems involve increasing communication between computers in the system. However, this then impacts the other side of the accuracy-efficiency trade-off, leading to inefficiencies from bottlenecks in coordination between computers.16
Modified on	28/02/2023 12:02

Name	Regulating Accuracy-Efficiency Trade-Offs in Distributed Machine Learning Systems Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Reproducible Model Sharing for AI Practitioners
Number of Coding References	16
Number of Codes Coding	3
Coverage	25.27%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 22:50
Coded Text	act as an individual, self-contained service in our application layer. Using Kubernetes as service orchestration along with asynchronous micro-service architecture, we can allocate computational resources to isolated and dependency-free ML services without creating network and communication bottlenecks. Model Serving and inference is a technique where the ML trained model output is served as an API endpoint to perform prediction, data generation or data transformation. By taking snapshot and storing the final state of an ML training process with all the parameters and leveraging inference-servers like TensorflowServe or ad-hoc execution, we can host ML models. This method perfectly fits into the Micro-Service architecture on top of a Kubernetes cluster. The async-communication handles the long-running inference process over Istio service mesh to preserve security and discoverability. Kubernetes resource allocation policies manage the heavy-duty tasks and resource-hungry ML models. All this is maintained across a cost-efficient, scalable platform.
Modified on	16/02/2023 09:21

Name	Reproducible Model Sharing for AI Practitioners
Number of Coding References	16
Number of Codes Coding	3
Coverage	25.27%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 22:50
Coded Text	BACKGROUND To enable seamless model sharing, MLC builds upon containers and their orchestration, microservices and model serving. We discuss containerized and container orchestration platforms and how
Modified on	16/02/2023 09:20

Name	Reproducible Model Sharing for AI Practitioners
Number of Coding References	16
Number of Codes Coding	3
Coverage	25.27%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 22:50
Coded Text	Comparison with Containerized Inference Platforms In recent years, Cloud Native Computing Foundation (CNCF) ML practitioners take advantage of platforms like Kubernetes for managing ML applications at scale. Platforms like KNative provide automation, security, and discoverability on a single ecosystem by providing multiple application layers. KFServing and KubeFlow are two of the leading ML platforms for serving models on Kubernetes, which are both based on KNative. Although KNative provides a fully managed platform, it requires more RAM and CPU on each Node. As both platforms are designed to support a wide range of models and deployments, they have more complexity for noncloud-engineers. Table 4 shows an overall comparison ofMLC with KF-Serving and Kubeflow in terms of resource consumption and complexity. We believe that the reduced resource consumption and overall much reduced complexity of used make MLC a prime candidate for the adoption toward model sharing for reproducibility purposes by the AI and ML communities at large.
Modified on	16/02/2023 09:24

Name	Reproducible Model Sharing for AI Practitioners
Number of Coding References	16
Number of Codes Coding	3
Coverage	25.27%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 22:50
Coded Text	Inference at scale in cloud computing is achieved through orches- tration platforms like Kubernetes [13]. KFserver [7], Polyaxon [24] and Kubeflow [4] are open-source ML inference platforms, optimized for scalability, high availability, and performance. Managing these platforms requires expert knowledge in cloud, security, networking and distributed processing. Serverless implementations also impose significant resource allocation and deployment effort [18]. Although all these platforms could be used for model sharing, they are not targeted to improve reproducibility and reviewing processes and require large-scale resources for deployment.
Modified on	16/02/2023 09:18

Name	Reproducible Model Sharing for AI Practitioners
Number of Coding References	16
Number of Codes Coding	3
Coverage	25.27%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 22:50
Coded Text	Main Architecture MLC is designed on top of Kubernetes with micro-service design patterns, communicating over Kafka message broker behind Istio Service Mesh. Istio provides an ingress gateway for incoming traffic and reroutes the appropriate services using the Virtual Service resources. Figure 1 presents the main architecture of MLC. API and Run are the two main micro-services processing in- coming requests from Virtual Services. The API micro-service is responsible for managing the Authentication of users and access management of deployments. It also manages to create new Inference deployments and the containerised ML model's build process's lifecycle. The Run micro-service handles the execution ofML models over a distributed container-as-a-function service. As our primary objective is to reduce dependency on cloud plat- forms, MLC provides a container registry for hosting deployed ML containers and Minio FileStorage Services similar to AWS S3, which enables researchers to upload source codes and supplementary artefacts of ML models. As the primary database, we use a self-managed stateful deployment of MongoDB which satisfy the simple purpose of storing the list of the deployed services and their history of execution with regard to each authorised user. We use OpenFaaS as our main container execution engine to control ML models over API endpoints. OpenFaaS is lightweight compared to KNative and a similar platform, making it the perfect candidate for our implementation.
Modified on	16/02/2023 09:21

Name	Reproducible Model Sharing for AI Practitioners
Number of Coding References	16
Number of Codes Coding	3
Coverage	25.27%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 22:50
Coded Text	micro-services can work along with service discoverability tools for serving ML models. Recent improvements in Cloud-based applications have pushed the boundaries of ML training and computational limits further down the line. Hardware limitation is fading, and scaling ML training is becoming painless and straightforward. Training on Petabytes of data on hundreds of powerful GPUs is now possible on public cloud providers. With each iteration of new GPUs and TPUs and increase in availability and cost reduction, we can expect a full transition both for training and inference to cloud platforms. Containerisation is one of the main drivers that enabled the transition of software to cloud-based architectures. By isolating dependencies over the software layer and network access, containers are able to replicate and distribute workloads over a collection of self-contained applications. These portable applications have also attracted the ML community to leverage industry best-practices for shipping software. By using containers, Machine Learners are able to preserve a well reproducible experiment regardless of the runtime environment. This phenomenon has enabled better reproducibility in ML research, driven by automation at the application layer and isolation at the data and information layer [29]. Docker is one the commonly used container run-time platforms adapted by the CNCF and widely used throughout software communities. Container orchestration comes into action as systems grow in the number of containers and communication and connectivity between them gains complexity. Orchestration tools mainly manage lifecycles of containerised applications, horizontal scalability and replications of containers as well as vertical scalability and resource management. Platforms like Docker Swarm [9], OpenShift [22] and Kubernetes [13] are amongst the most commonly used orchestration platforms. Resource management, scalability-tozero and GPU/TPU [19] adoption enabled the ML industry to use orchestration platforms as main drivers ofML applications. Kubernetes works on top of a descriptive configuration man- agement system to control resources across multiple server nodes inside an internal network topology. Resources are building blocks ofthe Kubernetes ecosystem as they communicate with the Kubelet to provision, manage and control containerized applications using CRI-O. Kubelet, by managing the smallest computational units, Pods, structure higher-order complex Deployment in a way that each building block share storage and network resources. Deployments can then replicate copies of Pods for horizontal scaling and managing network routing policies between each deployment. As resource consumption increases with new Deployments and Pods, Kubelet plays a significant role in managing the compute resources and provisioning new Nodes into the Kubernetes Cluster. Kubelet enables Kubernetes clusters to vertically scale while it preserves network sanity and controls pods and deployment lifecycles. By defining node-affinity configuration, we can allocate custom computational machines to a target deployment. Using node-affinity will enable us to mitigate GPU powered Nodes managed by the Kubernetes cluster. As these nodes follow the scalability principles of Kubelet, they also allow large scale distributed training over multiple GPUs and TPUs both at the inference and training layer. Self-management, auto-scalability, scale-to-zero and custom Node creation make Kubernetes a key role player in the feature of ML.
Modified on	16/02/2023 09:20

Name	Reproducible Model Sharing for AI Practitioners
Number of Coding References	16
Number of Codes Coding	3
Coverage	25.27%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 22:50
Coded Text	MLC enables access restriction on deployed models such that only reviewers and model submitters can interact within the MLC Model Showcase user interface. In addition to the use-case in the research community, MLC enables the industry to reproduce, test and evaluate ML models faster and more precise without going through the training process and provide a clear road-map for scalable inference. MLC is an open-source project hosted on Github1. In summary, the contributions of this work are the following: (1) The design and implementation of MLC, which is an infrastructure agnostic model sharing platform, built to help scientific venues, practitioners and industry at large to share and evaluate trained models without re-running expensive training processes, or without sharing confidential training data or parameters. (2) The prototype and initial performance evaluation of MLC, showing that MLC is efficient in sharing trained models. We use three real-world models and run inference on real-world data, deploying MLC in GCP.
Modified on	16/02/2023 09:20

Name	Reproducible Model Sharing for AI Practitioners
Number of Coding References	16
Number of Codes Coding	3
Coverage	25.27%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 22:50
Coded Text	Our prototype, MLC, proposes a scalable, cloud-based architecture, which can be deployed on standalone servers or distributed clusters within institutes or public/private cloud providers. At the infrastructure level, we designed MLC with a scale-to- zero policy for maximum cost efficiency and trained model privacy preservation. We have also introduced automation toolsets to facilitate running MLC internally within universities infrastructures, if needed. Although managing and bootstrapping MLC requires intermediate technical skills, our approach in providing such a platform is to minimise maintenance and ease of use for both parties—the MLC administrator and the MLC users (e.g., practitioners, reviewers). Thus, as a complementary toolset we introduce a Command Line Interface (CLI) for researchers to easily transfer trained models from a local environment to a remote MLC deployment for publishing their findings. Within the CLI, researchers can create shareable links with public, private or limited access for (re)viewers in an interactive presentation view.
Modified on	16/02/2023 09:19

Name	Reproducible Model Sharing for AI Practitioners
Number of Coding References	16
Number of Codes Coding	3
Coverage	25.27%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 22:50
Coded Text	Service-Mesh and network topology come into action as con- tainerised application deduce dependency on one another in form of communications over different network protocols. As systems grow in complexity and increase in dependency on third-party services, communication plays a crucial role in maintaining security, performance and discoverability ofvarious services across a system. The Service-mesh application layer allows individual self-contained software in a micro-service architecture [26] to have clear visibility to other members of the system while preserving and controlling information transfer inside a cluster of containers. Orchestration platforms create a local cluster networking which allocates internal IP addresses for each entity of the system. Service-mesh applications like Istio [16], Consul [14] and Linkerd [21] accommodate on top of this internal network and provide DNS management and service discoverability, network security and load balancing between services. A service-mesh plays a crucial role in network scalability and reliable communication between entities in an orchestration platform. Istio is a service mesh that brings advanced traffic management and observability features and the ability to centrally decide on security policies and rate limitations for the entities in a mesh. By injecting a high-performance Envoy proxy containers to each pod as a Sidecar, Istio can manage inbound and outbound traffic at the POD level. In addition, by enabling mutual TLS, Istio provides an extra security layer over the internal service mesh connections. For MLC, we enable Istio and its security features. Asynchronous micro-service architecture is a way of dis- tributing communication between building blocks of a system so that each entity gains autonomy and reduction in dependencies at the communication layer. By transporting a cluster's incoming request into a messaging bus like NATS [6], Kafka [2] or RabbitMQ [3], we can distribute a request payload to multiple micro-services without creating network bottlenecks. Compared to traditional request-response communication, async communication allows us to run long-running tasks without overhead on the service mesh as requests no longer need to wait for a response. This communication method enables the ML training process or inference to
Modified on	16/02/2023 09:20

Name	Reproducible Model Sharing for AI Practitioners
Number of Coding References	16
Number of Codes Coding	3
Coverage	25.27%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 22:49
Coded Text	Cold Start 4.2 Run-time Some ofthe known challenges in container-as-a-function platforms are the scale-to-zero and cold-start of containers problems. ML containers consist of several run-time packages and the trained model; this means they are usually large in size compared to webbased containers and it takes longer to scale them horizontally. In Figure 4, we demonstrate our results when we execute in- ference on scaled-to-zero models. OpenFaaS provides a built-in queue-worker that is responsible for caching and scaling down of containers. As deployed model have a similar CRD to Kubernetes PODs, they are almost always in the warm state. The results show that the performance penalty of cold starts is not severe, being smaller than 500 ms. We believe this is good enough performance To examine the performance of MLC, we performed ten inference runs for each model; all deployed on one Kubernetes node with 2GB ofRAM and 4 vCPUs. Our results in Table 3 shows MLC platform can run inference on models as big as GPT-2 with 5GB of trained weights and store the data securely in MLC database in less than 4 seconds. We consider this sufficiently performant for the purposes of model sharing and reproducibility. However, with larger models and GPU-based inference, it is trivial to make use of scaled-up containers.
Modified on	16/02/2023 09:23

Name	Reproducible Model Sharing for AI Practitioners
Number of Coding References	16
Number of Codes Coding	3
Coverage	25.27%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 22:49
Coded Text	So far, reproducibility has been tackled for ML, as for other fields, by sharing code and data, or following clear performance evaluation protocols [30]. This approach led to remarkable improvements in reproducibility. As a result, containerizing source-code and data is en route to become routine practice. Although this improves reproducibility [10], the increase in the size of training datasets and trained models, and the complexity of the deployments are major drawbacks for this approach. It is increasingly difficult to retrain models, as this process means utilizing specialized hardware, using large amounts ofenergy, and increasingly larger carbon emissions [23, 27]. At industry level, sharing data might also be impractical due to the inability to share sensitive training data. We therefore make the case for a model sharing platform to seamlessly allow practitioners from both industry and academia to share their models without the need for re-training and without the need to share training data and even the model itself. For (conference) reviewers to reproduce work, the process should be as easy as running inference in a sandboxed environment, with the input of choice. Large-scale models are difficult to retrain, thus reproduce. For example, in 2019 Google published BERT [8], a revolutionary language model with 360M parameters and a trained model of 1.5 GB. Following BERT, GPT-2 [25] published by OpenAI had 1.5 B parameters and almost 7 GB in size. In 2020, OpenAI pushed the language models further by publishing GPT-3 [5] with 176 B parameters and 600GB in trained model size. This extreme growth rate and complexity does not reflect in all ML research but it is significant. Image classification networks, Deep Reinforcement Learning [20] and Generative Networks [11] often require powerful GPUs and TPUs [19] to run for hours or days to reproduce results.
Modified on	16/02/2023 09:15

Name	Reproducible Model Sharing for AI Practitioners
Number of Coding References	16
Number of Codes Coding	3
Coverage	25.27%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	07/02/2022 22:49
Coded Text	The rapid advancements in AI and Machine Learning (ML) technol- ogy, from both industry and academia lead to the need oflarge-scale, efficient and safe model sharing. With recent models, reproducibility has gained tremendous complexity both on the execution and the resource consumption level. Although sharing source-code and access to data is becoming common practice, the training process is limited by software dependencies, (sometimes large-scale) computation power, specialized hardware, and is time-sensitive. Next to these limitations, trained models are gaining financial value and organizations are reluctant to release models for public access. All these severely hinder the timely dissemination and the scientific sharing and reviewing process, limiting reproducibility
Modified on	16/02/2023 09:15

Name	Reproducible Model Sharing for AI Practitioners
Number of Coding References	16
Number of Codes Coding	3
Coverage	25.27%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	Comparison with Containerized Inference Platforms In recent years, Cloud Native Computing Foundation (CNCF) ML practitioners take advantage of platforms like Kubernetes for managing ML applications at scale. Platforms like KNative provide automation, security, and discoverability on a single ecosystem by providing multiple application layers. KFServing and KubeFlow are two of the leading ML platforms for serving models on Kubernetes, which are both based on KNative. Although KNative provides a fully managed platform, it requires more RAM and CPU on each Node. As both platforms are designed to support a wide range of models and deployments, they have more complexity for noncloud-engineers. Table 4 shows an overall comparison ofMLC with KF-Serving and Kubeflow in terms of resource consumption and complexity. We believe that the reduced resource consumption and overall much reduced complexity of used make MLC a prime candidate for the adoption toward model sharing for reproducibility purposes by the AI and ML communities at large.
Modified on	16/02/2023 09:24
Name	Reproducible Model Sharing for AI Practitioners
Number of Coding References	16
Number of Codes Coding	3
Coverage	25.27%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	DESIGN AND IMPLEMENTATION High-performance ML serving on Cloud platforms like GCP and AWS has much complexity and requires maintainability and cloud expertise to be deployed. This complexity has created a barrier for the research community to take advantage of these services for more straightforward use cases like sharing models for research conferences and reproducibility at the inferential level. MLC tackles the maintenance and implementation complexity, focusing on zerodependency on Cloud Providers and is deployable on on-premise institutional infrastructures, as well as public/private clouds. This section will explain our platform's backbone and how con- tainerisedML applications can perform as standalone API endpoints that will use lightweight container schedulers over an asynchronous communication channel. Our platform enables a layer of privacy to researchers source code and a trained model.
Modified on	16/02/2023 09:25

Name	Reproducible Model Sharing for AI Practitioners
Number of Coding References	16
Number of Codes Coding	3
Coverage	25.27%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	Inference at scale in cloud computing is achieved through orches- tration platforms like Kubernetes [13]. KFserver [7], Polyaxon [24] and Kubeflow [4] are open-source ML inference platforms, optimized for scalability, high availability, and performance. Managing these platforms requires expert knowledge in cloud, security, networking and distributed processing. Serverless implementations also impose significant resource allocation and deployment effort [18]. Although all these platforms could be used for model sharing, they are not targeted to improve reproducibility and reviewing processes and require large-scale resources for deployment.
Modified on	16/02/2023 09:25

Name	Reproducible Model Sharing for AI Practitioners
Number of Coding References	16
Number of Codes Coding	3
Coverage	25.27%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	Main Architecture MLC is designed on top of Kubernetes with micro-service design patterns, communicating over Kafka message broker behind Istio Service Mesh. Istio provides an ingress gateway for incoming traffic and reroutes the appropriate services using the Virtual Service resources. Figure 1 presents the main architecture of MLC. API and Run are the two main micro-services processing in- coming requests from Virtual Services. The API micro-service is responsible for managing the Authentication of users and access management of deployments. It also manages to create new Inference deployments and the containerised ML model's build process's lifecycle. The Run micro-service handles the execution ofML models over a distributed container-as-a-function service. As our primary objective is to reduce dependency on cloud plat- forms, MLC provides a container registry for hosting deployed ML containers and Minio FileStorage Services similar to AWS S3, which enables researchers to upload source codes and supplementary artefacts of ML models. As the primary database, we use a self-managed stateful deployment of MongoDB which satisfy the simple purpose of storing the list of the deployed services and their history of execution with regard to each authorised user. We use OpenFaaS as our main container execution engine to control ML models over API endpoints. OpenFaaS is lightweight compared to KNative and a similar platform, making it the perfect candidate for our implementation.
Modified on	16/02/2023 09:25

Name	Reproducible Model Sharing for AI Practitioners Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	RMLIM~ A Runtime Machine Learning Based Identification Model for Approximate Computing on Data Flow Graphs
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	RMLIM~ A Runtime Machine Learning Based Identification Model for Approximate Computing on Data Flow Graphs Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Scalable Deep Learning on Distributed Infrastructures~ Challenges, Techniques, and Tools
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.00%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	Data Management One of the great challenges of large-scale DL is handling the data that is involved. On the one hand, this refers to the management of training data, whose volume easily exceeds the capabilities ofa single disk or multiple disks on a single server. On the other hand, it refers to the management ofthe DL models, both fully trained as well as snapshots ofmodels currently in the training phase. The training andmodel data need to be handled in a suitable manner, while taking into account the available distributed infrastructure, the running training processes and the resource scheduling in the data center. 3.5.1 Training Data. Obtaining large labeled training data sets is a hard problem. One approach to achieve this is to resort to manual labeling. For instance, to build the ImageNet data set, the authors relied on crowd sourcing via Amazon Mechanical Turk, which led to high accuracy of the ACM Computing Surveys, Vol. 53, No. 1, Article 3. Publication date: February 2020. Scalable Deep Learning on Distributed Infrastructures: Challenges, Techniques, and Tools 3:21 labels [40]. However, manual labeling is expensive and time-consuming. Hence, there are several approaches to allow for training with highly noisy training data that can be easily obtained, e.g., from web image search. Xiao et al. [194] embed a label noise model into a DL framework. They train two CNNs: one of the CNNs predicts the label while the other CNNs predicts the noise type ofthe training data set. For training, they first pre-train both CNNs with clean training data. Then, they train the models with the noisy data, but mix in data with clean labels to prevent model drift. Overall, learning from noisy data is a vast research area (cf., e.g., References [119, 168]), which we will not cover in its entirety in this survey.
Modified on	28/02/2023 12:25

Name	Scalable Deep Learning on Distributed Infrastructures~ Challenges, Techniques, and Tools
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.00%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	data-parallel training, they rely on 50G Ethernet, and forego using specialized interconnects such as RDMA or NCCL [1]. Similarly to Facebook, Tencent employs a heterogeneous infrastructure with both CPUs and GPUs. Their deep-learning system Mariana [211] consists of three different frameworks that are optimized for different infrastructures and use cases. Adam is a large-scale distributed system for DL at Microsoft [27]. It relies on a large number of commodity hardware CPU-servers to perform DL training. Besides many system-level optimizations, one of the hardware-centric features ofAdam is that they partition DL models in such a way that the model layers fit in the L3 cache to improve training performance. The paper on TensorFlow [4], a scalable ML framework developed by Google, provides some insights into the infrastructure at Google. Overall, Google follows a different approach from Facebook and Microsoft when it comes to the DL infrastructure. First, they employ TPUs, which are custom ASICs, as opposed to only using commercial-off-the-shelf (COTS) hardware. Second, they exploit specialized interconnects and use multiple communication protocols, such as gRPC over TCP and RDMA over Converged Ethernet (RoCE).2 Distributed TensorFlow supports communication via the message passing interface (MPI) [180].
Modified on	28/02/2023 12:23

Name	Scalable Deep Learning on Distributed Infrastructures~ Challenges, Techniques, and Tools
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.00%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	Deep Learning (DL) has had an immense success in the recent past, leading to state-of-the-art results in various domains, such as image recognition and natural language processing. One of the reasons for this success is the increasing size ofDL models and the proliferation of vast amounts of training data being available. To keep on improving the performance of DL, increasing the scalability of DL systems is necessary. In this survey, we perform a broad and thorough investigation on challenges, techniques and tools for scalable DL on distributed infrastructures. This incorporates infrastructures for DL, methods for parallel DL training, multi-tenant resource scheduling, and the management of training and model data. Further, we analyze and compare 11 current open-source DL frameworks and tools and investigate which of the techniques are commonly implemented in practice. Finally, we highlight future research trends in DL systems that deserve further research.
Modified on	28/02/2023 12:22

Name	Scalable Deep Learning on Distributed Infrastructures~ Challenges, Techniques, and Tools
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.00%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	DISTRIBUTED DEEP LEARNING Training large DL models with vast amounts of training data is a non-trivial task. Often, it is performed in a distributed infrastructure ofmultiple compute nodes, each of which may be equipped with multiple GPUs. This brings a number of challenges. First, the processing resources must be effectively used, i.e., one must avoid stalling ofcostly GPU resources due to communication bottlenecks. Second, the compute, storage and network resources are typically shared among different users or training processes to reduce costs and provide elasticity (i.e., the cloud computing paradigm [9]). To tackle those challenges in DL, research at the intersection ofcomputing systems and DL is receiving growing attention [4, 27, 36, 79, 141, 195]. This becomes evident with new workshops and conferences arising that particularly focus on DL/ML systems research, such as the Conference on Systems and Machine Learning (SysML).1 However, also established communities
Modified on	28/02/2023 12:23

Name	Scalable Deep Learning on Distributed Infrastructures~ Challenges, Techniques, and Tools
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.00%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	Infrastructure To understand the challenges on parallelization, scheduling and data management for DL, we first take a deeper look at the infrastructure on which DL training is performed. We divide the existing work into two categories: Hardware innovations and data-center scale infrastructure applied to real DL workloads. While the former can potentially be used on single compute nodes or small clusters, the latter describes how individual hardware components can be composed into a scalable, distributed infrastructure for DL. 3.1.1 Hardware Components for DL. While early DL deployments were based on clusters of multi-core CPUs, scalability limitations pushed the efforts to exploiting highly parallel hardware, and even developing special-purpose hardware dedicated to DL training and serving. The performance benefits of GPUs compared to CPU depend on many factors, such as whether the job is processing-bound or memory-bound, the efficiency of the implementation, as well as the hardware itself[97]. Both CPUs and GPUs hardware innovates at a fast pace, whichmakes comparisons difficult and short-living. Nevertheless, state-of-the-art infrastructures for DL typically comprise
Modified on	28/02/2023 12:23

Name	Scalable Deep Learning on Distributed Infrastructures~ Challenges, Techniques, and Tools
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.00%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	interdependencies between the model partition and the scheduling problem, which are yet to be fully explored. Additional challenges arise with the advent of dynamic control flow [79, 199]that renders static scheduling infeasible. Park et al. [138]propose layer placement,which is,however, limited to CNNs. STRADS [88] by Kim et al. is a model-parallel ML framework with an advanced scheduler. In particular, STRADS can take into account dependency structures in model partitions and is capable of prioritizing computations. To do so, the user has to implement his training task via three functions schedule, update,and aggregate. While the paper contains example implementations ofclassicalML algorithms such as LASSO and topic modeling, it is not straight-forward to implement a model-parallel DL training job via the STRADS interface. Litz [146] by Qiao et al. is an elastic ML framework that exposes an event-driven programming model. In Litz, computations are decomposed into micro-tasks that are dynamically scheduled on a cluster. The scheduler takes into account dependencies and consistency requirements of the ML model. To enable interruption-free elasticity, the input data is “over-partitioned” across logical executors, which are dynamically mapped to physical resources. This allows even for transparent scaling of stateful workers, i.e., workers that keep local state that is not shared via the parameter servers or directly with peer workers. This property is useful when different model state is affected by the training ofdifferent ranges ofinput data, such that for faster access that portion ofthemodel state is directly kept at the worker. Proteus [59] byHarlap et al. exploits transient resources such as Amazon EC2 spot instances and Google Compute Engine preemptible instances. Its main concepts are a parameter server framework that is optimized for bulk addition and revocation of transient resources, and a resource allocation component that dynamically allocates transient resources to minimize the overall monetary cost per work based on highly dynamic spot markets. CROSSBOW [89] by Koliousis et al. is a decentralized data-parallel DL system that can automatically tune the number of workers at run-time. To do so, the number of workers is increased during the training until no more increase in training throughput can be observed. This way, the available infrastructure can be utilized in an optimal way. Further, CROSSBOW comes with a dynamic task scheduler to execute workers on GPUs based on resource availability. FlexPS [71]by Huang et al. takes on the problem of varying workloads during the execution of ML training. As sources of varying workloads, Huang et al. mention adaptive hyper-parameters (specifically, the batch size), and advanced SGD methods such as SVRG [85]. As a result of this problem, the parallelism degree, i.e., the number ofworkers, needs to be adapted to re-balance the trade-off between communication and computation in data-parallel training. 3.4.2 Multi-tenant. In amulti-tenant environment,multiple training jobs (tenants) share a com- mon set of resources. Hence, a resource scheduler is responsible to schedule the processes of the different tenants on the resources. There is a large variety of general purpose resource schedulers such as Mesos [64], YARN [178], and Borg [179]. However, these are not tailored to the specific properties of DL training tasks. For instance, in DL, the convergence rate of a training task varies over time. Typically, in the beginning oftraining, progress is made very quickly; however, as training evolves over many epochs, the improvements on model accuracy decrease. Further, different DL training jobs may have very different training curves [205]. Taking into account these DLspecific properties allows for formulating new, DL-specific optimization goals, e.g., maximizing the overall training progress over all scheduled training jobs. Hence, new DL resource schedulers are being proposed. Dolphin [98] by Lee et al. is an elastic centralized data-parallel ML framework. In Dolphin, the configuration of the parameter servers and workers is adapted dynamically according to a cost model and continuous monitoring. Here, the configuration refers to the number of servers and
Modified on	28/02/2023 12:24

Name	Scalable Deep Learning on Distributed Infrastructures~ Challenges, Techniques, and Tools
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.00%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	Large-scale Infrastructure for DL. A large-scale DL infrastructure is composed of many inter-connected hardware components that together build a warehouse-scale computer [13]. In this subsection, we review current infrastructures as described by organizations that perform very large DL jobs, such as Facebook, Google, and Microsoft, as well as academic research. Facebook describes its ML infrastructure in a recent paper [62]. They use both CPUs and GPUs for training, and rely on CPUs for inference. To do so, they build specialized CPU-based and GPUbased compute servers to serve their specific needs of training and inference. For training, GPUs are preferred, as they perform better; however, in their data centers, they have abundant capacities of readily available CPUs, especially during off-peak hours, which they also exploit. For inference, they rely on CPUs, as GPU architectures are optimized for throughput over latency, but latency is a critical factor in inference. Interestingly, for inter-connecting training servers in distributed,
Modified on	28/02/2023 12:23

Name	Scalable Deep Learning on Distributed Infrastructures~ Challenges, Techniques, and Tools
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.00%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	Model Architecture and Hyper-parameter Search. Model architecture and hyperparameter search is a crucial problem in DL training. Given a specific task (e.g., image classification), what is the best model architecture (e.g., CNN with howmany layers and what layer dimensions) that can reach the best accuracy? And what are the best hyper-parameter settings to reach model convergence quickly? Finding the answer to those questions is difficult. The typical approach is to repeatedly try out different architectures and hyper-parameter settings to find the best one, i.e., a search based on experimental evaluations [166]. The search can be random [16]orguided by more sophisticated models, such as random forests and Bayesian optimization [73]orevenreinforcement learning [12, 210]. What all of those methods have in common is that they repeatedly spawn new training jobs with new configurations (architectures and hyper-parameter settings) that need to be scheduled on a shared set ofdistributed resources. Here, we discuss scheduling approaches that explicitly take into account workloads that are generated by such search strategies. TuPAQ [166] by Sparks et al. is a system for automatically generating and executing model search configurations. Based on performance profiles provided by a domain expert, TuPAQ automatically optimizes the amount of resources for data parallel training. Batching together training jobs that access the same training data reduces network load and allows for further optimizations in the execution. HyperDrive [148] by Rasley et al. is a scheduler that optimizes the hyper-parameter search more aggressively than TuPAQ does. In particular, HyperDrive supports early stopping of the training of poorly configured jobs. Further, by incorporating the trajectory of learning curves of the trained models, HyperDrive predicts the expected accuracy improvement. Based on that, more resources are assigned to training jobs that have a high expected accuracy improvement compared to other configurations. HiveMind [127] by Narayanan et al. is a system designed to optimize the execution of multiple DL training jobs on a single GPU.
Modified on	28/02/2023 12:25

Name	Scalable Deep Learning on Distributed Infrastructures~ Challenges, Techniques, and Tools
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.00%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	Model Data. Managing the trained models is as important as the training process itself. According to Vartak et al. [177], model management involves tracking, storing and indexing of trained models. The goal ofmodel management is to facilitate the sharing, querying and analyzing of the DL models. To make that possible, there are a number of current initiatives and approaches. To facilitate interoperability between different DL frameworks, the Open Neural Network Ex- change Format (ONNX) [3] is being developed. ONNX is the de-facto standard for exchange of model data between DL frameworks. DL frameworks that natively support ONNX are Caffe2, Chainer [7, 174], CNTK [158], MXNet [24], PyTorch [139], PaddlePaddle [137], Matlab, and SAS [155]. Moreover, model converters are available for TensorFlow [4], Keras, Apple CoreML [33], SciKit-learn [140], XGBoost [193], LIBSVM [22], and Tencent ncnn [128]. ModelDB [177]byVartak et al. is a system for model management that provides automatic tracking of ML models, indexing, and querying via SQL or via a visual interface. Beyond the models themselves, ModelDB also manages meta data (e.g., hyper-parameters of the training process), quality metrics and training and test data sets for each model. ModelHub [116] by Miao et al. is a system that serves a similar purpose as ModelDB. Beyond providing a versioned model storage and query engine and a domain specific language for model architecture and hyper-parameter search, ModelHub also ACM Computing Surveys, Vol. 53, No. 1, Article 3. Publication date: February 2020. 3:22 R. Mayer and H.-A. Jacobsen provides a repository-based model sharing system for easy exchange of DL models between different organizations.
Modified on	28/02/2023 12:25
Name	Scalable Deep Learning on Distributed Infrastructures~ Challenges, Techniques, and Tools
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.00%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	One of the driving factors of the success ofDL is the scale of training in three dimensions. The first dimension of scale is the size and complexity of the models themselves. Starting from simple, shallow neural networks, with increasing depth and more sophisticated model architectures, new breakthroughs in model accuracy were achieved [30, 38]. The second dimension of scale is the amount of training data. The model accuracy can, to a large extent, be improved by feeding more training data into the model [56, 63]. In practice, it is reported that 10s to 100s ofTerabyte (TB) of training data are used in the training of a DL model [27, 62]. The third dimension is the scale of the infrastructure. The availability ofprogrammable highly parallel hardware, especially graphics processing units (GPUs), is a key-enabler to training large models with a lot of training data in a short time [30, 206].
Modified on	28/02/2023 12:22

Name	Scalable Deep Learning on Distributed Infrastructures~ Challenges, Techniques, and Tools
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.00%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	Scheduling and Elasticity The scheduling problem in DL refers to how to map the (possibly parallel) DL training processes to the processing nodes in the distributed infrastructure. We identified three different aspects of scheduling in DL. First, there is single-tenant scheduling (Section 3.4.1): How to map the processes (e.g., workers and parameter servers) of a single tenant, i.e., training job, to the available infrastructure? In case that mapping is dynamic, and we can change the number of training processes (e.g., number of workers and number of parameter servers) as well as the infrastructure (e.g., number of compute nodes), we also talk about elasticity in the scheduling problem. Second, there is multi-tenant scheduling (Section 3.4.2): Given multiple competing training jobs (each having a number ofprocesses), how to map them to the available infrastructure? The multitenant case introduces additional challenges such as a larger complexity and additional requirements or constraints such as fairness among the tenants. Third, there is a specific scheduling problem that concerns the creation of training jobs in DL, namely, the model architecture and hyper-parameter search (Section 3.4.3). This problem is tightly coupled to single-tenant and multi-tenant scheduling. 3.4.1 Single-tenant. In single-tenant scheduling, we assume a dedicated, but possibly dynamic, set of resources (compute nodes, CPUs, GPUs) that is available to host a set of processes that originate from a single DL training job. With training job, we refer to all processes involved in performing the training of a single DL model. Depending on the parallelization method, this may comprise workers that train complete (data parallelism) or partial (model parallelism) model replicas as well as parameter servers. Now, scheduling needs to answer the following questions: (1) Which process is placed on which resource (such as compute node, CPU, or GPU)? (2) When or in what order are the processes that are placed on the same resource executed? (3) When and how are the number of processes and/or resources adapted? In model parallelism, one of the major problems to be solved is to partition the model into multiple parts. We have discussed this issue and state-of-the-art approaches for addressing it in Section 3.2. Once the model is partitioned, the next important questions are where to place the model parts and when to train which partition of the model. As a training iteration of a model partition can only be executed when all input data of that partition is available, there are dependencies in scheduling the different model partitions. Mayer et al. [113] have formalized the scheduling problem in model-parallel DL. While they propose a couple of heuristic algorithms, none of them have been implemented in the context of DL systems. In particular, there are
Modified on	28/02/2023 12:24

Name	Scalable Deep Learning on Distributed Infrastructures~ Challenges, Techniques, and Tools
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.00%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	workers, the distribution of training data across workers and the distribution ofmodel parameters across parameter servers. The system is implemented on top of Apache REEF [188], a framework for distributed applications. Optimus [141] by Peng et al. is a system that dynamically adjusts the number and placement ofworkers and parameter servers of a training job at run-time to achieve the best resource efficiency and training speed. To do so, it builds performance models based on sampling that estimate the number of training epochs needed until convergence and the impact of different configurations (number ofworkers and parameter servers) on the training speed. Then, a greedy algorithm computes the best allocation ofresources toworkers and parameter servers. Considering multiple concurrent training jobs to be scheduled, Optimus aims to minimize the average job completion time. An additional challenge tackled by Optimus is to divide the model parameters onto the parameter servers such that the load is balanced. Compared to the general-purpose scheduling policies Dominant Resource Fairness [49] and Tetris [52], Optimus shows significant improvements in average job completion time and makespan.3 Jeon et al. [78] analyze log traces from a large-scale DL cluster system. In particular, they analyze the trade-offbetween locality constraints and queuing delays for large training jobs that occupy a lot of (GPU) resources. Further, they observe that co-locating different jobs on the same server may significantly impact their performance. Finally, they also analyze failures in DL training and the root causes why they occur. They differentiate between failures caused by the infrastructure, by
Modified on	28/02/2023 12:25

Name	Scalable Deep Learning on Distributed Infrastructures~ Challenges, Techniques, and Tools
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.00%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	challenges in the problem of parameter synchronization. We discuss those challenges and stateof-the-art approaches to tackle them in Section 3.3. The main advantage of data parallelism is that it is applicable to any DL model architecture without further domain knowledge of the model. It scales well for operations that are computeintensive, but have only fewparameters, such as CNNs.However, data parallelism is limited for operations that havemany parameters, as the parameter synchronization becomes the bottleneck [82, 91]. This problem could be alleviated by using larger batch sizes; however, this increases data staleness on the workers and leads to poor model convergence. A further limitation of data parallelism is that it does not help when the model size is too large to fit on a single device. It is worth to note that in many data parallel training schemes, it is assumed or required that the training data is independent and identically distributed (i.i.d.), so that parameter updates computed by the parallel workers can simply be summed up to compute the new global model parameters [196]. 3.2.2 Model Parallelism. In model parallelism, the DL model is split, and each worker loads a different part of the DL model for training (see Figure 5). The worker(s) that hold the input layer of the DL model are fed with the training data. In the forward pass, they compute their output signal which is propagated to the workers that hold the next layer of the DL model. In the backpropagation pass, gradients are computed starting at the workers that hold the output layer of the DL model, propagating to the workers that hold the input layers of the DL model. A major challenge of model parallelism is how to split the model into partitions that are as- signed to the parallel workers [113]. A common approach to find a good model splitting is to use reinforcement learning [117, 118]: Starting from some initial partitioning, permutations on that partitioning are performed, and performance is measured (e.g., for one training iteration). In case ofan improvement, the permutation is maintained, and further permutations are performed, until the measured performance converges. Streaming rollout [47] is a specialized solution that only works for RNNs.
Modified on	28/02/2023 12:24

Name	Scalable Deep Learning on Distributed Infrastructures~ Challenges, Techniques, and Tools
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.00%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	CNTK (Microsoft Cognitive Toolkit) is a DL framework developed by Microsoft and community contributors. The API is available in C++, C# and Python. Additionally, CNTK provides a custom model description language called BrainScript. The model evaluation function can also be used from Java programs. Data-parallel and distributed training is supported out-of-the-box. The 1-bit stochastic gradient descent by Seide et al. [159] is integrated into the framework. CNTK supports the centralized architecture with parameter servers, using asynchronous training or blockwise model update and filtering (BMUF) [23], a variant of bounded asynchronous training. Currently, model parallelism is not supported by CNTK. Extending CNTK is easy. New operators, loss functions, and so on, can be implemented with an API. There were 138 commits to the official Github repository in the past six months, which is a comparably low value. On StackOverflow, there are 488 questions tagged with “CNTK,” an average value compared to other frameworks. Deeplearning4j is a DL framework developed by the company Skymind and community contributors organized in the Eclipse foundation. The framework is written in Java and C++ (for core components), and the API is available in Java, which makes it accessible for Java, Scala and Clojure projects (but not from Python). It supports distributed and parallel training by using Spark. There are two variants of data-parallel training implemented. First, a decentralized asynchronous approach proposed by Strom [167] that also incorporates quantization of gradients. Second, centralized synchronous training with a single parameter server. There is no support for model parallelism. It is easily possible to create custom layer implementations, but more sophisticated customization (loss functions, parallelization configurations, etc.) is not supported. There were 390 commits to the official Github repository in the past six months, which is an average value. On StackOverflow, there are 243 questions tagged with “Deeplearning4j,” a rather low value compared to other frameworks. Keras is not a DL framework, but a DL library that can be integrated into many other DL frame- works, such as CNTK, Deeplearning4j, TensorFlow, and Theano. It is developed as a community project, initiated by F. Chollet. Keras is written in Python, which allows for its easy integration into other Python-based frameworks. Parallel training on GPUs is naturally supported; higherlevel parallelization concepts must be implemented by the DL framework that uses Keras. Model quantization (to 8-bit model weights) is supported directly in Keras. The library is easily extensible with newmodules. There were 310 commits to the official Github repository in the past six months, which is an average value. On StackOverflow, there are 14,630 questions tagged with “Keras,” a very high value compared to other frameworks.
Modified on	28/02/2023 12:26

Name	Scalable Deep Learning on Distributed Infrastructures~ Challenges, Techniques, and Tools
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.00%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	COMPARISON OF DEEP-LEARNING FRAMEWORKS Since the rise of DL, a large number different DL frameworks and tools have been developed and many of them are open source. They implement different concepts of parallelization and distribution, which we have discussed in Section 3. Having a large choice of open-source DL frameworks is one of the drivers of innovative DL research. In this section, we review and compare current open-source DL frameworks and tools. 4.1 Evaluation Criteria We discuss and compare the frameworks according to the following criteria. (1) APIs. DL frameworks should support a large range of programming languages, so that ex- perts from different domains have easy access to them. Moreover, they should provide high-level abstractions so that a running DL use case can be created quickly without many obstacles. (2) Support for distribution and parallelization. In a cloud environment, resources are available abundantly and on demand. DL frameworks should allow for easy and intuitive support for distribution and parallelization without need for custom code. We specifically examine this point with regard to the parallelization methods and optimizations we have discussed in Section 3.Here, we also discuss the possibility for users to fine-tune their deployment according to their needs. This relates to the DL frameworks' support for custom definitions of the DL model and loss functions and developing custom code for parameter servers or custom topologies in decentralized systems. (3) Community. As the field ofDL is dynamically evolving, with newDL model architectures and parallelization methods being proposed, it is crucial for a DL framework to have an active community that discusses and implements the most promising approaches. We measure community activity by the number of commits on the official Github repositories in the past six months (i.e., between October 2018 and March 2019) as well as the total number of topics with the respective tags on StackOverflow4 (https://stackoverflow.com/). We emphasize that we do not discuss and compare the performance ofDL frameworks; a com- prehensive performance evaluation of DL frameworks is out of the scope of this survey article. There are other studies that compare performance, e.g., by Liu et al. [108] or Jäger et al. [77].
Modified on	28/02/2023 12:25

Name	Scalable Deep Learning on Distributed Infrastructures~ Challenges, Techniques, and Tools
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.00%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	MXNet is a DL framework and an Apache project (incubating). Its API is available for C++, Python, Julia, Matlab, JavaScript, Go, R, Scala, Perl, and Wolfram Language. MXNet supports a wide range of parallelization approaches. Model parallelism is supported for multiple GPUs on a single node; there is no support for multi-node model parallelism though. Data parallelism is realized via the centralized architecture with support for using multiple parameter servers via a sharded key-value store. Both synchronous and asynchronous training are supported out-of-thebox. MXNet also supports post-training 8-bit model quantization tailored to the Intel(R) Math Kernel Library for Deep Neural Networks (Intel(R) MKL-DNN) [126]. In the training process, 2-bit gradient quantization with error-feedback is supported [124]. It is easy to implement custom operators or layers as well as loss functions. There were 837 commits to the official Github repository in the past six months, which is an average value. On StackOverflow, there are 455 questions tagged with “MXNet,” an average value compared to other frameworks. PyTorch is a DL framework developed by Facebook and community contributors. Its API is available for C++ and Python. PyTorch has native support for distributed, data-parallel training, aswell as model-parallel training. For data-parallel training, PyTorch implements the decentralized architecture and supports synchronous as well as asynchronous training. PyTorch supports model quantization via the QNNPACK library [147]. Gradient quantization is not supported out-of-thebox. Writing new operators or layers is easily done via extending an interface; it is also possible to write custom loss functions. There were 3,484 commits to the official Github repository in the past six months, which is a comparably high value. On StackOverflow, there are 2,413 questions tagged with “PyTorch,” a rather high value compared to other frameworks. SINGA is a DL framework and Apache project (incubating) that is developed by community contributors. The initiators of the project are from the National University of Singapore. It has APIs in C++ and Python. Singa has native support for distributed, data-parallel and model-parallel training, as well as hybrid parallelism (combining data and model parallelism). Data parallelism is implemented via the centralized approach with support for multiple parameter servers. However, the decentralized architecture can be emulated by employing each worker with a local parameter server. Both synchronous and asynchronous training are supported. There is no support for model or gradient quantization. Customization is more difficult than in the other frameworks: The documentation does not contain any hints on how to implement custom layers or loss functions. There were 44 commits to the official Github repository in the past six months, which is a comparably low value. On StackOverflow, there are no questions tagged with “Singa” or “Apache Singa,” and only one single question is returned when searching for the keyword “Singa.” TensorFlow is an ML framework developed by Google and community contributors. The API is available for C++, Go, Java, JavaScript, Python, and Swift. Additionally, the community offers bindings for C#, Haskell, Ruby, Rust, and Scala. TensorFlow natively supports distributed and parallel training. In particular, it supports both model parallelism and data parallelism. In data parallelism, the centralized approach via parameter servers is supported, using either asynchronous or synchronous training. Trained models can be quantized using TensorFlow Lite [172]. Currently, there is no native support for gradient quantization or communication scheduling. Customization of layers and loss functions is straight forward via implementing the available interfaces. There were 10,930 commits to the official Github repository in the past six months, which is an extremely high value. On StackOverflow, there are 39,334 questions tagged with “TensorFlow,” which is the highest number among all analyzed DL frameworks.
Modified on	28/02/2023 12:26

Name	Scalable Deep Learning on Distributed Infrastructures~ Challenges, Techniques, and Tools
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.00%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Parallelization Methods DL comes with many possibilities for parallelization. Here, we introduce the three predominant parallelization methods in DL, namely data, model and pipeline parallelism, as well as hybrid forms of parallelism. 3.2.1 Data Parallelism. In data parallelism, a number of workers (machines or devices, e.g., GPUs) loads an identical copy of the DL model (see Figure 4). The training data is split into nonoverlapping chunks and fed into the model replicas of the workers for training. Each worker performs the training on its chunk of training data, which leads to updates of the model parameters. Hence, the model parameters between the workers need to be synchronized. There are many
Modified on	28/02/2023 12:23

Name	Scalable Deep Learning on Distributed Infrastructures~ Challenges, Techniques, and Tools
Number of Coding References	18
Number of Codes Coding	2
Coverage	11.00%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Pipeline Parallelism. Pipeline parallelism combines model parallelism with data paral- lelism. In pipeline parallelism, the model is split and each worker loads a different part of the DL model for training (see Figure 6). Further, the training data is split into microbatches. Now, every worker computes output signals for a set of microbatches, immediately propagating them to the subsequent workers. In the same way, in the backpropagation pass, the workers compute gradients for their model partition for multiple microbatches, immediately propagating them to preceding workers. By streaming multiple microbatches through the forward and backpropagation pass in parallel, the utilization of workers can be significantly increased compared to pure model parallelism, where only one batch is processed at a time. At the same time, the advantages of model parallelism are maintained, as a single worker does not need to hold the complete model. Current approaches that support pipeline parallelism are GPipe [70] and PipeDream [57, 58]. 3.2.4 HybridParallelism. Often, DLmodels are complex and composed ofmany different layers that follow a completely different architecture, which, in turn, requires different parallelization methods. Hence, hybrid approaches that mix data, model and pipeline parallelism are common. Mesh-TensorFlow [161] is a language extension of TensorFlow that allows for combining data parallelism and model parallelism. In Mesh-TensorFlow, tensors can be split across a “mesh” of processors (such as CPUs, GPUs, or TPUs). To achieve data parallelism, data is split into shards; to achieve model parallelism, tensors are split along any of their attributes. There are a couple of papers that propose optimizations of parallelization that are manually designed by domain experts. Krizhevsky [91] proposed to apply data parallelism for convolutional and pooling layers, as those layers are compute-heavy and only have few parameters, and model parallelism for fully connected layers, as they are light in computation, but have many parameters. In Google’s Neural Machine Translation System (GNMT) [191] that powers Google Translate, they apply data parallelism, but combine it with hand-crafted model parallelism for each model replica.
Modified on	28/02/2023 12:24

Name	Scalable Deep Learning on Distributed Infrastructures~ Challenges, Techniques, and Tools Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Scalable federated machine learning with FEDn
Number of Coding References	5
Number of Codes Coding	1
Coverage	12.84%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	Cross-device vs cross-silo use cases We can distinguish between two major settings based on the nature of the use-case and participant/client type [2]. In cross device learning the target is typically a very large amount of stateless, low-powered clients engaging in relatively cheap, short bursts of computation. Clients can for example be cell phones, IoT devices or vehicles, and data is partitioned horizontally. For cross-silo use-cases, clients are larger entities with more local computational power and storage capability. Data can be partitioned horizontally or vertically. In cross-device learning, communication overhead and handling failed connections are key challenges, whereas cross-silo learning can be both computationally and communication constrained and models are relatively large (MBs to GBs). FEDn aims to support the requirements across both these axes of federated learning though a horizontally scalable architecture. 2.3 Privacy-enhancing technologies The core contribution of FL in the wider context of privacy-preserving machine learning is to enhance input privacy by allowing training data to remain withing the full control of the data owner. Compared to alternatives such as homomorphic encryption (HE) which allows computation directly on encrypted data [19], FL accommodates a wider range of algorithms. Output privacy deals with the inverse problem of what can be learnt about the input data from making predictions with the resulting global model. Differential privacy (DP) can be used to add controlled noise to carefully chosen stages of the model construction pipeline in order to make it harder to reverse engineer input data [20]. It is common to combine FL with HE and DP [21], at least with synchronous model update primitives [2]. Multiparty computation (MPC) can also be used for secure aggregation of models to minimize risk for information leakage [4]. A comprehensive overview of the current state-of-the art in FL along with a survey of privacy-enhancing technologies and security is provided by Kairouz et al. [2]. In this paper we focus on efficient implementation of the core primitives of FL, but we note that privacy-enhancing technologies, in particular differential privacy and multiparty computation, could be accommodated in the framework with no or limited architectural modifications.
Modified on	16/02/2023 11:08

Name	Scalable federated machine learning with FEDn
Number of Coding References	5
Number of Codes Coding	1
Coverage	12.84%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	Federated machine learning (FL) [1, 2] is a promising solution to the input data privacy problem in machine learning. In FL, multiple parties, or clients, jointly train a machine learning model while keeping all training data local and private. Instead of moving data to a central storage system, computation is brought to data at each local client site, and incremental model updates are computed and then combined into a global model according to some aggregation scheme. In this way, only model parameters are shared. The majority of work has dealt with artificial neural networks [1, 3, 4, 5] but there is also research done on federated versions of other statistical learning methods such as random forests [6]. FL is structurally similar to distributed optimization for statistical learning [7, 8, 9] but differs in a number of important ways. Foremost, in FL there is no control over how the data is distributed over the participating computational nodes. Hence, data cannot be assumed to be balanced or IID across nodes. This is in stark contrast to distributed learning where we are in full control of the data partitioning and are this able to devise optimal partitions for convergence and load balancing. Moreover, whereas distributed learning typically occurs on reliable cluster infrastructure with high network performance and low failure rate, in FL clients can become unavailable at any time, and communication between client and server takes place over a high-latency network (the internet). Consequently, how to optimally balance local training iterations with global synchronization to avoid poor convergence and to minimize communication rounds are central research questions [4, 10, 2]. Benchmark suits targeted at the exploration of federated learning algorithms have also been proposed [11]. Real-world federated learning software needs to be robust, resilient, and highly scalable distributed systems capable of handling both scaling to a large number of edge-clients, and scaling to large model sizes. The design and implementation of such a framework is the main contribution of this paper. Due to a tiered architecture inspired by the MapReduce programming model, FEDn users can implement a wide range of federated learning schemes by adhering to a structured and familiar design pattern, with the framework then assuring that the schemes will be highly resilient and horizontally scalable. FEDn aims to be easy-to-use and agnostic when it comes to the ML-framework used by clients while supporting high-performance federated training in real distributed settings. It is thus well-suited as a tool for research that bridges the machine learning aspects of the problem and the systems aspect of the problem, as well as for running federated learning deployments in production. FEDn has been used to implement a number of federated learning projects including a Federated Electra by the Swedish National Library, Federated Object Detection for Baltic seabirds by AI Sweden and Zenseact researchers, and fully distributed deployments of FEDn is available for wide use by researchers and Swedish industry partners in the strategic edge computing testbed AI Sweden EdgeLab. Links to these examples and others are collected and at the Scaleout organization on GitHub. FEDn is also the core enabling framework for federated machine learning in the SESAR project AICHAIN. We make the following main contributions: • We propose a tiered system architecture based on hierarchical FL and inspired by the MapReduce programming model. In this way we provide a programming pattern for FL applications that ensures highly scalable and resilient federated learning for cross-silo and cross-device scenarios. • We provide a highly efficient open source framework, FEDn, implementing the proposed architecture. FEDn lets a user go from local testing and development to production-grade geographically distributed deployments with no code change. • We provide performance benchmarks based on thousands (cross-device) of geographically distributed clients and for machine learning model sizes ranging from a few kB to 1GB (cross-silo). Systematic and geographically distributed benchmarks of live federated training at this scale has, to the best of our knowledge, not been reported previously. The remainder of this paper is organized as follows. Section 2 gives a background and surveys related work. Section 3 explains the architecture and implementation of FEDn. In Section 4 we demonstrate the performance and scalability of the framework through experiments on both cross-silo and cross-device use-cases. Finally, Section 5 concludes the work and outlines future directions.
Modified on	16/02/2023 11:07

Name	Scalable federated machine learning with FEDn
Number of Coding References	5
Number of Codes Coding	1
Coverage	12.84%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	Federated machine learning promises to overcome the input privacy challenge in machine learning. By iteratively updating a model on private clients and aggregating these local model updates into a global federated model, private data is incorporated in the federated model without needing to share and expose that data. Several open software projects for federated learning have appeared. Most of them focuses on supporting flexible experimentation with different model aggregation schemes and with different privacy-enhancing technologies. However, there is a lack of open frameworks that focuses on critical distributed computing aspects of the problem such as scalability and resilience. It is a big step to take for a data scientist to go from an experimental sandbox to testing their federated schemes at scale in real-world geographically distributed settings. To bridge this gap we have designed and developed a production-grade hierarchical federated learning framework, FEDn. The framework is specifically designed to make it easy to go from local development in pseudo-distributed mode to horizontally scalable distributed deployments. FEDn both aims to be production grade for industrial applications and a flexible research tool to explore real-world performance of novel federated algorithms and the framework has been used in number of industrial and academic R&D projects. In this paper we present the architecture and implementation of FEDn. We demonstrate the framework's scalability and efficiency in evaluations based on two case-studies representative for a cross-silo and a cross-device use-case respectively.
Modified on	16/02/2023 11:07

Name	Scalable federated machine learning with FEDn
Number of Coding References	5
Number of Codes Coding	1
Coverage	12.84%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	It is common to divide federated learning cases in two distinct categories [12]. In horizontal FL the data is assumed to be partitioned by example, i.e. the feature space are assumed to be (nearly) the same between different participants (for 2 Scalable federated machine learning with FEDn example different scientific instruments sharing performance metrics but operating in different conditions and owned by different companies). In vertical FL the data is assumed to be partitioned by features, i.e. different participants might hold different types of information about the same example (for example a bank and an insurance company holding different types of data about the same customer). The present study focuses on horizontal FL applications. The majority of FL solutions in literature fall in the horizontal FL category, often based on Artificial Neural Networks (ANNs), and perform synchronous batch-based training rounds, see e.g. [13, 14]. Recently, Bonawitz et al. [4] proposed a high-level design of a scalable cross-device framework based on TensorFlow where the focus is on horizontal FL training. Kewei et al. [15] introduced a lossless federated training scheme based on secure XGBoost for vertical FL. Federated averaging (FedAvg) [1] is the most widely used method for horizontal FL. It is a decentralized version of stochastic gradient descent (or in general any method that relies on a gradient update scheme) where in one round of training, a subset ofM participating clients receive a copy of the latest global model and execute K local epochs (complete passes over data) of training, updating f(wk) locally on their own private datasets Dk. They then send the updated weights wk to the server which averages those weights:
Modified on	16/02/2023 11:08

Name	Scalable federated machine learning with FEDn
Number of Coding References	5
Number of Codes Coding	1
Coverage	12.84%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	Related work Several open-source frameworks for FL have been developed, including the following ones [18]. 1. TensorFlow Federated (TFF).2 TFF comprises two layers: Federated Learning and Federated Core. The former is a high-level interface that supports users to apply FL without the need for implementing FL algorithms personally. The latter enables users to implement and experiment new or customized FL algorithms. TFF supports the simulation of the distributed training of FL models but executes only on a single machine. 2. PySyft3 leverages deep learning, secure multiparty computation, and differential privacy to train privacypreserving FL models in untrusted environments [21]. The framework is based on PyTorch and provide a native Torch interface. 3. FATE 4 is an open-source framework developed by Webank [22]. It exploits secure computation protocols to train FL models using different algorithms, such as decision trees, logistic regression, transfer learning, and deep learning. 4. FedML 5 is an open-source library developed to facilitate the development and benchmarking of FL algorithms. FedML supports on-device training, distributed computing, and single-machine simulations. Further, it provides generic API design and reference baseline implementations [23]. 5. PaddleFL 6 is an open-source framework that supports the replication and comparison of FL algorithms as well as the deployment of FL systems in distributed clusters. 6. TiFL is a Tier-based Federated Learning System that groups clients into tiers based on their training performance to mitigate the straggler problem caused by the heterogeneity of clients' capabilities or data quantity. [24]. 7. FLOWER7 is a new open-source framework-agnostic by design that promotes various aggregation algorithms and deep learning frameworks (e.g. Tensorflow, MXNet,TFLite and PyTorch). Moreover, Flower supports training and evaluation on heterogeneous real-edge devices and multi-cluster nodes. [25]. In addition to the frameworks mentioned above, OpenFL proposed by Intel specializes in healthcare use-cases [26]. NVIDIA has recently open sourced a standalone python library called NVFlare [27]. Unlike the above mentioned frameworks, the main aim of FEDn is to provide a production-grade and frameworkagnostic distributed implementation with strong scalability and resilience features supporting both cross-silo and cross-device scenarios. To this effect, FEDn implements a two-tiered hierarchical federated learning architecture with a framework based loosely on the well-known MapReduce programming pattern. The work most closely related to ours is the architecture proposed in [4], where the authors seek to provide load-balancing capabilities through replication of local servers (with a similar role as combiners in our terminology, see the following sections). However, to the best of our knowledge no open source implementation associated with the work is available, and our work goes beyond that work in the size and scale for large model updates, and in terms of resiliency features.
Modified on	16/02/2023 11:09

Name	Scalable federated machine learning with FEDn Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Self-adaptive Machine Learning Systems~ Research Challenges and Opportunities
Number of Coding References	7
Number of Codes Coding	2
Coverage	27.50%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	04/02/2022 22:03
Coded Text	Degradation and Repair of Learning-Enabled Components Similarly to traditional components, LECs can fail, leading systems to undesirable states and to require repair of the under-performing components [40]. Thus, in this section we are interested in understanding what can lead an LEC to fail or produce erroneous outputs (Sect. 3.1), and what tactics are available to repair the components that are deteriorating system utility (Sect. 3.2). The different causes of degradation and the applicability of the tactics introduced in Sect. 3.2 will be exemplified in Sects. 4 and 5 using two use-cases from the fraud detection systems' and cyber-physical systems' domains, respectively. 3.1 Causes of Degradation of ML Components' Accuracy ML approaches rely on a data-set composed of multi-dimensional input data and labels. Since this data-set is used for training the ML model, the environment from which these data-points are collected is usually known as the training environment. Then, once the ML model has been trained, it can be used by the system to make predictions in run-time. This is typically considered the testing environment as the model has never seen the current data-points. We will use the notions of training and testing environments throughout this section to introduce typical causes of degradation of ML components. It is generally assumed that the distribution p(y, x) of labels y and multidimensional input data x does not change between training and testing environments. However, when this assumption does not hold, and hence the prior distribution P(x), or the posterior distribution P(y), or any of the conditional distributions P(y|x) or P(x|y) changes, one may be in the presence of a problem commonly known as data-set shift [51,56–58]. This raises the question of whether the current model is still fit for the current environment. As recent work has shown, not all data-set shifts are malign [58]. Thus, an effective SAS should not only detect shifts, but also assess their actual impact on system utility. The literature on ML has investigated several types of data-set shifts that have different characteristics. These different characteristics influence the impact that each type of shift has on a given system, and also how easy it is to deal 138 M. Casimiro et al. with/detect the shift. Specifically, problems such as anomaly detection, novelty detection, open set recognition, out of distribution detection, and outlier detection [70] are specific instances of the most common types of shift. We argue that the different types of shift are general enough to be representative of most of the issues addressed by the existing ML literature. The following paragraphs introduce these types of shift and give examples of typical sources of shift that can affect an LEC. Covariate Shift. When the distribution of the inputs to a model changes, such that it becomes substantially different from the distribution on which the model was trained, we find ourselves in the presence of a problem commonly known as covariate shift. That is, the distribution P(x) changes but the conditional distribution P(y|x) remains the same. More formally, P (x)train ~= P(x)test and P(y|x)train = P(y|x)test [48]. This type of shift is usually analyzed to evaluate how a model generalizes and how robust it is when the feature space is altered at test time, i.e. while the system is executing. Prior Probability Shift (Label Shift). Differently, when we are in the presence of prior probability shift, also known as label shift, this means that the distribution P(y) of the labels/outputs has changed, i.e., the class proportions differ between training and test. More formally, P(y)train ~= P(y)test and P(x|y)train = P(x|y)test [48]. This can be seen as the inverse of covariate shift in the sense that now the distribution of features is the same between training and testing while the distribution of the labels changes. Dealing with this type of shift is particularly challenging when the new distribution P(y)test is unknown. Concept Shift. Finally, concept shift corresponds to a change in the relationship between input and output distributions, although each remains the same. More formally, when this type of shift occurs, we can have P(y|x)train ~= P(y|x)test and P(x)train = P(x)test or P(x|y)train ~= P(x|y)test and P(y)train = P(y)test [48]. Although these are the most common types of shift, it is also possible that other types of shift occur. We list them for completeness but give examples only of the most common ones in the remainder of the paper. Other types of

shift that can happen are for instance when both the conditional distribution and the features/labels distribution changes. Formally, this would correspond to P(y|x)train ~= P(y|x)test and P(x)train ~= P(x)test or P(x|y)train ~= P(x|y)test and P(y)train ~= P(y)test [48]. These types of shift are typically less investigated in the literature since they are not so common in real world applications and also because they are extremely difficult to detect and deal with.

Sources of Data Shift. During the normal operation of a system, shift in the data can occur due to several reasons: to the passing of time, incorrect data or sample selection bias. Next, we provide details on each of these sources of shift.

Self-adaptive Machine Learning Systems 139

Natural Drift due to Time. An effect of the natural passing of time is that people's tastes and behavior patterns change [7,43]. For example, due to the passing of time and due to inflation, the value of money decreases. A static LEC, that is never adapted and does not account for these natural changes will gradually start producing worse predictions.

Incorrect Data. This problem arises when there are samples in the model's training set that are incorrectly labeled [66] or when test data is tampered with, thus leading the model to mispredict for inputs with specific characteristics. The former can happen for instance when unsupervised techniques are used to label examples in order to bootstrap the training set of a second supervised model [66]. Incorrect data can also make their way into a model's training set due to attackers that intentionally pollute it (e.g., by maliciously altering some of the input features) so as to cause the ML component to incorrectly predict outputs for certain inputs [33,35]. Finally, noise and uncertainty, due to sensor errors or due to errors from upstream components, may also change the input data to an LEC, possibly causing drift and mispredictions.

Sample Selection Bias. This occurs when selecting data-points for a training set or when performing data cleaning1. While selecting data-points, there may be environmental factors that cause some inputs or labels to be sampled more often. For example, when selecting participants for a survey, steps must be taken to ensure that the population of interest is accurately represented. Similarly, when performing data cleaning, for example for a digit recognition task, less clear digits may be thrown away. However, this may prevent the model from learning that some digits are intrinsically harder to write than others [57]. During these, arguably critical, phases of model construction, sample selection bias will cause the training distribution to follow a different distribution than the test distribution, hence causing data shift and potentially a drop in system utility.

3.2 Repair Tactics

Table 1 illustrates a collection of tactics that can be used to deal with issues introduced by ML-based components caused by the different types of shift previously introduced. These tactics were inspired by research on ML [12,46,52,61,67]. Next, we describe the tactics presented in the table, discussing their costs and benefits, and motivating them in the following sections with scenarios from enterprise systems (Sect. 4) and cyber-physical systems domains (Sect. 5).

Component Replacement. This tactic assumes the existence of a repository of components and respective meta-data that can be analyzed to determine if there exists a component that is better suited for the current system state, i.e., that is expected to lead to a higher system utility. If such a component exists, then this 1 In ML, data cleaning corresponds to the process of identifying and correcting errors in a dataset that may negatively impact a predictive model.

140 M. Casimiro et al.

Table 1. Examples of general adaptation tactics for ML-based systems with their strengths ('+') and weaknesses ('–'). Tactic

Description

Component replacement

Replace an under-performing component by one that better matches the current environment

Human-based labeling [46]

Rely on a human to classify incoming samples or to correct the labeling of samples in the training set

Transfer learning [52]

Reuse knowledge gathered previously on different tasks/problems to accelerate the learning of new tasks

Unlearning [12] Remove samples that are no longer representative from the training set and from the model

Retrain [67]

Retrain with new data and maybe choose new values for the ML model's hyper-parameters

Properties

+ Fast and inexpensive, when possible – Alternative components may not be available in all scenarios – Alternative estimators, when available, may be more robust but less precise

+ Accuracy of human-based labels

expected to be high – Expert knowledge may be expensive to obtain and/or introduce unacceptable latency

+ Less data-hungry than plain retrain – Effectiveness dependent on the similarities between old and new tasks/data – Computationally intensive process

	+ Fast when ratio between data to forget and data-set size is small – Cost/latency for identifying examples to unlearn can be large and context-dependent + Generic and robust method – Computationally intensive process – Accuracy and latency of the retrain process may vary significantly – Effective only once a relatively large number of instances of the new data are available tactic will replace the under-performing component by the one that is expected to be better. A benefit of this tactic, whenever it is available, is to enable a swift reaction to data-set shifts. Its main cost depends on the latency and resources used for the analysis of the candidate components available in the repository. Additionally, alternative components may not always be available and, when they do exist, they may be less precise albeit more robust. Human-Based Labeling. Humans are often able to recognize patterns, problems, and objects more accurately than ML components [46]. Thus, depending on the domain, humans may play a role in correcting these components or giving them correct samples [46,65]. For example, when an LEC is highly uncertain about a specific input, it may rely on a human to provide a label. Similarly, if incorrect data is found on a model's data-set, a human may be asked to correct those samples. While this tactic may provide high benefit if the human is an expert, it also has a high cost, since humans are expensive. Also, if there is a significant amount of samples to label, the latency of the process may be unacceptable. Transfer Learning. Transfer learning (TL) techniques leverage knowledge obtained when performing previous tasks that are similar to the current one so that learning the current task becomes easier [42,52]. For this tactic to be Self-adaptive Machine Learning Systems 141 applicable, it is necessary to evaluate the similarity between the source and target tasks/domains. Transferring knowledge between dissimilar tasks will not provide benefits. In order to compute this similarity, metrics such as LEEP [49] can be used. The advantages of this tactic are that it requires less data than is needed to retrain a model from scratch, thus allowing for a quicker model initialization phase. However, the process of TL, similarly to a model retrain, is also computationally intensive. Unlearning. This tactic corresponds to unlearning data that no longer reflects the current environment/state of the system and its lineage, thus eliminating the effect of that data on current predictions [12], while avoiding a full model retrain. A key problem that stands in the way of the execution of this tactic is the identification of incorrect labels. In scenarios in which the identification of incorrect samples is not readily available, one may leverage automatic techniques, such as the one described in [13], which are faster but typically less accurate than relying on humans. As such, the cost and complexity of this adaptation tactic vary depending on the context. Then, after identifying the incorrect samples, the model must be updated to accurately reflect the correct data. The advantage of unlearning techniques with respect to a typical full model retrain is the time savings (up to several orders of magnitude [12]) that can be achieved. Retrain and/or Hyper-parameter Optimization. This is a general tactic that involves retraining the model with new data that reflects recent relevant data-set shifts. There are many types of retraining, ranging from a simple model refresh (incorporate new data using old hyper-parameters), to a full retrain (including hyper-parameter optimization, possibly encompassing the search for different model types/architectures [26]). These imply different computational costs and lead to different benefits in terms of model accuracy improvements. In the presence of data-set shifts, when there is new data that already incorporates the new input distribution, this tactic often represents a simple, yet possibly expensive, approach to deal with this problem. However, this tactic usually requires a substantial amount of data to yield highly accurate models and is computationally intensive. The benefits of this tactic are dependent on the type of retrain process and on the quality of the new data. As for its cost, if retraining is performed on the cloud, it can be directly converted to the economic cost of the virtual machines. Several techniques exist to predict such costs [2,16,45,69].
Modified on	16/02/2023 11:23

Name	Self-adaptive Machine Learning Systems~ Research Challenges and Opportunities
Number of Coding References	7
Number of Codes Coding	2
Coverage	27.50%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	04/02/2022 22:03
Coded Text	Monitor The Monitor stage has to keep track of the inputs received by the ML components because shifts of the input distributions may affect the predictions. For instance, the detection of out-of-distribution inputs may mean that there has been a change in the environment and thus the model used by some ML component may no longer be representative of the current environment. The challenge here is not only detecting the occurrence of shifts in a timely and reliable fashion, but also how to effectively characterize them—since different types of shifts require different reaction methods. As in other SAS, typical attributes that contribute to the system's utility (e.g., latency, throughput) or the satisfaction of required system properties must be monitored. In addition to these, the Monitor stage must also gather the outputs of the ML component to account for situations in which changes in the inputs go by unnoticed, perhaps because they are too slow, but that manifest themselves faster in the outputs [72]. Examples of outputs to monitor are, for instance, shifts in the output distribution, model's accuracy and error—obtained by comparing predictions with real outcomes. A relevant challenge here is that often real outcomes are only known after a long time, if ever. For instance, in fraud detection, false negatives (i.e., undetected real fraud) are known only when users file a complaint. Approaches such as those proposed in [56,71,72]provide a good starting point for the implementation of a Monitor for self-adaptive learning-enabled systems. Challenges. Monitoring input and output distributions requires keeping track of a multitude of features and parameters, which would otherwise be disregarded. This is already challenging due to the amount of data that needs to be stored, Self-adaptive Machine Learning Systems 149 Self-Adaptive Learning Enabled System MANAGING SYSTEM Is LEC N affecting system utility or non-LEC M ? Monitor Aggregate, filter, or pre-process LEC inputs and outputs sensors MANAGED SYSTEM LEC 1 Non-LEC 1 sensors Environment Fig. 1. MAPE-K loop over an LES with a mix of LECs and components that are not learning-enabled, with specific challenges for each MAPE-K stage. White arrows represent dependencies between components. maintained, and analyzed. Finding suitable frequencies to gather these data and adapting them in the face of evolving time constraints is an even bigger challenge in time-critical domains [5,56]. 6.2 Analyze The Analyze stage is responsible for determining whether degradations of the prediction quality of LECs are affecting (or are predicted to affect) other system components and system utility to such an extent that adaptation may be required. To accomplish this, one can leverage techniques developed by the ML community to detect possible issues in the inputs and outputs of the LEC [56– 58,72], errors in its training set [1] and the appearance of new features relevant for prediction [54]. These techniques must then be adjusted for each system, which includes adapting them to different ML models and tasks. … … LEC N Non-LEC M actuators Analyze outcomes of past adaptations What are the costs & benefits of LEC repair tactics? Plan

Knowledge available
components & meta-data
environment model actuators Execute
How to efficiently deploy the selected LEC repair tactic?
Tactics
150 M. Casimiro et al.
Challenges. Estimating the impact of an LEC on other system components and on system utility can be challenging because often (mis)predictions affect the system's utility/dependability in ways that are not only application- but also context-dependent. For instance, during periods with higher transaction volumes, such as on Black Friday, mispredictions have higher impact on system utility, since during these periods it is more critical to accurately detect fraud, while maximizing accepted transactions. Architectural models can capture the information flows among components, but the challenge is to estimate how the uncertainty in the output of the LECs propagates throughout the system.
6.3 Plan
The Plan stage is responsible for identifying which adaptation tactics (if any) to employ to address issues with LECs affecting the system. As with other selfadaptation approaches, this reasoning should consider the costs and benefits of each viable tactic. Further, most of the proposed tactics have a non-negligible latency, which needs to be accounted for as in latency-aware approaches [47]. An additional concern is that some of these tactics may require a considerable use of resources to execute, either in the system itself or offloaded. This requires Plan to account for this impact or cost. For LESs that rely on multiple LECs, whenever a system property is (expected to be) violated or when system utility decreases, fault localization may be required to understand which component is under-performing and should be repaired/replaced [22].
Challenges. Although there are several approaches [2,16,69] that attempt to predict the time/cost of training ML models, this is a complex problem that is strongly influenced by the type of ML models considered, their hyper-parameters and the underlying (cloud) infrastructure used for training. These techniques represent a natural starting point to estimate the costs and benefits of adaptation tactics such as the ones presented. Yet, developing techniques for predicting the costs/benefits of complex tactics, e.g., unlearning, remains an open challenge. One interesting direction is to exploit techniques for estimating the uncer-
tainty [51] of ML models to quantify both the likelihood of models' mispredictions as well as the potential benefits deriving from employing corrective adaptation tactics. While some ML models can directly estimate their own uncertainty [50], others require additional techniques (e.g. ensembles [9]) to obtain uncertainty estimations. Still, existing techniques can suffer from significant shortcomings in practical settings [51]. Finally, tactics that modify LECs are typically computationally expen-
sive (e.g., non-negligible latency). Thus, Plan must have mechanisms to verify that the system can execute the tactic without compromising other components/properties, or even the entire system.
6.4 Execute
To execute a given adaptation tactic, the Execute stage must have access to mechanisms to improve or replace the LEC and/or its training set. As in the
Self-adaptive Machine Learning Systems 151
conventional MAPE-K loop, we require implementations of adaptation tactics that are not only efficient to execute, but also have predictable costs/benefits and are resilient to run-time exceptions.
Challenges. A key challenge is how to enhance the predictability of the execution of the ML adaptation tactics, which often require the processing of large volumes of data (e.g., to re-train a large scale model) possibly under stringent timing constraints. We argue that the community of SAS would benefit from the availability of open-source software frameworks that implement a range of generic adaptation tactics for LECs. These frameworks would allow to mask complexity, promote interoperability and comparability of SAS. Further, it would also provide an opportunity to assemble, in a common framework, techniques that have been proposed over many years in different areas of the AI/ML literature.
6.5 Knowledge
Finally, the Knowledge module is responsible for maintaining information that reflects what is known about the environment and the system. As in traditional systems, in the case of self-adaptive LESs Knowledge also needs to maintain information about the environment so that trends can be observed. These trends can be crucial to detect the shifts that may lead to mispredictions. Additionally, for LESs, the Knowledge component should evolve in order to keep track of the costs/benefits of each tactic on the affected LECs and system's utility. This corresponds to gathering: knowledge on how each tactic altered an LEC and on the context in which the tactic was executed; and meta information on training sets, for instance characterizing the most important features for predicting the costs and benefits of the different tactics. This added knowledge should be leveraged to improve the decision making process and thus improve adaptation. By gathering knowledge on how each tactic altered an LEC and on the context in which the tactic was executed, the Analyze and Plan stages can take more effective decisions on when to adapt and which tactic to execute, respectively. Finally, for a tactic that replaces under-

Modified on	16/02/2023 11:24

performing LECs, Knowledge must contain a repository of the available components and their meta-data. This meta-data, we argue, should provide information to enable reasoning on whether the necessary preconditions to enable a safe and timely reconfiguration hold.

Name	Self-adaptive Machine Learning Systems~ Research Challenges and Opportunities
Number of Coding References	7
Number of Codes Coding	2
Coverage	27.50%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	04/02/2022 22:03
Coded Text	Repair Tactics This section illustrates in which scenarios each of the previously introduced adaptation tactics could be applied in the context of smart homes. Component Replacement. This tactic could be applied to enhance system utility in cases when the face recognition system behaves poorly for example due to low lighting at night, or due to sun rays illuminating faces at different angles at dusk or dawn. In such situations, the face recognition LEC could be replaced by a different component (LEC or non-LEC) that is known to perform better under the current environmental conditions. A benefit of this tactic, for example when compared with a retrain tactic that could potentially be applied in the same situation, is a quick improvement of system utility due to a speedy replacement of the under-performing component by one that is guaranteed to achieve a minimum desired system utility. Human-Based Labeling. When adversaries have manipulated smart meters, such that the appliances currently working in a home cannot be properly detected, a human can be queried to clarify which appliances are working, thus improving the accuracy of the LEC and enabling an increase of system utility by contributing to reducing energy consumption. Self-adaptive Machine Learning Systems 147 Transfer Learning. This tactic can be applied to bootstrap LECs of houses/rooms with new inhabitants [7]. Specifically, since different homes can have different configurations of sensors, rooms, and occupancy, the LECs will require fine-tuning not only to each user but also to each house. In this setting, TL techniques (e.g., based on multi-task Bayesian Optimization [64]), can exploit knowledge gathered from homes with similar configurations and whose LECs have already been optimized. Unlearning. This tactic could be applied for example when adversaries change smart meter data. In this context, this tactic could be applied to forget these incorrect data points so as not to pollute the LEC's training set. Similarly, this tactic could also be applied when the behaviors of the inhabitants have changed. In such a situation, unlearning old behaviors may be a suitable tactic to prevent mispredictions. However, the benefits of applying this tactic are dependent on the amount of data that needs to be forgotten: in case there are plenty of examples to forget, retraining may actually be faster at achieving the same results [12]. Retrain and/or Hyper-parameter Optimization. In a smart home, the tactic of retraining an LEC could be available in a self-adaptive LES repertoire to deal with scenarios such as when inhabitants have new routines, which may cause occupancy prediction LECs to misbehave. In this situation, the LEC can be retrained on new examples that represent these new routines, thus improving the quality of its predictions and, ultimately, user satisfaction and thermal comfort. Similarly, whenever a voice recognition system fails to recognize a voice or a control, it is possible to retrain the model with examples of the voice/control such that it learns to predict them. This tactic can also be applied in settings in which modifying the model structure also yields better predictions. For example, for face recognition systems used for security purposes, which have more stringent deadlines and require higher accuracies, a retrain tactic could train a new type of model to replace the old one, ensuring that the new model is faster (e.g., replacing neural networks by decision trees) or train a set of models to increase confidence in the predictions [9]. Table 2 summarizes the examples of situations provided in the previous sec- tions that can occur in each domain (enterprise systems and cyber-physical systems). Each situation is an instance of each type of shift, and each tactic exemplifies how to deal with the shift in the different situations. 6 MAPE-K Loop for Learning-Enabled Systems In SAS, the MAPE-K loop typically actuates over a system composed of traditional components, i.e., non-LEC components. However, as illustrated in Fig. 1, LESs generally encompass both non-LEC and LEC components. We argue that the MAPE-K loop should be revised in order to be able to cope with the unique issues described in Sect. 3.1 that affect LECs by effectively leveraging the adaptation tactics presented in Sect. 3.2. In the following, we discuss the research 148 M. Casimiro et al.

	Table 2. Examples of causes of ML misbehavior within each domain—enterprise systems (ES) and cyber-physical systems (CPS)—and tactics available for adaptation in each scenario. Problem Domain Example situation Covariate shift ES CPS Transaction patterns change Adversaries poison data Noise/uncertainty in sensors Different lighting conditions for face recognition LEC Adversaries manipulate Label shift ES CPS Concept shift ES CPS Smart meter data Variable fraud rate Inhabitant’s living patterns Applicable tactics • Component replacement • Unlearning • Transfer learning • Component replacement • Human-based labeling • Unlearning • Human-based labeling Unknown command for voice controller • Human-based labeling New fraud strategies • Transfer learning • Retrain • Unlearning challenges and opportunities that arise in each of the MAPE-K loop stages due to the LEC specific adaptation tactics.
Modified on	16/02/2023 11:24

Name	Self-adaptive Machine Learning Systems~ Research Challenges and Opportunities
Number of Coding References	7
Number of Codes Coding	2
Coverage	27.50%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	04/02/2022 22:03
Coded Text	Repair Tactics This section motivates the applicability of each repair tactic by providing examples of their usage when different causes of degradation have affected the system's LECs. Component Replacement. When the volume of transactions changes, for instance during special days such as Black Friday, ML models may consider the increased frequency of transactions as an indicator of fraud and erroneously flag legitimate transactions as fraudulent. Such mispredictions can lead to significant financial losses [8], thus requiring timely fixes that render the use of high latency tactics infeasible (note that in this context transactions need to be accepted/rejected within a few hundreds milliseconds [8]). As such, only low latency tactics can be applied. An example is to replace the under-performing models with rule-based models, e.g., developed by experts for specific situations, and/or to switch to previously trained models that are known to perform well in similar conditions. Human-Based Labeling. Whenever the ML component suspects a transaction of being fraudulent it can automatically block that transaction. Then, the user can be informed of the decision and asked whether the transaction should be authorized or declined in the future. Another possibility is to add humans to the loop when adding samples to the ML component's training set. In this scenario, an expert can be asked to review the most uncertain classifications so as to improve 144 M. Casimiro et al. the quality of the training samples. In the former scenario, the benefits are easily quantifiable, since the risk of accepting a possibly fraudulent transaction can be measured via its economic value. However, users may get annoyed if their transactions are canceled too often, to the extent that they may stop purchasing using that credit card provider. As for relying on experts to review uncertain classifications, having an on-demand expert performing this task is expensive and the latency of the manual labeling process may be unacceptable2. Transfer Learning. Suppose that: (i) a fraud detection company has a set of clients (such as banks), (ii) the company has a unique ML model for each client, so that it complies with data privacy regulations, and (iii) one of its clients is affected by a new attack pattern, which is eventually learned by that client's model. In this scenario, TL techniques can be used to improve other clients' models so that they can react to the same attack pattern. In fact, since privacy is important in this domain, there are techniques that can be used to deal with the problem of ensuring data confidentiality and anonymity in information transfer between clients [29,42] instead of typical TL techniques that do not provide this assurance [52]. Estimating the benefits of executing this tactic for a given client boils down to estimating the likelihood that this client may be targeted by the same attack, which comes at an added cost and time. Yet, the execution of this tactic typically implies high computational costs (e.g., if cloud resources are used) and non-negligible latency, which may render this tactic economically unfavorable, or even inadequate, e.g., if the attack on a different client is imminent and the TL process is slow. Unlearning. In the domain of fraud detection, if after a specific amount of time (e.g. 1month) the fraud detection system does not receive complaints about a set of transactions, these will be labeled as legitimate. However, since users typically take a long time reviewing their statements and complaining when they do not recognize some transactions, it is possible that there are incorrectly labeled transactions in the data-set. In this scenario, and if a model has been trained with the incorrect samples, it is possible to leverage this tactic to remove the incorrect samples from the model without requiring a full model retrain. Retrain and/or Hyper-parameter Optimization. Full model retraining can be leveraged for example when there is a new fraud pattern for which there is already enough data. By retraining the model with this data, the model is likely to increase the amount of fraudulent transactions detected, thus also increasing system utility. However, as this is a slow tactic, while it executes system utility is likely to either drop or remain as unsatisfactory as it was prior to the execution of the tactic. To prevent such situations, hybrid planning approaches can be leveraged [53] to execute a swift tactic that slightly improves system utility while the slow tactic is executing. 2 Fraud detection systems normally rely on a fixed set of humans at any given time. This determines a

Modified on	16/02/2023 11:23 maximum load of transactions that can be processed with a human in the loop.
Name	Self-adaptive Machine Learning Systems~ Research Challenges and Opportunities
Number of Coding References	7
Number of Codes Coding	2
Coverage	27.50%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	04/02/2022 21:53
Coded Text	Causes of Degradation of ML Components' Accuracy Similarly to the enterprise systems domain, in the CPS domain one must also analyze the possible causes of ML mispredictions. As such, we now provide examples of each type of ML misprediction for the smart home example. Covariate Shift. An example of covariate shift in the context of smart homes consists of noise/uncertainty in sensors that measure air quality. This noise changes the distribution of features that go into the predictive model possibly leading a ML classifier to mispredict air quality and thus mispredict the need to ventilate a room. Noise and uncertainty could also affect smart meters used to reduce energy consumption. Finally, different types of light (night versus day, dusk versus dawn) illuminating faces may lead a face recognition system to mispredict whether someone is an intruder. This example is an instance of covariate shift since the distribution of the features is altered due to the different types of light but the distribution of labels is the same and the conditional distribution of labels given the features also remains the same. It is also possible, in both domains (CPS and enterprise systems), that faults in the input data fed by some system component (LEC or non-LEC) to an LEC lead to mispredictions. This highlights the need for a system-wide perspective that considers both LEC and non-LEC aspects of the system. 146 M. Casimiro et al. As an example of LEC mispredictions in the CPS domain due to incorrect data, the literature on adversarial ML has shown how if a data point has been adversarially manipulated, a classification model using smart-meter data as input may not correctly identify which appliances are functioning in a smart home, thus not being able to properly reduce energy consumption [62]. Prior Probability Shift (label shift). Voice controllers in smart homes serve the purpose of executing commands issued by users. In such a setting, a voice controller in a smart home is subject to label shift when a command which was very rarely used is now used very often. Since the voice controller does not account for a change in its frequency of use, it will often mispredict the required action to execute when the command is issued. In this situation, the features and the relationship between features and labels is the same, but the actual correct class that should be derived from those features has changed, thus leading to an error. Concept Shift. Smart homes rely on models that predict inhabitants' activity patterns. When these patterns are altered [7] this corresponds to concept shift. This can happen for instance due to big life events such as the birth of a child, adoption of a new pet, or having visitors stay over for a few days. When the patterns change, the features that are fed to the model do not change, neither do the labels. The only change is the relation between the two distributions which can cause the LEC to incorrectly predict the need for different tasks/settings.
Modified on	16/02/2023 11:24

Name	Self-adaptive Machine Learning Systems~ Research Challenges and Opportunities
Number of Coding References	7
Number of Codes Coding	2
Coverage	27.50%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	04/02/2022 21:53
Coded Text	Causes of Degradation of ML Components' Accuracy This section illustrates each type of data-shift with examples from the fraud detection systems domain. Covariate Shift. In a fraud detection system, covariate shift occurs when patterns of legitimate transactions change, for instance due to busy shopping days like Black Friday and Christmas [3]. Although the actual features used for classification may not change, their distribution does. For example, suppose a user usually purchases items online from shop A. The distribution of fraud given the feature online shop is 10% for shop A. The user then discovers that shop B actually sells the same items but at a cheaper price, so they start purchasing from shop B. In such a setting, the distribution of the feature online shop, given by P(x) is altered, but the distribution of fraud given the feature, P(y|x), remains the same, i.e., there is the same amount of fraud in shops A and B, regardless of where the user buys. This scenario will possibly lead the fraud detection system Self-adaptive Machine Learning Systems 143 to suspect that the change in the user's behavior is actually fraud because it learned that the user typically buys from shop A. As an example of incorrect data leading to shifts, security breaches could lead attackers to poison the data used for training ML models, hence causing them to make incorrect predictions. Prior Probability Shift (Label Shift). In the context of fraud detection systems, this type of shift occurs for example when the proportion of fraudulent transactions in the training set is different than for the test set [43], i.e., P(y)train ~= P(y)test. This type of shift requires assuming that the distribution of input data given fraud, P(x|y), does not change between training and testing environments. This corresponds to a mathematical abstraction over the power of an adversary that is capable of generating fraudulent transactions that follow the same pattern as legitimate transactions. Since the model has learned the typical distribution of fraud, it's predictions will follow that distribution, which is no longer representative of the system's environment. Concept Shift. This is the most common type of data-set shift in the fraud detection domain and occurs when fraudsters adapt their strategies and new fraud patterns emerge, such that the ML model is no longer able to effectively distinguish fraudulent from legitimate transactions [43]. In this case, it is the distribution P(y|x) that changes while P(x) remains the same.
Modified on	16/02/2023 11:23

Name	Self-adaptive Machine Learning Systems~ Research Challenges and Opportunities
Number of Coding References	7
Number of Codes Coding	2
Coverage	27.50%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	04/02/2022 21:53
Coded Text	Degradation and Repair of Learning-Enabled Components Similarly to traditional components, LECs can fail, leading systems to undesirable states and to require repair of the under-performing components [40]. Thus, in this section we are interested in understanding what can lead an LEC to fail or produce erroneous outputs (Sect. 3.1), and what tactics are available to repair the components that are deteriorating system utility (Sect. 3.2). The different causes of degradation and the applicability of the tactics introduced in Sect. 3.2 will be exemplified in Sects. 4 and 5 using two use-cases from the fraud detection systems' and cyber-physical systems' domains, respectively. 3.1 Causes of Degradation of ML Components' Accuracy ML approaches rely on a data-set composed of multi-dimensional input data and labels. Since this data-set is used for training the ML model, the environment from which these data-points are collected is usually known as the training environment. Then, once the ML model has been trained, it can be used by the system to make predictions in run-time. This is typically considered the testing environment as the model has never seen the current data-points. We will use the notions of training and testing environments throughout this section to introduce typical causes of degradation of ML components. It is generally assumed that the distribution p(y, x) of labels y and multidimensional input data x does not change between training and testing environments. However, when this assumption does not hold, and hence the prior distribution P(x), or the posterior distribution P(y), or any of the conditional distributions P(y|x) or P(x|y) changes, one may be in the presence of a problem commonly known as data-set shift [51,56–58]. This raises the question of whether the current model is still fit for the current environment. As recent work has shown, not all data-set shifts are malign [58]. Thus, an effective SAS should not only detect shifts, but also assess their actual impact on system utility. The literature on ML has investigated several types of data-set shifts that have different characteristics. These different characteristics influence the impact that each type of shift has on a given system, and also how easy it is to deal 138 M. Casimiro et al. with/detect the shift. Specifically, problems such as anomaly detection, novelty detection, open set recognition, out of distribution detection, and outlier detection [70] are specific instances of the most common types of shift. We argue that the different types of shift are general enough to be representative of most of the issues addressed by the existing ML literature. The following paragraphs introduce these types of shift and give examples of typical sources of shift that can affect an LEC. Covariate Shift. When the distribution of the inputs to a model changes, such that it becomes substantially different from the distribution on which the model was trained, we find ourselves in the presence of a problem commonly known as covariate shift. That is, the distribution P(x) changes but the conditional distribution P(y|x) remains the same. More formally, P (x)train ~= P(x)test and P(y|x)train = P(y|x)test [48]. This type of shift is usually analyzed to evaluate how a model generalizes and how robust it is when the feature space is altered at test time, i.e. while the system is executing. Prior Probability Shift (Label Shift). Differently, when we are in the presence of prior probability shift, also known as label shift, this means that the distribution P(y) of the labels/outputs has changed, i.e., the class proportions differ between training and test. More formally, P(y)train ~= P(y)test and P(x|y)train = P(x|y)test [48]. This can be seen as the inverse of covariate shift in the sense that now the distribution of features is the same between training and testing while the distribution of the labels changes. Dealing with this type of shift is particularly challenging when the new distribution P(y)test is unknown. Concept Shift. Finally, concept shift corresponds to a change in the relationship between input and output distributions, although each remains the same. More formally, when this type of shift occurs, we can have P(y|x)train ~= P(y|x)test and P(x)train = P(x)test or P(x|y)train ~= P(x|y)test and P(y)train = P(y)test [48]. Although these are the most common types of shift, it is also possible that other types of shift occur. We list them for completeness but give examples only of the most common ones in the remainder of the paper. Other types of

shift that can happen are for instance when both the conditional distribution and the features/labels distribution changes. Formally, this would correspond to P(y|x)train ~= P(y|x)test and P(x)train ~= P(x)test or P(x|y)train ~= P(x|y)test and P(y)train ~= P(y)test [48]. These types of shift are typically less investigated in the literature since they are not so common in real world applications and also because they are extremely difficult to detect and deal with.

Sources of Data Shift. During the normal operation of a system, shift in the data can occur due to several reasons: to the passing of time, incorrect data or sample selection bias. Next, we provide details on each of these sources of shift.

Self-adaptive Machine Learning Systems 139

Natural Drift due to Time. An effect of the natural passing of time is that people's tastes and behavior patterns change [7,43]. For example, due to the passing of time and due to inflation, the value of money decreases. A static LEC, that is never adapted and does not account for these natural changes will gradually start producing worse predictions.

Incorrect Data. This problem arises when there are samples in the model's training set that are incorrectly labeled [66] or when test data is tampered with, thus leading the model to mispredict for inputs with specific characteristics. The former can happen for instance when unsupervised techniques are used to label examples in order to bootstrap the training set of a second supervised model [66]. Incorrect data can also make their way into a model's training set due to attackers that intentionally pollute it (e.g., by maliciously altering some of the input features) so as to cause the ML component to incorrectly predict outputs for certain inputs [33,35]. Finally, noise and uncertainty, due to sensor errors or due to errors from upstream components, may also change the input data to an LEC, possibly causing drift and mispredictions.

Sample Selection Bias. This occurs when selecting data-points for a training set or when performing data cleaning1. While selecting data-points, there may be environmental factors that cause some inputs or labels to be sampled more often. For example, when selecting participants for a survey, steps must be taken to ensure that the population of interest is accurately represented. Similarly, when performing data cleaning, for example for a digit recognition task, less clear digits may be thrown away. However, this may prevent the model from learning that some digits are intrinsically harder to write than others [57]. During these, arguably critical, phases of model construction, sample selection bias will cause the training distribution to follow a different distribution than the test distribution, hence causing data shift and potentially a drop in system utility.

3.2 Repair Tactics

Table 1 illustrates a collection of tactics that can be used to deal with issues introduced by ML-based components caused by the different types of shift previously introduced. These tactics were inspired by research on ML [12,46,52,61,67]. Next, we describe the tactics presented in the table, discussing their costs and benefits, and motivating them in the following sections with scenarios from enterprise systems (Sect. 4) and cyber-physical systems domains (Sect. 5).

Component Replacement. This tactic assumes the existence of a repository of components and respective meta-data that can be analyzed to determine if there exists a component that is better suited for the current system state, i.e., that is expected to lead to a higher system utility. If such a component exists, then this 1 In ML, data cleaning corresponds to the process of identifying and correcting errors in a dataset that may negatively impact a predictive model.

140 M. Casimiro et al.

Table 1. Examples of general adaptation tactics for ML-based systems with their strengths ('+') and weaknesses ('–'). Tactic

Description

Component replacement

Replace an under-performing component by one that better matches the current environment

Human-based labeling [46]

Rely on a human to classify incoming samples or to correct the labeling of samples in the training set

Transfer learning [52]

Reuse knowledge gathered previously on different tasks/problems to accelerate the learning of new tasks

Unlearning [12] Remove samples that are no longer representative from the training set and from the model

Retrain [67]

Retrain with new data and maybe choose new values for the ML model's hyper-parameters

Properties

+ Fast and inexpensive, when possible – Alternative components may not be available in all scenarios – Alternative estimators, when available, may be more robust but less precise

+ Accuracy of human-based labels

expected to be high – Expert knowledge may be expensive to obtain and/or introduce unacceptable latency

+ Less data-hungry than plain retrain – Effectiveness dependent on the similarities between old and new tasks/data – Computationally intensive process

	+ Fast when ratio between data to forget and data-set size is small – Cost/latency for identifying examples to unlearn can be large and context-dependent + Generic and robust method – Computationally intensive process – Accuracy and latency of the retrain process may vary significantly – Effective only once a relatively large number of instances of the new data are available tactic will replace the under-performing component by the one that is expected to be better. A benefit of this tactic, whenever it is available, is to enable a swift reaction to data-set shifts. Its main cost depends on the latency and resources used for the analysis of the candidate components available in the repository. Additionally, alternative components may not always be available and, when they do exist, they may be less precise albeit more robust. Human-Based Labeling. Humans are often able to recognize patterns, problems, and objects more accurately than ML components [46]. Thus, depending on the domain, humans may play a role in correcting these components or giving them correct samples [46,65]. For example, when an LEC is highly uncertain about a specific input, it may rely on a human to provide a label. Similarly, if incorrect data is found on a model's data-set, a human may be asked to correct those samples. While this tactic may provide high benefit if the human is an expert, it also has a high cost, since humans are expensive. Also, if there is a significant amount of samples to label, the latency of the process may be unacceptable. Transfer Learning. Transfer learning (TL) techniques leverage knowledge obtained when performing previous tasks that are similar to the current one so that learning the current task becomes easier [42,52]. For this tactic to be Self-adaptive Machine Learning Systems 141 applicable, it is necessary to evaluate the similarity between the source and target tasks/domains. Transferring knowledge between dissimilar tasks will not provide benefits. In order to compute this similarity, metrics such as LEEP [49] can be used. The advantages of this tactic are that it requires less data than is needed to retrain a model from scratch, thus allowing for a quicker model initialization phase. However, the process of TL, similarly to a model retrain, is also computationally intensive. Unlearning. This tactic corresponds to unlearning data that no longer reflects the current environment/state of the system and its lineage, thus eliminating the effect of that data on current predictions [12], while avoiding a full model retrain. A key problem that stands in the way of the execution of this tactic is the identification of incorrect labels. In scenarios in which the identification of incorrect samples is not readily available, one may leverage automatic techniques, such as the one described in [13], which are faster but typically less accurate than relying on humans. As such, the cost and complexity of this adaptation tactic vary depending on the context. Then, after identifying the incorrect samples, the model must be updated to accurately reflect the correct data. The advantage of unlearning techniques with respect to a typical full model retrain is the time savings (up to several orders of magnitude [12]) that can be achieved. Retrain and/or Hyper-parameter Optimization. This is a general tactic that involves retraining the model with new data that reflects recent relevant data-set shifts. There are many types of retraining, ranging from a simple model refresh (incorporate new data using old hyper-parameters), to a full retrain (including hyper-parameter optimization, possibly encompassing the search for different model types/architectures [26]). These imply different computational costs and lead to different benefits in terms of model accuracy improvements. In the presence of data-set shifts, when there is new data that already incorporates the new input distribution, this tactic often represents a simple, yet possibly expensive, approach to deal with this problem. However, this tactic usually requires a substantial amount of data to yield highly accurate models and is computationally intensive. The benefits of this tactic are dependent on the type of retrain process and on the quality of the new data. As for its cost, if retraining is performed on the cloud, it can be directly converted to the economic cost of the virtual machines. Several techniques exist to predict such costs [2,16,45,69].
Modified on	16/02/2023 11:22

Name	Self-adaptive Machine Learning Systems~ Research Challenges and Opportunities Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	SHMEM-ML~ Leveraging OpenSHMEM and Apache Arrow for Scalable, Composable Machine Learning
Number of Coding References	3
Number of Codes Coding	2
Coverage	10.57%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 23:01
Coded Text	Data science and machine learning techniques have found broad applications, from proxy modeling in scientific applications to consumer recommendation engines to autonomous vehicles. Most DS/ML frameworks are written to maximize programmability and portability, sacrificing performance. For example, most are written for Python, an extremely flexible but also interpreted programming language with high overheads. Pandas, a popular Python library for data scientists, mostly follows a copy-on-write semantic for mutating large n-dimensional arrays. This can lead to massive memory consumption on moderately-sized datasets. While this tradeoff makes sense for small-scale projects, this causes problems for even simple data processing, exploration, and visualization workflows on the large-scale datasets that are common place today. As a result, several efforts have explored taking well-known techniques and frameworks from the HPC community and applying them to DS/ML frameworks to yield both productive and high performance domain specific libraries/languages. These past works generally fall in to two buckets: (1) efforts to transparently use HPC frameworks underneath existing, industry-standard DS/ML frameworks, or (2) effort to replace existing DS/ML frameworks with new ones built with HPC technologies from the start. 1.1 Related Work: Using HPC Frameworks Under Existing DS/ML Frameworks For example, in [11] the authors used OpenSHMEM [7] to accelerate distributed Caffe training jobs of the LeNet Solver network by replacing the existing MPIbased gradient exchange with equivalent OpenSHMEM operations. While this yielded a 30% improvement in training time over the existing implementation, this application of HPC technologies ignores the rest of the data science workflow. Projects like this one focus on a relatively small segment of the data science workflow (in this case, model gradient updates). Additionally, given that these optimizations are generally done at the lowest level of the data science software stack, they may miss optimizations that are only possible when higher level semantics are exposed.
Modified on	16/02/2023 10:25

Name	SHMEM-ML~ Leveraging OpenSHMEM and Apache Arrow for Scalable, Composable Machine Learning
Number of Coding References	3
Number of Codes Coding	2
Coverage	10.57%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	SHMEM-ML is a domain specific library for distributed array computations and machine learning model training & inference. Like other projects at the intersection of machine learning and HPC (e.g. dask, Arkouda, Legate Numpy), SHMEM-ML aims to leverage the performance of the HPC software stack to accelerate machine learning workflows. However, it differs in a number of ways. First, SHMEM-ML targets the full machine learning workflow, not just model training. It supports a general purpose nd-array abstraction commonly used in Python machine learning applications, and efficiently distributes transformation and manipulation of this ndarray across the full system. Second, SHMEM-ML uses OpenSHMEM as its underlying communication layer, enabling high performance networking across hundreds or thousands of distributed processes. While most past work in high performance machine learning has leveraged HPC message passing communication models as a way to efficiently exchange model gradient updates, SHMEM-ML's focus on the full machine learning lifecycle means that a more flexible and adaptable communication model is needed to support both fine and coarse grain communication. Third, SHMEM-ML works to interoperate with the broader Python machine learning software ecosystem. While some frameworks aim to rebuild that ecosystem from scratch on top of the HPC software stack, SHMEM-ML is built on top of Apache Arrow, an in-memory standard for data formatting and data exchange between libraries. This enables SHMEM-ML to share data with other libraries without creating copies of data. This paper describes the design, implementation, and evaluation of SHMEMML – demonstrating a general purpose system for data transformation and manipulation while achieving up to a 38× speedup in distributed training performance relative to the industry standard Horovod framework without a regression in model metrics.
Modified on	16/02/2023 10:24

Name	SHMEM-ML~ Leveraging OpenSHMEM and Apache Arrow for Scalable, Composable Machine Learning
Number of Coding References	3
Number of Codes Coding	2
Coverage	10.57%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	04/02/2022 22:44
Coded Text	SHMEM-ML is built on top of a number of open source or third party software packages. This section offers a brief overview of the fundamental building blocks of SHMEM-ML, as well as how they are put together to support distributed arrays and integration with the broader Python ecosystem. At a high level, SHMEM-ML uses: – Apache Arrow [1] for in-memory data storage and zero-copy data exchange with third party Python libraries. – OpenSHMEM [7] for distributed job creation, inter-process communication, and inter-process coordination. – Tensorflow, Keras, scikit-learn, numpy and other Python data science libraries for the implementation of more algorithmically complex data science functionality such as training and inference of deep neural networks. 3.1 Background: OpenSHMEM The OpenSHMEM library provides a single program, multiple data (SPMD) execution model in which N instances of the program are executed in parallel. Each instance is referred to as a processing element (PE) and is identified by its integer ID in the range from 0 to N − 1. PEs exchange information through one-sided get (read) and put (write) operations that access remotely accessible symmetric objects. Symmetric objects are objects that are present at all PEs and they are referenced using the local address to the given object. By default, all objects within the data segment of the application are exposed as symmetric; additional symmetric objects are allocated through OpenSHMEM API routines. OpenSHMEM's communication model is unordered by default. Point-to-point ordering is established through fence operations, remote completion is established through quiet operations, and global ordering is established through barrier operations. 3.2 Background: Apache Arrow Apache Arrow is an open community effort to define a universal in-memory data format for n-dimensional arrays. Arrow's aim is to enable zero-copy, efficient data exchange between different libraries regardless of language and without each library having to provide explicit support for every other library. Apache Arrow defines a number of commonly used objects and functionalities, including one-dimensional arrays, two-dimensional tables, and file I/O. Title Suppressed Due to Excessive Length 3.3 Background: scikit-learn, Tensorflow, and Horovod scikit-learn and Tensorflow/Keras are industry standard libraries for training and applying data-driven or machine learned models. scikit-learn focuses on providing classes for more classical and statistically-derived machine learning models (e.g. linear regressors, support vector machines, random forests, gaussian mixtures). Tensorflow/Keras focus on more recent developments in deep learning models, making it simple and straightforward to create deeply layered models with a variety of built-in layer types supported (e.g. Dense, Convolutional, Pooling, Recurrent, Normalization). Custom layer types can also be added by programmers. While Keras was started as an independent framework for training deep learning models, it was eventually merged into Tensorflow in 2017 as an alternative API. While scikit-learn does not support distributed training today, Tensorflow/Keras offer a number of options. The most commonly used framework for distributed training of Keras models is Horovod [9] which uses an efficient ring-allreduce method to distributed gradient updates while sitting on top of high performance communication libraries (e.g. MPI) when supported. The introduction of Horovod to the Tensorflow/Keras communities drastically improved the scalability and productivity of distributed training.
Modified on	16/02/2023 10:25

Name	SHMEM-ML~ Leveraging OpenSHMEM and Apache Arrow for Scalable, Composable Machine Learning Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Shuffler~ A Large Scale Data Management Tool for Machine Learning in Computer Vision
Number of Coding References	13
Number of Codes Coding	3
Coverage	14.95%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	04/02/2022 13:03
Coded Text	
Modified on	04/02/2022 13:03

Name	Shuffler~ A Large Scale Data Management Tool for Machine Learning in Computer Vision
Number of Coding References	13
Number of Codes Coding	3
Coverage	14.95%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	04/02/2022 13:03
Coded Text	
Modified on	04/02/2022 13:14

Name	Shuffler~ A Large Scale Data Management Tool for Machine Learning in Computer Vision
Number of Coding References	13
Number of Codes Coding	3
Coverage	14.95%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	04/02/2022 13:03

Coded Text	as commercial Supervisely4, Playmate5, and Labelbox6. These systems offer sophisticated tools for human annotators to label images in order to prepare training data for different types of Machine Learning tasks. Though our proposed toolbox, Shuffler, offers basic functionality for image labelling, its primary focus is processing the output of such image annotation systems. Second, an important part of a Machine Learning pipeline is loading and augmenting image data. Numerous libraries, including a library from NVIDIA, DALI7, have been proposed for this task. In Figure 2, we refer to this part of the pipeline as step 3. In turn, Shuffler is employed on step 2 to prepare a dataset of training data that will be further loaded and augmented during training. Next, end-to-end product life-cycle management systems have been proposed, such as ModelHub [13], and commercial Allegro8. These systems focus on Machine Learning model management, while the goal of Shuffler is to provide instruments to manage training data.
Modified on	14/02/2022 15:51

Name	Shuffler~ A Large Scale Data Management Tool for Machine Learning in Computer Vision
Number of Coding References	13
Number of Codes Coding	3
Coverage	14.95%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	04/02/2022 13:03
Coded Text	First, some systems are designed specifically for annotating data for Computer Vision applications. Examples include publicly available LabelMe [17], VGG Image Annotator [7], and CVAT3, as well
Modified on	14/02/2022 15:49

Name	Shuffler~ A Large Scale Data Management Tool for Machine Learning in Computer Vision
Number of Coding References	13
Number of Codes Coding	3
Coverage	14.95%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	04/02/2022 13:03

Coded Text	In this work, we close this gap by proposing a software tool- box, Shuffler, designed specifically for manipulating annotations. It employs widely known relational databases and the associated SQL query language for storing and manipulating annotations. The proposed toolbox is heavily based on SQL and allows to chain multiple operations in a single command. Annotations are stored in an relational database (Sqlite, MySql, ...) with schema designed to cover the bulk of the common tasks in computer vision. The proposed solution satisfies the following properties: • it has basic manipulation tools and allows to easily add new functions; • annotations are fast to load and to modify and convenient to store; • annotations are stored in a human readable format that can be manually edited; • it is agnostic to the format of how images are stored on disk; • it supports image-level classification, object detection, semantic segmentation, and object matching tasks in computer vision.
Modified on	04/02/2022 13:10

Name	Shuffler~ A Large Scale Data Management Tool for Machine Learning in Computer Vision
Number of Coding References	13
Number of Codes Coding	3
Coverage	14.95%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	04/02/2022 13:03
Coded Text	In this work, we present Shuffler, an open source tool that makes it easy to manage large computer vision datasets. It stores annotations in a relational, human-readable database. Shuffler defines over 40 data handling operations with annotations that are commonly useful in supervised learning applied to computer vision and supports some of the most well-known computer vision datasets. Finally, it is easily extensible, making the addition of new operations and datasets a task that is fast and easy to accomplish.
Modified on	04/02/2022 13:07

Name	Shuffler~ A Large Scale Data Management Tool for Machine Learning in Computer Vision
Number of Coding References	13
Number of Codes Coding	3
Coverage	14.95%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	04/02/2022 13:05
Coded Text	Datasets in the computer vision academic research community are primarily static. Once a dataset is accepted as a benchmark for a computer vision task, researchers working on this task will not alter it in order to make their results reproducible. At the same time, when exploring new tasks and new applications, datasets tend to be an ever changing entity. A practitioner may combine existing public datasets, filter images or objects in them, change annotations or add new ones to fit a task at hand, visualize sample images, or perhaps output statistics in the form of text or plots.
Modified on	04/02/2022 13:05

Name	Shuffler~ A Large Scale Data Management Tool for Machine Learning in Computer Vision
Number of Coding References	13
Number of Codes Coding	3
Coverage	14.95%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	04/02/2022 13:05
Coded Text	Datasets typically are in a custom format, which usually includes an image and an annotation file in one of the following formats: xml, txt, or json. Table 1 presents an overview of several popular object detection datasets in the area of computer vision and the formats of the associated annotation files. On the one hand, these formats are human-readable, but on the other hand, quite slow to load. Additionally, changing annotations and saving them as a copy means duplicating the whole directory with the annotation files, which is inconvenient and slow. Many development kits cache annotations by serializing them with formats such as pickle1 or protobuf2. Such formats are easy to load by a machine
Modified on	04/02/2022 13:10

Name	Shuffler~ A Large Scale Data Management Tool for Machine Learning in Computer Vision
Number of Coding References	13
Number of Codes Coding	3
Coverage	14.95%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	04/02/2022 13:05
Coded Text	Given that ML and deep learning call for large volumes of data to produce satisfactory results, it is no surprise that the resulting data and software management associated to dealing with live datasets can be quite complex. As far as we know, there is no flexible, publicly available instrument to facilitate manipulating image data and their annotations throughout a ML pipeline.
Modified on	04/02/2022 13:06

Name	Shuffler~ A Large Scale Data Management Tool for Machine Learning in Computer Vision
Number of Coding References	13
Number of Codes Coding	3
Coverage	14.95%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	04/02/2022 13:05
Coded Text	In the computer vision academic community, day-to-day work emphasizes primarily algorithms rather than data. From this point of view, annotated image datasets are ideally built once and remain fixed. This approach allows the community to use datasets as benchmarks. Researchers choose to store these datasets in formats that are most common and fast to load for machine learning (ML) packages. In contrast, for a data scientist in industry, the task is not necessarily to improve an algorithm, but rather to try different algorithms and tasks on various partitions and modifications of the same dataset. In this case, a dataset is not considered static, but rather constantly altered to fit the task at hand. In turn, multiple versions of the same dataset need to co-exist in a centralized or a distributed storage system. Ideally, a practitioner would want 1) a simple way to manipulate image data and its annotations, and 2) a file format that allows to store multiple copies of the annotation set in an organized and efficient way and to inspect them manually. Data manipulation tools are sometimes packaged with a dataset, but they typically allow to perform only a limited number of operations only on that particular dataset and often for a single programming language.
Modified on	04/02/2022 13:09

Name	Shuffler~ A Large Scale Data Management Tool for Machine Learning in Computer Vision
Number of Coding References	13
Number of Codes Coding	3
Coverage	14.95%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	04/02/2022 13:05
Coded Text	To sum up, we consider (Figure 2) a typical data preparation workflow of a computer vision practitioner to consist of three steps: 1) download or collect a dataset, 2) modify annotations, and 3) serialize the dataset. We further consider a common situation when multiple modifications of annotations are used. Modifications could be a chain of trivial tasks, for example, removing objects at image boundaries and then increasing the size ofbounding boxes. We note 1) the lack ofsoftware for manipulating image data and annotations, and 2) a convenient format to store annotations.
Modified on	04/02/2022 13:11

Name	Shuffler~ A Large Scale Data Management Tool for Machine Learning in Computer Vision
Number of Coding References	13
Number of Codes Coding	3
Coverage	14.95%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Research gap in Maintaining ML systems
Created on	14/02/2022 15:40
Coded Text	As far as we know, there is no flexible, publicly available instrument to facilitate manipulating image data and their annotations throughout a ML pipeline
Modified on	14/02/2022 15:40

Name	Shuffler~ A Large Scale Data Management Tool for Machine Learning in Computer Vision
Number of Coding References	13
Number of Codes Coding	3
Coverage	14.95%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Research gap in Maintaining ML systems
Created on	14/02/2022 15:40
Coded Text	We note 1) the lack ofsoftware for manipulating image data and annotations, and 2) a convenient format to store annotations.
Modified on	14/02/2022 15:46

Name	Software Architecture Challenges for ML Systems
Number of Coding References	7
Number of Codes Coding	3
Coverage	16.30%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	Developing machine learning (ML) systems, just like any other system, requires architecture thinking. However, there are characteristics of ML components that create challenges and unique quality attribute (QA) concerns for software architecture and design activities, such as data-dependent behavior, detecting and responding to drift over time, and timely capture of ground truth to inform retraining. This paper presents four categories of software architecture challenges that need to be addressed to support ML system development, maintenance and evolution: software architecture practices for ML systems, architecture patterns and tactics for ML-important QAs, monitorability as a driving QA, and co-architecting and co-versioning. These challenges were collected from targeted workshops, practitioner interviews, and industry engagements. The goal of our work is to encourage further research in these areas and use the information presented in this paper to guide the development of empiricallyvalidated practices for architecting ML systems
Modified on	22/07/2022 10:35

Name	Software Architecture Challenges for ML Systems
Number of Coding References	7
Number of Codes Coding	3
Coverage	16.30%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	B. Architecture Patterns and Tactics for ML-Important QAs Quality attributes (QAs) — properties used to evaluate the quality and fitness of a system to meet its business goals — drive the selection of architecture approaches, includ- ing patterns and tactics, and consequently the structure and behavior of software systems [2]. While organization- and domain-specific business goals shape architectural and other system requirements, ML-system-specific QA concerns also play an important role in ML systems. These attributes include explainability, data centricity, verifiability, monitorability, observability, and fault tolerance, at a minimum, in addition to elevated importance of security and privacy [19]. Analysis techniques to assure their correct design and implementation will need to be developed. These attributes will also drive the development of architecture-level techniques for addressing fairness, unintended bias, and ethics, in particular to limit propagation of unintended consequences.
Modified on	24/02/2022 14:36

Name	Software Architecture Challenges for ML Systems
Number of Coding References	7
Number of Codes Coding	3
Coverage	16.30%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	D. Co-Architecting and Co-Versioning A ML system has two software architectures that need to be developed and sustained: (1) the architecture of the ML system as described in Figure 2 (the system that uses the ML components), and (2) the architecture of the system that supports the ML model life cycle shown in Figure 1 (the system that produces the ML model, often called the model development pipeline). This latter architecture is often neglected as models are developed in trial-and-error, experimental mode, often by people whom are not trained in software engineering [10]. We use the term co-architecting to refer to the fact that both architectures need to be developed in sync, such that design decisions are driven by both system and model requirements, as well as the perspectives of the different stakeholders and development teams. An architectural approach for the model development pipeline also promotes potential for reuse, especially for data pipeline components (i.e., components that extract and transform raw data into training data). Successful co-architecting requires additional work in three areas: (1) practices that enable synchronization and integration between the two architectures, (2) architecture representations for ML-relevant concerns (e.g., data quality, model accuracy), as well as ML-relevant components (e.g., data pipelines, model elements), and (3) architecture views for model development pipelines that reflect and communicate design decisions and concerns related to data and feature engineering (e.g., data distribution, algorithm selection, feature selection). Work in these areas needs to consider effective communication, simple representation, and visualization tools because co-architecting will likely happen in teams that combine data scientists, software engineers, and potentially other disciplines (i.e., domain experts).
Modified on	24/02/2022 14:37

Name	Software Architecture Challenges for ML Systems
Number of Coding References	7
Number of Codes Coding	3
Coverage	16.30%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	operations perspectives that occur when these groups work independently [10]. III. CHALLENGES FOR ARCHITECTING ML SYSTEMS We present four categories of software architecture chal- lenges that need to be addressed for the process depicted in Figure 2 to support ML system development, as well as their maintenance and evolution. A. Software Architecture Practices for ML Systems Existing established software architecture practices to sup- port design, development, and deployment of software systems [2], in addition to data-intensive system paradigms (e.g., big data analytics systems [18]), provide a foundation for architecting ML systems. A ML component can be considered a software component with characteristics that are not common in traditional software components. The behavior of a traditional software component is defined by rules programmed in code that address its QA requirements and expected response measures. However, the behavior of a ML component is defined by characteristics of the datasets it is trained with, in addition to the system's QA concerns [24]. Therefore, software architecture practices will need to take into account how to address requirements specification, design specification, and interpretability concerns driven by datasets [9]. Software development processes, including agile software development processes, went through an alignment stage to incorporate architecture tasks effectively. A similar adjustment will be needed to align the experimental, iterative and incremental nature that is inherent in architecting and development of ML models and ML components [1] [21]. Although continuous evolution and iterative development are not new to software architecting, the uncertainty introduced by the volatility of the data that drives ML component development is certainly not common. ML component development relies on generate-and-test approaches which make them hard to align with sprint boundaries and identification of "done criteria" common in most software development processes. In summary, while many existing software architecture practices and design paradigms are applicable to ML systems, some will need to be adapted to account for data-dependent behavior of ML components. New practices will need to be developed to account for ML-important QAs (next section). To note is that any existing, adapted, or new software architecture practice will need to take into account perspectives of new sets of stakeholders [6] [12] [17], including data curators, data
Modified on	24/02/2022 14:36

Name	Software Architecture Challenges for ML Systems
Number of Coding References	7
Number of Codes Coding	3
Coverage	16.30%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	The software architecture of a system is the collection of structures that depict the behavior of the system and inform how well the system meets its business and quality goals, including how well the longevity of the system is supported from a maintenance and evolution perspective. Development, deployment, maintenance, and evolution of systems that include ML components pose different architecting challenges. In this paper, we summarized these challenges collected from researchers and practitioners through workshops, interviews, and industry engagements. Research needs to question existing software architecture concepts and practices for their fitness to support ML systems. The challenges include understanding how well existing software architecture practices support ML system development, as well as developing patterns and tactics to respond to ML-important QAs. In addition, there are architectural
Modified on	24/02/2022 14:37
Name	Software Architecture Challenges for ML Systems
Number of Coding References	7
Number of Codes Coding	3
Coverage	16.30%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	Understanding monitorability of ML systems requires ad- ditional work in several areas. First, we need to understand what different monitoring techniques will be needed for data quality vs. model quality vs. software quality vs. service quality. Existing patterns and tactics for monitorability and observability will apply for some, but new ones will need to be developed as well. Second, there are opportunities to relate monitorability to self-adaptation [17]: (1) of ML:ML models self-adapt to system changes (one of the goals of MLOps), (2) for ML: ML system adapts to changes in the system that affect quality of service (QoS), and (3) by ML: system uses ML to adapt. And lastly, we need to understand
Modified on	24/02/2022 14:36

Name	Software Architecture Challenges for ML Systems
Number of Coding References	7
Number of Codes Coding	3
Coverage	16.30%
Folder Location	Codes\\Snowballing
Parent Name	Codes\\Snowballing\\Snowballing
Created on	01/06/2022 13:19
Coded Text	There is limited synthesized knowledge on software engineering (SE) methods for development, maintenance, and evolution of ML systems. Several recent literature reviews analyze the state-of-the-art of SE for ML systems. A systematic literature review was performed by Kumeno [8] that identified SE challenges for ML systems and mapped them to the SWEBOK body of knowledge areas. A more recent systematic literature review by Mart´ınez-Fern´andez et al. [13] looked at studies published between 2010 and 2020 and identified that more than 2/3 of the studies were published in the last three years, which is not surprising given that this is an emerging field. In the context of large-scale ML systems, Lwakatare et al. [12] conducted a similar review of SE challenges faced by practitioners. The challenges identified were structured according to different quality attributes (adaptability, scalability, privacy, and safety) and stages of ML pipeline development (data acquisition, training, evaluation, and deployment). There are fewer studies on software architecture practices for ML systems. Four key areas of software architecture and their challenges for architecting ML systems were highlighted by Muccini et al. [17], which include architecture framework, architecting process, self-adaptive architecting, and architecture evolution. An earlier survey by Masuda et al. [14] analyzed ML systems from a software quality perspective, summarizing software quality methods and techniques for ML systems. The suitability of using the SQuaRE software quality model for ML systems was analyzed in work by Kuwajima et al. [9], which provided insights into some of the major differences between ML systems and conventional software systems due to the characteristics of ML components. Finally, patterns and anti-patterns for architecting ML systems were identified through a multi-vocal literature review conducted by Washizaki et al. [25]. These identified patterns were further
Modified on	22/07/2022 11:12

Name	Software design patterns for ai-systems
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Software Engineering for AI-Based Systems~ A Survey
Number of Coding References	22
Number of Codes Coding	11
Coverage	4.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:22
Coded Text	Software requirements (17 studies) For software requirements, we identified 17 studies. Based on our thematic analysis, we derived five subcategories, with most papers belonging to several categories. Nine papers address the Requirements Engineering (RE) process for AI-based systems. Vogelsang and Borg are the only ones to cover the complete process, from elicitation to verification and validation [205]. They summarized important characteristics of the RE process for ML systems, such as detecting
Modified on	03/02/2022 16:01

Name	Software Engineering for AI-Based Systems~ A Survey
Number of Coding References	22
Number of Codes Coding	11
Coverage	4.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:22
Coded Text	Software requirements Most of the challenges relate to one activity in the requirements engineering cycle, except a first subset related to fundamentals: • Fundamentals. A number of challenges relate to functional and non-functional requirements fundamentals, arguing that: (i) our understanding of NFRs for ML is fragmented and incomplete, (ii) some new NFR types need to be considered, e.g. related to explainability and freedom for discrimination, (iii) measurement of functional requirements in practice for both functional and non-functional requirements is needed. • Elicitation. The use of data as a source of requirements is attractive, but due to the high volume of such data, it requires tool support to detect features from massive
Modified on	03/02/2022 16:03

Name	Software Engineering for AI-Based Systems~ A Survey
Number of Coding References	22
Number of Codes Coding	11
Coverage	4.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:23
Coded Text	Software design (34 studies) In the area of software design, we formed seven categories to group the 34 studies. One of the largest of these – design for X – is concerned with design approaches or techniques to improve or assure one specific quality attribute in AI-based systems. It comprises 11 papers, of which the majority is concerned with safety (seven papers). Most of these studies propose design strategies for AI systems in safety-critical domains where unsafe behaviour can have immense negative consequences, like the use of architectural components as a protection layer for safety in autonomous vehicles [139], design best practices for safety verification methods [16] or behavior-bounded assurance [174] for autonomous aerial vehicles, safety design strategies (e.g., inherently safe design, safety reserves, safe fail, and procedural safeguards) for ML components [118, 202], and specific safety strategies for cyber-physical systems [7, 203]. The remaining four design approaches are for reliability [130], user experience [222], and for quality
Modified on	03/02/2022 16:02

Name	Software Engineering for AI-Based Systems~ A Survey
Number of Coding References	22
Number of Codes Coding	11
Coverage	4.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:23
Coded Text	Software design Software design challenges occur at different levels: • Design principles. It is a challenge to overcome the CACE principle (Changing Anything Changes Everything), which is mainly due to the entanglement created by ML models. • Software structure and architecture. Several situations particular to AI-based systems result in challenges to their structure: (i) ethics is a main issue of these applications, and a challenge is where to place the logic that governs ethical behaviour responding to criteria like scalability (needed with the advent of technologies such as 5G that demand huge amounts of sensors); (ii) AI-based systems need to deal with undeclared customers, who need to consume predictions ofmodels; (iii) it becomes necessary to orchestrate different systems that are glued together in real-world deployments; (iv) it is requested to provide automatic exposure ofML metadata to automate and accelerate model lifecycle management; (v) it is needed to manage the consequences of the concurrent processing required by most AI-based systems, especially
Modified on	03/02/2022 16:03

Name	Software Engineering for AI-Based Systems~ A Survey
Number of Coding References	22
Number of Codes Coding	11
Coverage	4.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:23
Coded Text	Software construction Contrary to the two former Knowledge Areas, challenges reported for software construction are of very diverse nature, most of them belonging to the Practical Considerations Topic: • Implementation ofML algorithms involves several issues, among them strong dependency on data, high level of parallelism or use of complex tools like TensorFlow. • End-to-end AI-based systems often comprise components written in different programming languages, making the management of applications challenging (e.g., ensuring consistency with error-checking tools across different languages). • Implementation of AI-based systems
Modified on	03/02/2022 16:03

Name	Software Engineering for AI-Based Systems~ A Survey
Number of Coding References	22
Number of Codes Coding	11
Coverage	4.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:23
Coded Text	Software construction (23 studies) A total of 23 studies were concerned with software construction. Many of these studies either provided specialized tools (six papers) or more holistic platforms (four papers) to support the development of AI systems. Exemplary application contexts for tools were the deployment and serving of general prediction systems [42], lowering the barrier for using ML techniques [14, 99, 100, 155], or model-based development toolchains [76]. For platforms, the common goal was to provide a more comprehensive infrastructure to improve and accelerate the AI development workflow [14, 40, 178], sometimes in more specialized domains like safety-critical robotics [47]. Furthermore, eight studies reported construction practices and guidelines for AI systems based on diverse experiences, such as implementing ML components to detect and correct transaction errors in SAP [162], a systematic comparison of DL frameworks and platforms [73], experiences from improving Airbnb search results
Modified on	03/02/2022 16:02

Name	Software Engineering for AI-Based Systems~ A Survey
Number of Coding References	22
Number of Codes Coding	11
Coverage	4.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:24
Coded Text	Software testing (115 studies) We identified 115 studies focused on software testing. During the analysis, we formed 11 subcategories (bugs &debugging, challenges, explanations, quality testing, test architecture & languages, test case, test case generation, test case selection, testing methods, testing metrics, and generic), which we partially refined into sub-themes. For each of the 115 studies, we assigned one or more of these subcategories. In three cases, we assigned five different subcategories to a paper. The different subcategories are described below: Four papers addressed bugs & debugging. Three of
Modified on	03/02/2022 16:02

Name	Software Engineering for AI-Based Systems~ A Survey
Number of Coding References	22
Number of Codes Coding	11
Coverage	4.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:24
Coded Text	Software testing We found some challenges directly related to Key Issues of software testing, majorly related to data and one also to process: • Data and models cannot be strongly specified a priori, which means that testing of AI-based systems is dependent upon uncertain data and models. • Achieving scalability of testing with
Modified on	03/02/2022 16:04

Name	Software Engineering for AI-Based Systems~ A Survey
Number of Coding References	22
Number of Codes Coding	11
Coverage	4.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:24
Coded Text	Software maintenance (6 studies) The small software maintenance area only comprises six studies, which we group further into three categories. Two studies empirically analyze the nature and prediction impact of bugs in AI software [122, 190]. Similarly, two studies are concerned with providing specialized approaches or tool support for the debugging ofML software by focusing on explanatory debugging in interactive ML [113], and proposing their Tensorflow debugger based on dataflow graphs [34]. The remaining two papers elicited and reported maintenance challenges
Modified on	03/02/2022 16:02

Name	Software Engineering for AI-Based Systems~ A Survey
Number of Coding References	22
Number of Codes Coding	11
Coverage	4.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	Software engineering process (31 studies) Many of the studies mapped to the SE process area deal with an AI/ML process (13 studies). Out of those, a few discuss processes in specific application areas, such as recommendation systems [123]. The remaining ones address processes in general. Although the majority of papers focuses on processes for developing AI-based systems, some papers (e.g. [187]) address the topic of processes for creating new ML algorithms and tools. Four papers [24, 80, 123, 173] present an overview of current practices and challenges faced during AI system development
Modified on	03/02/2022 16:02

Name	Software Engineering for AI-Based Systems~ A Survey
Number of Coding References	22
Number of Codes Coding	11
Coverage	4.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	Software engineering process Most of the challenges are related to the Software Life Cycle Topic, as follows: • Life cycle models. It is necessary to have highly iterative models that allow to evolve solutions quickly. Going further, there is
Modified on	03/02/2022 16:04

Name	Software Engineering for AI-Based Systems~ A Survey
Number of Coding References	22
Number of Codes Coding	11
Coverage	4.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:27
Coded Text	Software engineering models and methods (38 studies) From the 38 unique primary studies in this category, the majority formulate concrete proposals on models and methods, while a few also elaborate on challenges, practices, and roadmaps. In terms of topics, the papers lean a little more towards verification & validation (V&V) methods (24 papers) than models (12 papers, one of them also in the former V&V methods category), with three papers reporting generic challenges, practices, and roadmaps. In general, there was a strong dominance of safety as a non-functional aspect and application domains
Modified on	03/02/2022 16:02

Name	Software Engineering for AI-Based Systems~ A Survey
Number of Coding References	22
Number of Codes Coding	11
Coverage	4.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:27
Coded Text	Software Engineering Models and Methods The Modeling Topic has several challenges associated to its sub-topics: • Modeling Principles. It is necessary to avoid model overfit, which
Modified on	03/02/2022 16:04

Name	Software Engineering for AI-Based Systems~ A Survey
Number of Coding References	22
Number of Codes Coding	11
Coverage	4.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:27
Coded Text	Software quality (59 studies) Quality management is a very broad area of SE, including a number of topics such as specifying quality requirements, measuring / assessing quality, and assuring quality (SWEBOK). So far, quality management during engineering of AI-based systems has been dominated by testing and formal verification methods (see sections on testing and models and methods above). Only 17 publications address the topic of defining and assessing quality of AI-based systems. Among these, two papers [60, 179] discuss software technical debt, a derivative of software quality which refers most commonly to increased costs for maintenance and evolution due to earlier quality deficits. In particular, the authors warn software engineers tempted by quick wins of data-driven software systems of forgetting that these wins are not coming for free. To avoid incurring significant technical debt in terms of ongoing maintenance costs with AI systems, the authors explore technical debt-related risks, e.g., related to a software system itself as well as to associated data and data management systems.
Modified on	03/02/2022 16:03

Name	Software Engineering for AI-Based Systems~ A Survey
Number of Coding References	22
Number of Codes Coding	11
Coverage	4.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:27
Coded Text	Software Quality A first set of challenges are related to the Software Quality Management Process in its three subtopics: • Software Quality Assurance. Challenges are manifold: (i) define quality assurance standards for AI-based systems and ensure that they scale well (e.g., they should adapt well to the continuous evolution of ML models and thus system behaviour); (ii) establish quality assurance criteria in the presence of big data (as required by AI-based systems); (iii) dealing
Modified on	03/02/2022 16:04

Name	Software Engineering for AI-Based Systems~ A Survey
Number of Coding References	22
Number of Codes Coding	11
Coverage	4.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 16:04
Coded Text	Software Engineering Professional Practice We grouped the challenges related to professional practice into the following topics: • Interacting with Stakeholders. (i) We find challenges especially related with understanding of, and interaction with the customer because customers may have unrealistic expectations regarding the functionality, accuracy (requiring even 100% accuracy) or adoption process
Modified on	03/02/2022 16:04

Name	Software Engineering for AI-Based Systems~ A Survey
Number of Coding References	22
Number of Codes Coding	11
Coverage	4.22%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 16:05
Coded Text	Software Runtime Behaviour As explained in Section 3 and 7, we added this Knowledge Area that is not proposed in SWEBOK due to the importance of this aspect in AI-based systems. We further distinguished the following Topics: • Robustness and Operability. It refers to the fact that an AI-based system must be robust and easy to operate while in use. At this respect, mentioned challenges are: (i) the need to avoid negative side effects, so that the behaviour of the application does not damage
Modified on	03/02/2022 16:05
Name	Software Engineering for AI-Based Systems~ A Survey
Number of Coding References	22
Number of Codes Coding	11
Coverage	4.22%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56
Coded Text	Annual distribution of considered publications 100 120 20 40 60 80 0 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 Publications in DBLP Selected publications per year Fig. 2. Annual distribution of publications. 100000 200000 300000 400000 500000 0
Modified on	22/07/2022 14:56

Name	Software Engineering for AI-Based Systems~ A Survey
Number of Coding References	22
Number of Codes Coding	11
Coverage	4.22%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56
Coded Text	Fig. 4. Number of publications per author affiliation.
Modified on	22/07/2022 14:57

Name	Software Engineering for AI-Based Systems~ A Survey
Number of Coding References	22
Number of Codes Coding	11
Coverage	4.22%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56
Coded Text	Fig. 5. Distribution of publications by continent and country.
Modified on	22/07/2022 14:56

Name	Software Engineering for AI-Based Systems~ A Survey
Number of Coding References	22
Number of Codes Coding	11
Coverage	4.22%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56
Coded Text	Several secondary studies in the broad area of SE4AI have been published so far (see Table 1). Masuda et al. [134] conducted a review to identify techniques for the evaluation and improvement of the software quality ofML applications. They analyzed 101 papers and concluded that the field is still in an early state, especially for quality attributes other than functional correctness and safety. Washizaki et al. [211] conducted a multivocal review to identify architecture and design patterns for ML systems. From 35 resources (both white and grey literature), they extracted 33 unique patterns and mapped them to the different ML phases. They discovered that, for many phases, only very few or even no patterns have been conceptualized so far. Serban and Visser [181] performed both a case study and a systematic literature review on the topic of software architecture for machine learning. They reviewed 42 studies and performed 10 semi-structured interviews with practitioners from 10 different organisations. In their paper, the authors report 20 challenges and potential solutions. On a similar topic, John et al. [137], performed a systematic review of both scientific (13 studies) and grey literature (6 studies) on the topic of deployment ofML systems. They report a total of 27 challenges and 52 practices. Lorenzoni et al. [124] performed a systematic literature review on the topic of development of machine learning systems. They analysed 33 studies between 2010 and 2020 and classified 10 issues and 13 solutions into seven SE practices. A number of reviews have been conducted in the area of software testing. Borg et al. [23] performed a review of verification and validation techniques for deep neural networks (DNNs) in the automotive industry. From 64 papers, they extracted challenges and verified them with workshops and finally a questionnaire survey with 49 practitioners. They conclude, among other challenges, that a considerable gap exists between safety standards and nature of contemporary ML-based systems. Another study was published by Ben Braiek and Khomh [27]. In their review of testing practices for ML programs, they selected a total of 37 primary studies and extracted challenges, solutions, and gaps. The primary studies were assigned to the categories ofdetect errors in data (five papers), in ML models (19 papers), and in the training program (13 papers). Riccio et al. [166] extracted testing challenges from 70 primary studies and propose the following 3 ACM TOSEM, Accepted in ACM TOSEM Martínez-Fernández, et al. categories: realism of test input data (5 papers), adequacy of test criteria (12 papers), identification of behavioural boundaries (2 papers), scenario specification and design (3 papers), oracle (13 papers), faults and debugging (8 papers), regression testing (5 papers), online monitoring and validation (8 papers), cost of testing (10 papers), integration ofML models (2 papers), and data quality assessment (2 papers). Similarly, Zhang et al. [226] surveyed the literature on ML testing and selected 138 papers. From these, they summarized the tested quality attributes (e.g. correctness or fairness), the tested components (e.g. the data or the learning program), workflow aspects (e.g. test generation or evaluation), and application contexts (e.g. autonomous driving or machine translation). In addition to these specialized reviews, there are also some secondary studies more similar to ours, i.e. that have a general SE focus. Serban et al. [180] conducted a multivocal review with 21 relevant documents (both white and grey literature) to identify and analyze SE best practices for ML applications. They extracted 29 best practices and used a follow-up questionnaire survey with 313 software professionals to find out the degree of adoption and impact of these practices. Furthermore, Wang et al. [208] took a broader view and conducted a systematic literature review about general synergies between ML/DL and SE, i.e. covering both machine learning for SE (ML4SE) and SE for machine learning (SE4ML) research. However, only 15 of the 906 identified studies covered the SE4ML direction. Based on their results, the authors concluded that “it remains difficult to apply SE practices to develop ML/DL systems”. Another systematic literature review was performed by Kumeno [115]. He focused solely on the extraction of SE challenges for ML applications and mapped them to the different SWEBOK areas. In total, he selected 115 papers (47 papers focusing on SE-related challenges for ML and 68 papers focusing on ML-techniques or ML-applications challenges) from 2000 to 2019. Moreover, Lwakatare et al. [126] conducted a similar review of SE challenges faced by industrial practitioners in the context of large-scale ML systems. They categorized 23 challenges found in the 72 papers selected according to four quality attributes (adaptability, scalability, privacy, safety) and four ML process stages (data acquisition, training, evaluation, deployment). Adaptability and scalability were reported to face a significantly larger number of challenges than privacy and safety. They also

	identified 8 solutions, e.g. transfer learning and synthetically generated data, solving up to 13 ofthe challenges, especially adaptability. Giray [71] also conducted a systematic literature review to identify the state of the art and challenges in the area ofML systems engineering. In his sampling method, he exclusively targeted publications from SE venues and selected 141 studies. These studies were then analyzed for their bibliometrics, the used research methods, plus mentioned challenges and proposed solutions. Lastly, Nascimento et al. [147] performed an SLR to analyze how SE practices have been applied to develop AI or ML systems, with special emphasis on limitations and open challenges. They also focus on system contexts, challenges, and SE contribution types. While they considered publications between 1990 and 2019, they only selected 55 papers. We summarize in Table 1 the findings of the related work. This summary has a three-fold objective: (i) to provide a synthetic view of the approaches aforementioned; (ii) to make evident our claim that no systematic mapping or general review with a breadth and depth similar to ours has been published so far; (iii) to facilitate the comparison of the results of our study with the related work. In summary, even though several secondary studies have recently been published or submitted (a few of the aforementioned studies are pre-prints), there are still very few works that broadly summarize and classify research in the field of SE for AI-based systems. No systematic mapping or general review with a breadth and depth similar to ours has been published so far. Existing general reviews focus either exclusively on challenges, analyze considerably fewer studies, or only take publications with an industry context into account, i.e. they partially fail to describe the wide spectrum of results in this research area.
Modified on	22/07/2022 14:56
Name	Software Engineering for AI-Based Systems~ A Survey
Number of Coding References	22
Number of Codes Coding	11
Coverage	4.22%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56
Coded Text	Table 1. Summary of relevant aspects on bibliometrics, AI-based system properties, SE approaches, and challenges found in related work.
Modified on	22/07/2022 14:56
Name	Software Engineering for AI-Based Systems~ A Survey ~ ACM Transactions on Software Engineering and Methodology
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Software engineering for artificial intelligence and machine learning software~ A systematic literature review
Number of Coding References	28
Number of Codes Coding	14
Coverage	10.82%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	
Modified on	22/06/2022 10:16

Name	Software engineering for artificial intelligence and machine learning software~ A systematic literature review
Number of Coding References	28
Number of Codes Coding	14
Coverage	10.82%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	
Modified on	22/06/2022 10:17

Name	Software engineering for artificial intelligence and machine learning software~ A systematic literature review
Number of Coding References	28
Number of Codes Coding	14
Coverage	10.82%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	
Modified on	22/06/2022 10:17

Name	Software engineering for artificial intelligence and machine learning software~ A systematic literature review
Number of Coding References	28
Number of Codes Coding	14
Coverage	10.82%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	In the Design area, most practices are specific for supporting the design of ML systems, such as the proposal of an architectural structure to provide automated continuous experimentation (Mattos et al., 2017). New architectural standards have also been proposed to improve the operational stability of ML systems (Yokoyama, 2019). Additionally, there are architectural proposals to design specific AI systems, such as the proposal for an architectural structure to facilitate the design and implementation of hybrid intelligent forecasting applications (Lertpalangsunti and Chan, 1998). Table 14 details the proposed practices for the software design area.
Modified on	22/06/2022 10:16

Name	Software engineering for artificial intelligence and machine learning software~ A systematic literature review
Number of Coding References	28
Number of Codes Coding	14
Coverage	10.82%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	
Modified on	22/06/2022 10:10

Name	Software engineering for artificial intelligence and machine learning software~ A systematic literature review
Number of Coding References	28
Number of Codes Coding	14
Coverage	10.82%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	
Modified on	22/06/2022 10:10

Name	Software engineering for artificial intelligence and machine learning software~ A systematic literature review
Number of Coding References	28
Number of Codes Coding	14
Coverage	10.82%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Software Mantainability Aspects
Created on	20/06/2022 11:56
Coded Text	
Modified on	22/06/2022 10:14

Name	Software engineering for artificial intelligence and machine learning software~ A systematic literature review
Number of Coding References	28
Number of Codes Coding	14
Coverage	10.82%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Software Mantainability Aspects
Created on	20/06/2022 11:56
Coded Text	
Modified on	22/06/2022 10:14

Name	Software engineering for artificial intelligence and machine learning software~ A systematic literature review
Number of Coding References	28
Number of Codes Coding	14
Coverage	10.82%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Software Mantainability Aspects
Created on	20/06/2022 11:56
Coded Text	
Modified on	22/06/2022 10:14

Name	Software engineering for artificial intelligence and machine learning software~ A systematic literature review
Number of Coding References	28
Number of Codes Coding	14
Coverage	10.82%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Software Mantainability Aspects
Created on	20/06/2022 11:56
Coded Text	
Modified on	22/06/2022 10:15

Name	Software engineering for artificial intelligence and machine learning software~ A systematic literature review
Number of Coding References	28
Number of Codes Coding	14
Coverage	10.82%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Software Mantainability Aspects
Created on	20/06/2022 11:56
Coded Text	
Modified on	22/06/2022 10:15

Name	Software engineering for artificial intelligence and machine learning software~ A systematic literature review
Number of Coding References	28
Number of Codes Coding	14
Coverage	10.82%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	03/02/2022 14:16
Coded Text	4.4.1. Software Testing Testing includes the verification of the AI software or ML model to see if it behaves as expected. The need to validate behaviour of modules using ML, Deep Learning, Recurrent Neural Network, etc. is acknowledged in many primary studies. It is not surprising that the “Software Quality” and “Tests” categories are the most cited topics as challenging in the development of ML/AI systems. Alshangiti et al. (2019), Guo et al., (2019), Rahman et al., (2019), and Amershi et al. (2019) also reported that performing tests on ML systems is challenging because the procedures to perform the tests on these systems are different from the traditional techniques known and applied in non-ML systems. Traditional testing techniques and methods that aim to increase structural coverage towards the exploration of diverse software states is not very useful for DL systems (Kim et al., 2019; Rahman et al., 2019; Xie et al., 2011). The current understanding about characteristics of bugs found in ML systems is currently limited (Thung et al., 2012), and leads to the need for systematic and efficient bug detection and fixing approaches. Challenges related to testing regarding different aspects of AI/ML include bug detection, creating test cases, data testing, debugging, and test criteria. In AI/ML systems, the rules are inferred from training data (that is, they are generated inductively) and not deductively as occurs in non-ML systems (Braiek and Khomh, 2020; Ishikawa, 2019; L. Ma et al., 2018; Xie et al., 2011). “Bug detection” was also one of the most cited main subcategories (Braiek and Khomh, 2020; Selsam et al., 2017). A bug in an ML program may come from its training data, program code, execution environment, or third-party frameworks (Braiek and Khomh, 2020). Compared with traditional software, the dimension and potential testing space of an ML 19 program are both much larger. This leads to the problem of estimating and assigning budgets and efforts to AI/ML testing activities. Implementation errors are difficult to detect and resolve, as there can be many other potential causes of unwanted behaviour in an ML system. For example, an implementation error can lead to incorrect gradients and cause a learning algorithm to be interrupted, but this symptom can also be caused by noise in the training data, poor model choice, unfavourable optimization scenario, research strategy inadequate, or numerical instability (Selsam et al., 2017). Another significant challenge in testing AI software is the lack of oracle (Braiek and Khomh, 2020; Ishikawa, 2019; L. Ma et al., 2018; Xie et al., 2011). It is difficult to clearly define the correctness criteria for system outputs or the right outputs for each individual input. Take away 3: Testing space for AI software is much larger, more heterogeneous and, in many cases, it is difficult to formally define in comparison to traditional software testing.
Modified on	22/06/2022 10:00

Name	Software engineering for artificial intelligence and machine learning software~ A systematic literature review
Number of Coding References	28
Number of Codes Coding	14
Coverage	10.82%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	03/02/2022 14:16
Coded Text	AI Software Quality In the challenge of obtaining “AI Software Quality” in AI/ML systems, three main subcategories stand out: (i) defining the ethics requirements in AI, (ii) interpretability, and (iii) scalability are perceived as the most challenging aspects in these systems (Table 11). For AI ethics requirements, the methods, and tools to implement AI ethics are lacking (Bryson and Winfield, 2017; Vakkuri et al., 2019; Vakkuri and Kemell, 2019). Furthermore, ethics is not a formal requirement in AI projects and developers are not well informed or aware of how ethics should be applied in AI projects, so they use known practical methods (Vakkuri et al., 2019; Vakkuri and Kemell, 2019). Regarding interpretability, the activity of labelling data is related to the transparency of the ML model (Arpteg et al., 2018; Braiek and Khomh, 2020; Lwakatare et al., 2019). In the activity of labelling data, there is a need for processes and tools to accurately and consistently label large data sets (Lwakatare et al., 2019). Due to a lack of transparency in the ML model, the model is inherently irreducible, so an accurate explanation of the model will be as complex as the model (Arpteg et al., 2018). Furthermore, when it comes to implementing ML algorithms, ML engineers sometimes struggle to understand these formulas, rules, or concepts, and often require sophisticated numerical calculations that can be difficult to implement in sophisticated, high performance hardware architectures (Braiek and Khomh, 2020). As for scalability, the challenges are in the process of integrating ML models into scalable software systems (Amershi et al., 2019) and data quality assessment (DIng et al., 2017; Kim et al., 2018). One of the reasons for obtaining poor data quality is due to flaws in the data collection and sampling procedure.
Modified on	22/06/2022 10:01

Name	Software engineering for artificial intelligence and machine learning software~ A systematic literature review
Number of Coding References	28
Number of Codes Coding	14
Coverage	10.82%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	In the challenge of obtaining “AI Software Quality” in AI/ML systems, three main subcategories stand out: (i) defining the ethics requirements in AI, (ii) interpretability, and (iii) scalability are perceived as the most challenging aspects in these systems (Table 11). For AI ethics requirements, the methods, and tools to implement AI ethics are lacking (Bryson and Winfield, 2017; Vakkuri et al., 2019; Vakkuri and Kemell, 2019). Furthermore, ethics is not a formal requirement in AI projects and developers are not well informed or aware of how ethics should be applied in AI projects, so they use known practical methods (Vakkuri et al., 2019; Vakkuri and Kemell, 2019). Regarding interpretability, the activity of labelling data is related to the transparency of the ML model (Arpteg et al., 2018; Braiek and Khomh, 2020; Lwakatare et al., 2019). In the activity of labelling data, there is a need for processes and tools to accurately and consistently label large data sets (Lwakatare et al., 2019). Due to a lack of transparency in the ML model, the model is inherently irreducible, so an accurate explanation of the model will be as complex as the model (Arpteg et al., 2018). Furthermore, when it comes to implementing ML algorithms, ML engineers sometimes struggle to understand these formulas, rules, or concepts, and often require sophisticated numerical calculations that can be difficult to implement in sophisticated, high performance hardware architectures (Braiek and Khomh, 2020). As for scalability, the challenges are in the process of integrating ML models into scalable software systems (Amershi et al., 2019) and data quality assessment (DIng et al., 2017; Kim et al., 2018). One of the reasons for obtaining poor data quality is due to flaws in the data collection and sampling procedure.
Modified on	22/06/2022 10:00
Name	Software engineering for artificial intelligence and machine learning software~ A systematic literature review
Number of Coding References	28
Number of Codes Coding	14
Coverage	10.82%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	RQ3. Which challenges are related to specific aspects of AI/ML development? Fig. 10 presents a hierarchy graph that shows the number of references from different codes with the categories most referenced in the data. The more coding a category has, the larger its area. In addition, the subcategories (the child codes) are grouped into the parent category. Below, we present the categories of challenges ordered by their popularity, including Software Testing (30 references) and AI Software Quality (27). Followed by the categories of Model Development (16), Data Management (16), Project Management (15), Infrastructure (14), and Requirements Engineering (13). The categories of 10 to 6 references were AI Engineering (10), Architecture Design (8), and Model Implementation (6). The categories that had up to two references were Integration (2), Operational Support (1), and Education (1). Table B.18 presents the challenges faced by professionals in the development of AI/ML systems, and in it we present the most evident categories and subcategories, with the highest number of citations.
Modified on	22/06/2022 09:58

Name	Software engineering for artificial intelligence and machine learning software~ A systematic literature review
Number of Coding References	28
Number of Codes Coding	14
Coverage	10.82%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	31/01/2022 14:44
Coded Text	State-of-the-art AI/ML systems rely on high-effort data management tasks, such as data exploration, data preparation, and data cleaning. Challenges regarding the data collection, processing, data availability, and quality are highlighted in our primary studies. The lack of data, the lack of values, the delay in sending data, the lack of metadata, the granularity of data, the scarcity of different samples are challenges related to the availability of data for ML projects. Other challenges are data manipulation and deviation, preparing the data set that includes data dependency, data quality, and data integration with various sources. In addition, the modelling of this data is one of the challenges related to data pre-processing, regarding data cleanliness, categorical data/sequence. In real-life applications, the following are common data problems: lack of metadata missing values data granularity integration data from multiple sources shortage of diverse samples design and management of the database, data lake quality of training data vs. real data One study has highlighted the importance of data dependency, and states that data dependencies cost more than code dependencies in AI/ML systems, i.e. unstable or underutilized data dependencies (Sculley et al., 2015). Another issue mentioned is data drift, meaning that the statistical properties of predicting variables changing in an unforeseen way (Lwakatare et al., 2019; Munappy et al., 2019). Handling of data drifts in uploaded data, invalidation of models, e.g., due to changes in data sources, and the need to monitor models in production for staleness are problems mentioned in Lwakatare et al. (2019). Take away 5: AI development processes need to integrate infrastructures, processes and tools for managing data as their integral parts. It is not AI software, but AI data and software engineering.
Modified on	03/02/2022 15:57

Name	Software engineering for artificial intelligence and machine learning software~ A systematic literature review
Number of Coding References	28
Number of Codes Coding	14
Coverage	10.82%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	06/01/2022 15:15
Coded Text	Architecture Design Challenges are related to the architectural aspects of ML models, AI software or that which covers the entire developed system. Sculley et al. (2015) list the following risk factors when designing AI/ML systems: boundary erosion, entanglement, hidden feedback loops, undeclared consumers, data dependencies, configuration issues, changes in the external world. These authors (Sculley et al., 2015) also emphasize the concern of entanglement with the CACE principle of “changing anything changes everything”. From a system perspective, the design of AI/ML software needs to be flexible in order to accommodate the rapidly evolving ML components (Rahman et al., 2019). Lwakatare et al. (2019) describe a specific type of design trade-off in customization of AI/ML platform functionalities. 4.4.9. Model deployment Challenges regarding the deployment of the ML model in real or test environments involve dependency management, maintaining the glue code, monitoring and logging, and the unintended feedback loops. For the deployment process, when deploying the trained models from a testing environment to an operating one, there lacks a benchmarking understanding of the migration and quantization processes, such as the impacts on prediction accuracy and performance (Guo et al., 2019). Relating to the deployment process, changing hardware and software, issues to maintain reproducible results, incur engineering costs for keeping software and hardware up to date (Munappy et al., 2019). 4.4.10. Integration The integration challenge occurs when integrating AI/ML modules to a larger system (Amershi et al., 2019; Tamir and Kandel, 1995), which impacts other engineering activities, such as deployment, testing, and maintenance. 24 4.4.11. AI Engineering Challenges involve engineering models, processes, and practices when developing AI products, such as automation, lack of unifying different processes, lack of defined development process, lack of patterns, and the need for continuous engineering. Take away 8: The major challenges professionals face in the development of AI/ML systems include: testing, AI software quality, data management, model development, project management, infrastructure, and requirement engineering.
Modified on	22/06/2022 10:07

Name	Software engineering for artificial intelligence and machine learning software~ A systematic literature review
Number of Coding References	28
Number of Codes Coding	14
Coverage	10.82%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	06/01/2022 15:15
Coded Text	Infrastructure Infrastructure is a practical consideration when building and operating ML models. The challenges are related to the acquisition, installation, configuration, and maintenance of the necessary infrastructure for the development and operation of AI/ML Systems. One of the main engineering challenges perceived from managers was difficulty in building infrastructure (Lwakatare et al., 2019), and include the following: Hardware: whether local infrastructure or a cloud resource is used for each task of the ML development pipeline Platform: operating systems and installed packages. It might be a requirement that multiple platforms need to be executed simultaneously Tools: end-to-end pipeline supports include adoption of existing AI tools. Tools are the most mentioned item in the category Infrastructure. Kim et al. (2018) state that “despite a large suite of diverse tools, it is difficult for data scientists to access the right tools, because generic tools do not work for the specific problems that they have. It difficult to stay current with evolving tools, as they have other responsibilities and occasionally engage in data science work.” Literature also reveals the lack of logging and tracking tools for AI development processes. As seen in Hilllaz et al. (2016), AI developers tend to track workflow via informal methods, such as emails among colleagues or notes to themselves. Complex and poor logging mechanisms during AI experiments are also stated, as seen in Lwakatare et al. (2019) and Hilllaz et al. (2016).
Modified on	22/06/2022 10:07
Name	Software engineering for artificial intelligence and machine learning software~ A systematic literature review
Number of Coding References	28
Number of Codes Coding	14
Coverage	10.82%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	03/02/2022 10:24
Coded Text	Model development The list of the challenges regarding the building of the ML model is related to different aspects, such as a large number of model inputs, the AI ethics implementation, formal representation of complex models, optimization of feature engineering, imperfection and accuracy assurance, invalidation of models, model localization with data constraints, module documentation, uncertainty in input-output relationships, and uncertainty in model behaviour. In model development, the main challenge is to obtain a large number of model inputs (Ishikawa, 2019; Renggli et al., 2019). It is difficult to clearly define the correction criteria for system outputs or correct outputs for each individual model input. Furthermore, for systems with a supervised learning paradigm, it is difficult to obtain labelled data that will serve as an 22 input for the model, mainly when there is a large volume of unlabelled data. For a supervised learning paradigm, all samples must be labelled before the continuous integration service starts to work.
Modified on	23/02/2022 19:45

Name	Software engineering for artificial intelligence and machine learning software~ A systematic literature review
Number of Coding References	28
Number of Codes Coding	14
Coverage	10.82%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:24
Coded Text	Testing includes the verification of the AI software or ML model to see if it behaves as expected. The need to validate behaviour of modules using ML, Deep Learning, Recurrent Neural Network, etc. is acknowledged in many primary studies. It is not surprising that the “Software Quality” and “Tests” categories are the most cited topics as challenging in the development of ML/AI systems. Alshangiti et al. (2019), Guo et al., (2019), Rahman et al., (2019), and Amershi et al. (2019) also reported that performing tests on ML systems is challenging because the procedures to perform the tests on these systems are different from the traditional techniques known and applied in non-ML systems. Traditional testing techniques and methods that aim to increase structural coverage towards the exploration of diverse software states is not very useful for DL systems (Kim et al., 2019; Rahman et al., 2019; Xie et al., 2011). The current understanding about characteristics of bugs found in ML systems is currently limited (Thung et al., 2012), and leads to the need for systematic and efficient bug detection and fixing approaches. Challenges related to testing regarding different aspects of AI/ML include bug detection, creating test cases, data testing, debugging, and test criteria. In AI/ML systems, the rules are inferred from training data (that is, they are generated inductively) and not deductively as occurs in non-ML systems (Braiek and Khomh, 2020; Ishikawa, 2019; L. Ma et al., 2018; Xie et al., 2011). “Bug detection” was also one of the most cited main subcategories (Braiek and Khomh, 2020; Selsam et al., 2017). A bug in an ML program may come from its training data, program code, execution environment, or third-party frameworks (Braiek and Khomh, 2020). Compared with traditional software, the dimension and potential testing space of an ML 19 program are both much larger. This leads to the problem of estimating and assigning budgets and efforts to AI/ML testing activities. Implementation errors are difficult to detect and resolve, as there can be many other potential causes of unwanted behaviour in an ML system. For example, an implementation error can lead to incorrect gradients and cause a learning algorithm to be interrupted, but this symptom can also be caused by noise in the training data, poor model choice, unfavourable optimization scenario, research strategy inadequate, or numerical instability (Selsam et al., 2017). Another significant challenge in testing AI software is the lack of oracle (Braiek and Khomh, 2020; Ishikawa, 2019; L. Ma et al., 2018; Xie et al., 2011). It is difficult to clearly define the correctness criteria for system outputs or the right outputs for each individual input.
Modified on	03/02/2022 15:56

Name	Software engineering for artificial intelligence and machine learning software~ A systematic literature review
Number of Coding References	28
Number of Codes Coding	14
Coverage	10.82%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:24
Coded Text	Testing space for AI software is much larger, more heterogeneous and, in many cases, it is difficult to formally define in comparison to traditional software testing.
Modified on	03/02/2022 15:56

Name	Software engineering for artificial intelligence and machine learning software~ A systematic literature review
Number of Coding References	28
Number of Codes Coding	14
Coverage	10.82%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	Project Management area competence (Alshangiti et al., 2019; Ishikawa, 2019; Martin and Schreckenghost, 2004; Selsam et al., 2017) and communication (Ishikawa, 2019; Vakkuri and Kemell, 2019) are challenges that stand out. As for competence, the main challenges are in resource management to meet the different platforms and technologies, knowledge of statistics and mathematics (need for specialization in these areas) (Ishikawa, 2019; Selsam et al., 2017), and skills required in different ML domains. For instance, the problem formulation phase requires a good overall understanding of how ML techniques work in order to help transform the problem in hand into an appropriate ML learning task. Model development requires hands-on and practical knowledge of tools, libraries and algorithms to build models (Alshangiti et al., 2019). Model fitting and tuning requires a solid understanding of the mathematical and statistical intuition behind the models (Alshangiti et al., 2019). Communicating technical decisions and rationale during AI/ML model development is a non-trial task. Customers, even one with a technical background, do not often understand the technical details of AI models. It is challenging when one needs to explain the model functions to business stakeholders, which makes it difficult for the client to make decisions and evaluate the results (Ishikawa, 2019). Besides, communication of process, practice, and ethical regulation within the technical team is also overlooked. The direct consequence of this is the additional difficulty in estimating and assigning time and resources for the AI/ML parts (Munappy et al., 2019). Take away 6: AI project managers need competence and knowledge to act as a boundary-spanning role across AI/ML and non-AI/ML worlds
Modified on	03/02/2022 15:58

Name	Software engineering for artificial intelligence and machine learning software~ A systematic literature review
Number of Coding References	28
Number of Codes Coding	14
Coverage	10.82%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:27
Coded Text	In the challenge of obtaining “AI Software Quality” in AI/ML systems, three main subcategories stand out: (i) defining the ethics requirements in AI, (ii) interpretability, and (iii) scalability are perceived as the most challenging aspects in these systems (Table 11). For AI ethics requirements, the methods, and tools to implement AI ethics are lacking (Bryson and Winfield, 2017; Vakkuri et al., 2019; Vakkuri and Kemell, 2019). Furthermore, ethics is not a formal requirement in AI projects and developers are not well informed or aware of how ethics should be applied in AI projects, so they use known practical methods (Vakkuri et al., 2019; Vakkuri and Kemell, 2019). Regarding interpretability, the activity of labelling data is related to the transparency of the ML model (Arpteg et al., 2018; Braiek and Khomh, 2020; Lwakatare et al., 2019). In the activity of labelling data, there is a need for processes and tools to accurately and consistently label large data sets (Lwakatare et al., 2019). Due to a lack of transparency in the ML model, the model is inherently irreducible, so an accurate explanation of the model will be as complex as the model (Arpteg et al., 2018). Furthermore, when it comes to implementing ML algorithms, ML engineers sometimes struggle to understand these formulas, rules, or concepts, and often require sophisticated numerical calculations that can be difficult to implement in sophisticated, high performance hardware architectures (Braiek and Khomh, 2020). As for scalability, the challenges are in the process of integrating ML models into scalable software systems (Amershi et al., 2019) and data quality assessment (DIng et al., 2017; Kim et al., 2018). One of the reasons for obtaining poor data quality is due to flaws in
Modified on	03/02/2022 15:56

Name	Software engineering for artificial intelligence and machine learning software~ A systematic literature review
Number of Coding References	28
Number of Codes Coding	14
Coverage	10.82%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:27
Coded Text	Besides Interpretability, empirical research highlights various engineering challenges in regard to AI ethics.
Modified on	03/02/2022 15:57

Name	Software engineering for artificial intelligence and machine learning software~ A systematic literature review
Number of Coding References	28
Number of Codes Coding	14
Coverage	10.82%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:27
Coded Text	For example, for Tests, bug detection practices in ML systems (Guo et al., 2019; Thung et al., 2012) have been adjusted to identify errors in ML systems and on Deep Learning (DL) platforms. In the area of configuration management, Lwakatare et al. (2019), Mattos et al. (2017) and Renggli et al. (2019) proposed the creation of experimentation environments and automation/continuous integration. In addition, different guidelines proposed by Rahman et al. (2019), Amershi et al. (2019), Kim et al. (2018), Martin and Schreckenghost (2004), Hilllaz et al. (2016), Martin (2016), and Hannay et al. (2009) were tried by professionals in real ML/AI projects. These practices include guidelines for reuse, data automation, and data traceability to support configuration management. In the area of Software Quality practices to meet AI ethics, the authors cited existing proposals to support ethics in AI projects in planning and design, such as ART principles (Dignum, 2019), Guidelines for Ethically Aligned Design (How, 2018), Ethics Guidelines for Trustworthy AI (High Level Independent Group on Artificial Intelligence (AI HLEG), 2019). In addition to these, we found new proposals, such as practices to meet some aspects of AI ethics (Vakkuri et al., 2019) that are based on practices developed by professionals working in the area. The practice for ETHICS IN AI - are ethical considerations in Artificial Intelligence and Autonomous Systems (Bryson and Winfield, 2017), which gathers compliance standards regarding the confidence in the effectiveness of a system in important areas for users, such as safety, protection, and reliability. It is an approach called 'RESOLVEDD' and is strategic to project the AI ethics (Vakkuri and Kemell, 2019) defining aspects, such as the commitment to the ethically aligned project, transparency in design, and responsibility in design. For Testing, SE practices were directed to give specific Support in ML systems Tests, General Techniques/Guidelines that include practices for ML systems tests, bug detection in ML systems, stress test, and specific tests for neural networks/Deep Learning. Table 12 details the proposed practices for the software testing area.
Modified on	23/02/2022 19:48

Name	Software engineering for artificial intelligence and machine learning software~ A systematic literature review
Number of Coding References	28
Number of Codes Coding	14
Coverage	10.82%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:27
Coded Text	In the challenge of obtaining “AI Software Quality” in AI/ML systems, three main subcategories stand out: (i) defining the ethics requirements in AI, (ii) interpretability, and (iii) scalability are perceived as the most challenging aspects in these systems (Table 11). For AI ethics requirements, the methods, and tools to implement AI ethics are lacking (Bryson and Winfield, 2017; Vakkuri et al., 2019; Vakkuri and Kemell, 2019). Furthermore, ethics is not a formal requirement in AI projects and developers are not well informed or aware of how ethics should be applied in AI projects, so they use known practical methods (Vakkuri et al., 2019; Vakkuri and Kemell, 2019). Regarding interpretability, the activity of labelling data is related to the transparency of the ML model (Arpteg et al., 2018; Braiek and Khomh, 2020; Lwakatare et al., 2019). In the activity of labelling data, there is a need for processes and tools to accurately and consistently label large data sets (Lwakatare et al., 2019). Due to a lack of transparency in the ML model, the model is inherently irreducible, so an accurate explanation of the model will be as complex as the model (Arpteg et al., 2018). Furthermore, when it comes to implementing ML algorithms, ML engineers sometimes struggle to understand these formulas, rules, or concepts, and often require sophisticated numerical calculations that can be difficult to implement in sophisticated, high performance hardware architectures (Braiek and Khomh, 2020). As for scalability, the challenges are in the process of integrating ML models into scalable software systems (Amershi et al., 2019) and data quality assessment (DIng et al., 2017; Kim et al., 2018). One of the reasons for obtaining poor data quality is due to flaws in the data collection and sampling procedure.
Modified on	03/02/2022 15:57

Name	Software engineering for artificial intelligence and machine learning software~ A systematic literature review
Number of Coding References	28
Number of Codes Coding	14
Coverage	10.82%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56
Coded Text	2.1. AI/ML software system A simple definition of Artificial Intelligence would be that it is the science of mimicking human mental faculties on a computer (Hopgood, 2005). Thus, AI is used to describe machines that mimic "cognitive" functions that humans associate with other human minds, such as "learning" and "problem-solving" (Russel and Norvig, 2012). AI systems include modules that present an opportunity to generate types of learning. ML is a type of artificial intelligence technique that makes decisions or predictions based on data, according to Agrawal et al. (2017). According to Bosch et al. (2020) the emerging field of AI engineering is an extension of software engineering, and one which includes new processes and technologies needed for the development and evolution of AI systems, i.e. systems that include AI components, such as ML. Thus, researchers have investigated the challenges, processes, and different ways to understand the processes involved in developing these systems. 2.2. Challenges and Practices to AI/ML systems Previous research has examined software engineering challenges and practices for ML, such as Rahman et al. (2019), Amershi et al. (2019), Nascimento et al., (2019), Ghofrani et al. (2019), Braiek and Khomh (2020), Lwakatare et al. (2019), Arpteg et al. (2018), and NguyenDuc et al. (2020). However, they focused on presenting a set of challenges and practices analyzed separately in specific contexts. For example, Amershi et al. (2019) carried out a study with professionals working in different AI/ML teams in a large company, in this case, Microsoft. The authors' aim was to collect challenges and the best practices for SE that are used in the organization's internal projects. They reported different challenges and practices used in the lifecycle stages of the development of AI/ML projects. However, the set of challenges and practices reported by the authors is broad and the SE practices are basically a contribution of lessons learned in the projects, however, these practices are not actionable in other situations. Sculley et al. (2015) investigated the risk factors of ML components in software projects and used the technical debt framework to explore these factors. The authors concluded that ML components are more likely to incur technical debts because they have the same maintenance problems as non-ML systems, as well as a set of problems specific to ML components. As a contribution, the authors present a set of anti-patterns and practices designed to avoid technical debt in systems that use ML components. Rahman et al. (2019) performed a study in an industrial context and compiled a set of recommendations regarding SE practices found in the development of ML applications, the authors' summarized the practices in three distinct perspectives: (1) software engineering, (2) machine learning, and (3) industry-academia collaboration. Munappy et al. (2019) present challenges applied in specific domains, such as data management for Deep Learning (DL) systems. The authors identified and categorized data management challenges faced by practitioners in different stages of end-to-end development. In addition, Arpteg et al. (2018) investigated SE challenges in DL systems and presented a set of twelve challenges categorized in three areas of development, production, and organizational challenges. Nascimento et al. (2019) analyzed the challenges of ML system development in three small companies. The authors found that the challenges faced by the developers are, for instance, not having a defined process, difficulty identifying customer business metrics and 3 difficulty designing the database structure. The authors suggest two checklists to support developers in overcoming the challenges faced in the stages of problem understanding and data handling. Nguyen-Duc et al. (2020) proposed a research agenda for the continuous development of AI software. In their proposal, several challenges of maintaining the engineering practices and processes of AI software development at the continuous level are reported, i.e., AI software quality, requirement engineering, and tools and infrastructure. However, none of the authors mentioned above explored the relationships and dependency between the challenges and practices of SE, or identify which contexts and which SE areas may be applicable, or even which challenges are met. Even though these studies provide an understanding of specific contexts of AI software development, they do not give an overall picture of AI

	challenges in different experimental or industrial contexts. Moreover, the SE practices recommended by the authors are not reproduceable, in other words, they are not actionable or can be used by professionals and in research. This motivates us to create a more useful systematic literature review. 2.3. Existing literature reviews regarding SE for AI/ML In order to avoid repeating previous work, it is necessary to identify existing reviews on the topic. At the time of writing this study (summer 2020), we noted several literature reviews relevant to our topic. However, their objective, scope, and types of outcomes are not similar to what our intentions in this paper. For instance, Zhang et al. (2020) conducted a review of 138 papers on testing in ML systems. The authors identified different ML test properties, test components, and test workflows. In addition, they summarized data sets used for experiments and the open-source testing tools/frameworks available, and analyzed the research trend, directions, opportunities and challenges in ML testing (Zhang et al., 2020). Gonçalves et al. (2019) conducted an SLR to investigate intelligent user interface (IUI) design trends including interfaces generating by artificial intelligence systems, in the context of contemporary software systems, such as software systems based on the Internet of Things (IoT) or dedicated to smart cities. The preliminary results show that the models and technologies most used to develop IUIs are through tools, processes, frameworks. Interestingly, the most mentioned domain is health. Washizaki et al. (2019) conducted an SLR and identified software quality methods and techniques for ML applications. The authors highlighted the problems in ML applications and revealed SE approaches and software testing research areas in order to address these problems. Nevertheless, there are systematic reviews on more specific domains of AI/ML systems, such as architectural and design standards for ML projects (Washizaki et al., 2019). The authors studied and classified patterns and anti-patterns of software architecture for ML, extracted from white and gray literature. Hutchins et al. (2017) carried out a literature review on AI/Autonomous Systems (AS) design guidelines and a case study of the different incidences in which AI/AS systems can be involved in the production of many of the social and ethical dilemmas caused by the development of the system AI/AS. However, ours is the first research that makes a comprehensive review of the literature on SE challenges and practices, and details the context in which these practices were applied and what type of contribution each practice proposes in order to support the development of AI/ML based on the classification of Paternoster et al. (2014). Furthermore, we analyze the challenges jointly with the proposed solutions.
Modified on	23/02/2022 19:39

Name	Software engineering for artificial intelligence and machine learning software~ A systematic literature review
Number of Coding References	28
Number of Codes Coding	14
Coverage	10.82%
Folder Location	Codes\\Snowballing
Parent Name	Codes\\Snowballing\\Snowballing
Created on	21/06/2022 14:00
Coded Text	Recent technological advancements regarding cloud computing, big data management, algorithms, and tools have enabled a lot of opportunities for businesses, industries, and societies to make use of Artificial Intelligence. As such, Artificial Intelligence (AI), more specifically Machine Learning (ML) systems have widely been adopted by companies in all industries worldwide as value propositions in order to create or extend the services and products they offer (Rahman et al., 2019; Amershi et al., 2019). Almost all organizations today, from private to public sectors, are involved in some forms of AI initiatives (Bosch et al., 2020). The state-of-the-art AI/ML systems are quickly moving from a laboratory environment to an industrial one, and aim to increase the massive amount of accessible data available. Due to the growing popularity of industrial AI/ML, a rapidly increasing amount of studies have been performed to understanding the processes, practices, and challenges faced by professionals in different companies (Byrne et al., 2017; Kim et al., 2018; Rahman et al., 2019; Amershi et al., 2019; Nascimento et al., 2019). Developing industrial AI/ML has come with several engineering problems that are different from those in traditional, non-AI/ML software development (Brynjolfsson and Mitchell, 2017; Byrne et al., 2017; Guo et al., 2019; Rahman et al., 2019; Amershi et al., 2019). For example, studies show some of the challenges that professionals face when developing AI/ML systems, for example: identifying customer business metrics, lack of a defined process, and data-centric engineering challenges (Brynjolfsson and Mitchell, 2017; Byrne et al., 2017; Nascimento et al., 2019). In comparison to AI research, which focuses on algorithmic development or ML applications, engineering AI systems, i.e., tools, processes, and practices of managing, testing, 1 and deploying ML models is a research area that has recently emerged (Kim et al., 2018; Amershi et al., 2019). In the last five years, we have observed that the area has rapidly evolved. A search on the Scopus database using the term (“artificial intelligence” OR “machine learning”) AND (“software development”) shows 2292 results, indicating a comparatively large amount of research articles about the topic. Existing literature has revealed separate challenges faced by professionals in the development of ML systems and software engineering (SE) (Arpteg et al., 2018; Braiek and Khomh, 2020; Ghofrani et al., 2019; Kim et al., 2018; Lwakatare et al., 2019; Rahman et al., 2019; Amershi et al., 2019; Nascimento et al., 2019). Literature suggests that some practices are adopted by expert professionals or specialized software teams in AI/ML for different aspects of their engineering activities (Barash et al., 2019; Chakravarty, 2010; Ghofrani et al., 2019; Martin, 2016; Renggli et al., 2019; Amershi et al., 2019; Nascimento et al., 2019; Washizaki et al., 2019), however, they are often case-specific, and are due to a lack information regarding the context, and about problems on how these practices can be adopted. Hence, for both research and practice, it is necessary to have a consolidated body of knowledge that connects problems and SE practices that have been applied the development of AI/ML in industry.
Modified on	22/06/2022 09:52

Name	Software engineering for machine learning~ a case study
Number of Coding References	16
Number of Codes Coding	3
Coverage	11.15%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Software Mantainability Aspects
Created on	20/06/2022 11:56
Coded Text	Compliance Microsoft issued a set of principles around uses of AI in the open world. These include fairness, accountability, transparency, and ethics. All teams at Microsoft have been asked to align their engineering practices and the behaviors of fielded software and services in accordance with these principles. Respect for them is a high priority in software engineering and AI and ML processes and practices.
Modified on	20/06/2022 12:00

Name	Software engineering for machine learning~ a case study
Number of Coding References	16
Number of Codes Coding	3
Coverage	11.15%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Software Mantainability Aspects
Created on	20/06/2022 11:56
Coded Text	Data availability, collection, cleaning, and management Since many machine learning techniques are centered around learning from large datasets, the success of ML-centric projects often heavily depends on data availability, quality and management [28]. Labeling datasets is costly and timeconsuming, so it is important to make them available for use within the company (subject to compliance constraints). Our respondents confirm that it is important to “reuse the data as much as possible to reduce duplicated effort.” In addition to availability, our respondents focus most heavily on supporting the following data attributes: “accessibility, accuracy, authoritativeness, freshness, latency, structuredness, ontological typing, connectedness, and semantic joinability.” Automation is a vital cross-cutting concern, enabling teams to more efficiently aggregate data, extract features, synthesize labelled examples. The increased efficiency enables teams to “speed up experimentation and work with live data while they experiment with new models.” We found that Microsoft teams have found it necessary to blend data management tools with their ML frameworks to avoid the fragmentation of data and model management activities. A fundamental aspect of data management for machine learning is the rapid evolution of data sources. Continuous changes in data may originate either from (i) operations initiated by engineers themselves, or from (ii) incoming fresh data (e.g., sensor data, user interactions).
Modified on	20/06/2022 11:58

Name	Software engineering for machine learning~ a case study
Number of Coding References	16
Number of Codes Coding	3
Coverage	11.15%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Software Mantainability Aspects
Created on	20/06/2022 11:56
Coded Text	Education and Training The integration of machine learning continues to become more ubiquitous in customer-facing products, for example, machine learning components are now widely used in productivity software (e.g., email, word processing) and embedded devices (i.e., edge computing). Thus, engineers with traditional software engineering backgrounds need to learn how to work alongside of the ML specialists. A variety of players within Microsoft have found it incredibly valuable to scaffold their engineers' education in a number of ways. First, the company hosts a twice-yearly internal conference on machine learning and data science, with at least one day devoted to introductions 295 to the basics of technologies, algorithms, and best practices.
Modified on	20/06/2022 11:59

Name	Software engineering for machine learning~ a case study
Number of Coding References	16
Number of Codes Coding	3
Coverage	11.15%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Software Mantainability Aspects
Created on	20/06/2022 11:56
Coded Text	End-to-end pipeline support As machine learning components have become more mature and integrated into larger software systems, our participants recognized the importance of integrating ML development support into the traditional software development infrastructure. They noted that having a seamless development experience covering (possibly) all the different stages described in Figure 1 was important to automation. However, achieving this level of integration can be challenging because of the different characteristics of ML modules compared with traditional software components. For example, previous work in this field [18], [19] found that variation in the inherent uncertainty (and error) of data-driven learning algorithms and complex component entanglement caused by hidden feedback loops could impose substantial changes (even in specific stages) which were previously well understood in software engineering (e.g., specification, testing, debugging, to name a few). Nevertheless, due to the experimental and even more iterative nature of ML development, unifying and automating the dayto-day workflow of software engineers reduces overhead and facilitate progress in the field.
Modified on	20/06/2022 11:56

Name	Software engineering for machine learning~ a case study
Number of Coding References	16
Number of Codes Coding	3
Coverage	11.15%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Software Mantainability Aspects
Created on	20/06/2022 11:56
Coded Text	Varied Perceptions We found that as a number of product teams at Microsoft in- tegrated machine learning components into their applications, their ability to do so effectively was mediated by the amount of prior experience with machine learning and data science. Some teams fielded data scientists and researchers with decades of experience, while others had to grow quickly, picking up their own experience and more-experienced team members on the way. Due to this heterogeneity, we expected that our survey respondents' perceptions of the challenges their teams' faced in practicing machine learning would vary accordingly. We grouped the respondents into three buckets (low, medium, and high), evenly divided by the number of years of experience respondents personally had with AI. First, we ranked each of the card sorted categories of respondents' challenges divided by the AI experience buckets. This list is presented in Table II, initially sorted by the respondents with low experience with AI.
Modified on	20/06/2022 12:01

Name	Software engineering for machine learning~ a case study
Number of Coding References	16
Number of Codes Coding	3
Coverage	11.15%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	31/01/2022 14:42
Coded Text	
Modified on	03/02/2022 15:48

Name	Software engineering for machine learning~ a case study
Number of Coding References	16
Number of Codes Coding	3
Coverage	11.15%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	31/01/2022 14:42
Coded Text	In this paper, we describe a study in which we learned how various Microsoft software teams build software applications with customer-focused AI features. For that, Microsoft has integrated existing Agile software engineering processes with AI-specific workflows informed by prior experiences in developing early AI and data science applications. In our study, we asked Microsoft employees about how they worked through the growing challenges of daily software development specific to AI, as well as the larger, more essential issues inherent in the development of large-scale AI infrastructure and applications. With teams across the company having differing amounts of work experience in AI, we observed that many issues reported by newer teams dramatically drop in importance as the teams mature, while some remain as essential to the practice of largescale AI. We have made a first attempt to create a process maturity metric to help teams identify how far they have come on their journeys to building AI applications. As a key finding of our analyses, we discovered three funda- mental differences to building applications and platforms for training and fielding machine-learning models than we have seen in prior application domains. First, machine learning is all about data. The amount of effort and rigor it takes to discover, source, manage, and version data is inherently more complex and different than doing the same with software code. Second, building for customizability and extensibility of models require teams to not only have software engineering skills but almost
Modified on	03/02/2022 15:48

Name	Software engineering for machine learning~ a case study
Number of Coding References	16
Number of Codes Coding	3
Coverage	11.15%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	31/01/2022 14:42
Coded Text	Many teams at Microsoft have put significant effort into developing an extensive portfolio of AI applications and platforms by integrating machine learning into existing software engineering processes and cultivating and growing ML talent. In this paper, we described the results of a study to learn more about the process and practice changes undertaken by a number of Microsoft teams in recent years. From these findings, we synthesized a set of best practices to address issues fundamental to the large-scale development and deployment of ML-based applications. Some reported issues were correlated with the respondents' experience with AI, while others were applicable to most respondents building AI applications. We presented a ML process maturity metric to help teams selfassess how well they work with machine learning and offer guidance towards improvements. Finally, we identified three aspects of the AI domain that make it fundamentally different than prior application domains. Their impact will require significant research efforts to address in the future.
Modified on	03/02/2022 15:49

Name	Software engineering for machine learning~ a case study
Number of Coding References	16
Number of Codes Coding	3
Coverage	11.15%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	31/01/2022 14:42
Coded Text	The lessons we identified via studies of a variety of teams at Microsoft who have adapted their software engineering processes and practices to integrate machine learning can help other software organizations embarking on their own paths towards building AI applications and platforms. In this paper, we offer the following contributions. 1) A description of how several Microsoft software engineering teams work cast into a nine-stage workflow for integrating machine learning into application and platform development. 2) A set of best practices for building applications and platforms relying on machine learning. 3) A custom machine-learning process maturity model for assessing the progress of software teams towards excellence in building AI applications. 4) A discussion of three fundamental differences in how software engineering applies to machine-learning–centric components vs. previous application domains.
Modified on	03/02/2022 15:48

Name	Software engineering for machine learning~ a case study
Number of Coding References	16
Number of Codes Coding	3
Coverage	11.15%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	31/01/2022 14:42
Coded Text	This goal has forced organizations to evolve their development processes. We report on a study that we conducted on observing software teams at Microsoft as they develop AI-based applications. We consider a nine-stage workflow process informed by prior experiences developing AI applications (e.g., search and NLP) and data science tools (e.g. application diagnostics and bug reporting). We found that various Microsoft teams have united this workflow into preexisting, well-evolved, Agile-like software engineering processes, providing insights about several essential engineering challenges that organizations may face in creating large-scale AI solutions for the marketplace. We collected some best practices from Microsoft teams to address these challenges. In addition, we have identified three aspects of the AI domain that make it fundamentally different from prior software application domains: 1) discovering, managing, and versioning the data needed for machine learning applications is much more complex and difficult than other types of software engineering, 2) model customization and model reuse require very different skills than are typically found in software teams, and 3) AI components are more difficult to handle as distinct modules than traditional software components — models may be “entangled” in complex ways and experience non-monotonic error behavior. We believe that the lessons learned by Microsoft teams will be valuable to other organizations.
Modified on	03/02/2022 15:41

Name	Software engineering for machine learning~ a case study
Number of Coding References	16
Number of Codes Coding	3
Coverage	11.15%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	
Modified on	20/06/2022 11:54

Name	Software engineering for machine learning~ a case study
Number of Coding References	16
Number of Codes Coding	3
Coverage	11.15%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	In addition, we have identified three aspects of the AI domain that make it fundamentally different from prior software application domains: 1) discovering, managing, and versioning the data needed for machine learning applications is much more complex and difficult than other types of software engineering, 2) model customization and model reuse require very different skills than are typically found in software teams, and 3) AI components are more difficult to handle as distinct modules than traditional software components — models may be “entangled” in complex ways and experience non-monotonic error behavior. We believe that the lessons learned by Microsoft teams will be valuable to other organizations.
Modified on	20/06/2022 11:42

Name	Software engineering for machine learning~ a case study
Number of Coding References	16
Number of Codes Coding	3
Coverage	11.15%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	related line of thought, recent work also discusses the impact that the use of ML-based software has on risk and safety concerns of ISO standards [21]. In the last five years, there have been multiple efforts in industry to automate this process by building frameworks and environments to support the ML workflow and its experimental nature [1], [22], [23]. However, ongoing research and surveys show that engineers still struggle to operationalize and standardize working processes [9], [24], [23]. The goal of this work is to uncover detailed insights on ML-specific best practices used by developers at Microsoft. We share these insights with the broader community aspiring that such take-away lessons can be valuable to other companies and engineers.
Modified on	20/06/2022 11:51

Name	Software engineering for machine learning~ a case study
Number of Coding References	16
Number of Codes Coding	3
Coverage	11.15%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	The need for adjusting software engineering practices in the recent era has been discussed in the context of hidden technical debt [18] and troubleshooting integrative AI [19], [20]. This work identifies various aspects of ML system architecture and requirements which need to be considered during system design. Some of these aspects include hidden feedback loops, component entanglement and eroded boundaries, nonmonotonic error propagation, continuous quality states, and mismatches between the real world and evaluation sets. On a
Modified on	20/06/2022 11:51

Name	Software engineering for machine learning~ a case study
Number of Coding References	16
Number of Codes Coding	3
Coverage	11.15%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	The software engineering discipline has long considered software process improvement as one of its vital functions. Researchers and practitioners in the field have developed several well-known metrics to assess it, including the Capability Maturity Model (CMM) [26] and Six Sigma [27]. CMM rates the software processes of organizations on five levels, from initial (ad hoc processes), repeatable, defined, capable (i.e., quantitatively measured), and efficient (i.e., deliberate process improvement). Inspired by CMM, we build a first maturity model for teams building systems and platforms that integrate machine learning components
Modified on	20/06/2022 11:54

Name	Software engineering for machine learning~ a case study
Number of Coding References	16
Number of Codes Coding	3
Coverage	11.15%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	their journeys to building AI applications. As a key finding of our analyses, we discovered three funda- mental differences to building applications and platforms for training and fielding machine-learning models than we have seen in prior application domains. First, machine learning is all about data. The amount of effort and rigor it takes to discover, source, manage, and version data is inherently more complex and different than doing the same with software code. Second, building for customizability and extensibility of models require teams to not only have software engineering skills but almost
Modified on	20/06/2022 11:48

Name	Software Framework for Data Fault Injection to Test Machine Learning Systems
Number of Coding References	6
Number of Codes Coding	3
Coverage	9.28%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	
Modified on	03/02/2022 15:36

Name	Software Framework for Data Fault Injection to Test Machine Learning Systems
Number of Coding References	6
Number of Codes Coding	3
Coverage	9.28%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24

Coded Text	Conceptual view of the system with examples of questions that the system can answer. software framework, illustrated conceptually in Fig. 1, with the following goals: – Easy and flexible modeling of the types of data faults the system is likely to encounter via a combination of predefined parameterizable fault models and new userdefined ones. Ideally, the set of predefined fault models should gradually grow as a result of the integration of new fault models for new purposes. – Ability to work with different kinds of structured and unstructured data as well as with highly different ML models or systems. – Parameterization of the data fault generation so that developers can study how sensitive their systems are to different kinds of faults and what are the thresholds of data problems when the system starts to lose its performance. – Visualization of the results with different fault models and parameters. – Bookkeeping to allow going back to the sources of problems. – Embedding the fault injection and the visualization of the effects of faulty data to the development pipeline. – Integration of data fault emulation to different development pipelines. To satisfy these goals, we have created a generator frame- work for emulating data problems, called dpEmu. The framework can generate faults in training or testing data in a controlled and documentable manner, and it enables emulating data problems in the use and training of ML systems as depicted in Fig. 2. The Runner routine introduces faults in a dataset, following the definitions set in the Fault generation tree. Then, the resulting data is preprocessed and run by ML models, which
Modified on	03/02/2022 15:36

Name	Software Framework for Data Fault Injection to Test Machine Learning Systems
Number of Coding References	6
Number of Codes Coding	3
Coverage	9.28%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	To develop robust and reliable ML systems we have created dpEmu to encourage developers to evaluate how their models and systems work when system input data has faults. The system can be used for multiple purposes, such as investigating how a trained model or an entire system tolerate different kinds of faults in its input data; studying which model and hyperparameterization are the best when input data has certain kinds of faults or how alternative data cleaning approaches influence the operation of the resulting model; evaluating tradeoffs between model accuracy versus model robustness; and quantifying the accuracy difference when the model is trained with clean or faulty data. At present, dpEmu still is a prototype and as usual, it is not perfect in terms of functionality and usability. However, it already acts as demonstrator regarding how robustness and tolerance to data faults can be integrated into the development pipeline of ML systems. Popular ML libraries, such as Sklearn or TensorFlow, have extensive collections of functions for evaluating the models as well as splitting the datasets for training and testing parts. However, none of these, seem to have built-in support for studying the model behavior in case of erroneous input data. DpEmu can be used together with these libraries to add one step before the actual training of the model. The addition of one more step, however, will increase the training effort a lot. In addition to the actual training, it is possible to have another training loop that searches for the best possible architecture and hyperparameterization for the system [13].
Modified on	03/02/2022 15:37
Name	Software Framework for Data Fault Injection to Test Machine Learning Systems
Number of Coding References	6
Number of Codes Coding	3
Coverage	9.28%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	Data-intensive systems are sensitive to the quality of data. Data often has problems due to faulty sensors or network problems, for instance. In this work, we develop a software framework to emulate faults in data and use it to study how machine learning (ML) systems work when the data has problems. We aim for flexibility: users can use predefined or their own dedicated fault models. Likewise, different kind of data (e.g. text, time series, video) can be used and the system under test can vary from a single ML model to a complicated software system. Our goal is to show how data faults can be emulated and how that can be used in the study and development of ML
Modified on	03/02/2022 15:34

Name	Software Framework for Data Fault Injection to Test Machine Learning Systems
Number of Coding References	6
Number of Codes Coding	3
Coverage	9.28%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	we face questions that not only influence the testing phase but also the development decisions. Such questions include the following: – Should we train the system with perfect or with faulty data? Examples of faulty data are far less common than examples of correct data but we may still have a good understanding of the kinds of data faults the system will face over its lifetime. – Are some ML algorithms, architectures, or hyperparameter selections more robust towards data faults than others? – How trustworthy the results of the algorithms are when used in real-life settings, which include faulty input data? The difficulty of making a system deal with data faults comes from multiple sources. To begin with, data faults come in different forms. Some of them are systematic (e.g. sensor drift), whereas others are more random (e.g. a broken network connection). They happen infrequently so the training material there may not have many examples of faulty cases. Furthermore, it is not obvious what we should do to deal with faults – change the associated training data, change the model, or simply ignore the faulty output somehow. Unfortunately, testing how a system behaves with different kinds of data faults has been difficult.
Modified on	03/02/2022 15:35

Name	Software Framework for Data Fault Injection to Test Machine Learning Systems
Number of Coding References	6
Number of Codes Coding	3
Coverage	9.28%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Research gap in Maintaining ML systems
Created on	31/01/2022 14:42
Coded Text	we face questions that not only influence the testing phase but also the development decisions. Such questions include the following: – Should we train the system with perfect or with faulty data? Examples of faulty data are far less common than examples of correct data but we may still have a good understanding of the kinds of data faults the system will face over its lifetime. – Are some ML algorithms, architectures, or hyperparameter selections more robust towards data faults than others? – How trustworthy the results of the algorithms are when used in real-life settings, which include faulty input data? The difficulty of making a system deal with data faults comes from multiple sources. To begin with, data faults come in different forms. Some of them are systematic (e.g. sensor drift), whereas others are more random (e.g. a broken network connection). They happen infrequently so the training material there may not have many examples of faulty cases. Furthermore, it is not obvious what we should do to deal with faults – change the associated training data, change the model, or simply ignore the faulty output somehow. Unfortunately, testing how a system behaves with different kinds of data faults has been difficult.
Modified on	03/02/2022 15:35

Name	Software Logs for Machine Learning in a DevOps Environment
Number of Coding References	10
Number of Codes Coding	2
Coverage	11.63%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	04/02/2022 22:03
Coded Text	
Modified on	11/02/2022 14:29

Name	Software Logs for Machine Learning in a DevOps Environment
Number of Coding References	10
Number of Codes Coding	2
Coverage	11.63%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	04/02/2022 22:03
Coded Text	As presented in this section and in figure 4, generating logs for machine learning requires that engineers, R&D teams and the organization change the way log entries are generated. However, the process by which system logs for machine learning are generated is, in principle, no more difficult than adding a normal log statement. The main difference is the organizational alignment and agreement on the structure and semantics of log entries. As usual, although most of the attention is quickly drawn towards the technical framework, it is the introduction of new processes and activities that will require the most effort and attention. Especially early in the process of adopting the approach outlined in this paper, it is beneficial to add a new AI log statement in the code at every place that there is an existing log statement, leaving the existing log statement. In this way, it is possible to generate two separate logs: the original log and the new AI log. The information that is presented in the AI log statement should be an encoded/normalised version of what is presented in the human readable log. In the case that a machine learning algorithm finds an anomaly in the AI log, the link with the human understandable entry in the human-readable log significantly helps the investigation into the detected anomaly.
Modified on	11/02/2022 14:29

Name	Software Logs for Machine Learning in a DevOps Environment
Number of Coding References	10
Number of Codes Coding	2
Coverage	11.63%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	04/02/2022 22:03
Coded Text	DevOps scenarios, but not for autonomous system deployments. Finally, we need to govern evolution and backward compatibility in response to a constant flow of change requests from the R&D teams, customers, data scientists and others. The evolution of the system log entry model needs to be carefully managed as allowing for breaking changes may also invalidate data sets predating the change due to changes in semantics and/or structure. In the cases where introducing breaking changes is unavoidable, it may be necessary to develop mapping functions that allow for the generation of data sets that are based on system logs both before and after the breaking change. As virtually all machine learning algorithms perform better with a greater quantity of data, it is frequently beneficial to combine multiple logs into one data set for training and validation.
Modified on	11/02/2022 14:28

Name	Software Logs for Machine Learning in a DevOps Environment
Number of Coding References	10
Number of Codes Coding	2
Coverage	11.63%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	04/02/2022 22:03
Coded Text	In this paper, we present the main challenges of contemporary approaches to generating, storing and managing the evolution of system logs data for large, complex, software-intensive systems based on an in-depth case study at a world-leading telecommunications company. Second, we present an approach for generating and managing the evolution of log data in a DevOps environment that does not suffer from the aforementioned challenges and is optimized for use in machine learning. Third, we provide validation of the approach based on expert interviews that confirm that the approach addresses the identified challenges and problems.
Modified on	11/02/2022 14:26

Name	Software Logs for Machine Learning in a DevOps Environment
Number of Coding References	10
Number of Codes Coding	2
Coverage	11.63%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	04/02/2022 22:03
Coded Text	IV. SYSTEM LOGS FOR MACHINE LEARNING To address the challenges of using system logs for ML, we have developed a novel approach consisting of three main parts. First, we discuss the DevOps scenario that logs optimized for ML could be applied to and the success factors which would emerge in it. Second, we propose the technical realization. Finally, we present the required process changes for realizing the proposed approach in an industrial context. A. Logging in a DevOps Environment Based on our research at the case study company, as well as experience from other companies, we identified three distinct
Modified on	11/02/2022 14:28

Name	Software Logs for Machine Learning in a DevOps Environment
Number of Coding References	10
Number of Codes Coding	2
Coverage	11.63%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	04/02/2022 21:53
Coded Text	
Modified on	11/02/2022 14:27

Name	Software Logs for Machine Learning in a DevOps Environment
Number of Coding References	10
Number of Codes Coding	2
Coverage	11.63%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	04/02/2022 21:53
Coded Text	Although logging may seem trivial, in practice most R&D teams aim to manually observe the functionality of their systems. This leads to a high degree of variance in log generation, multiple log files for generating log entries of different types, and, in a DevOps environment, continuous and unmanaged changes to internal logging practices. For ML, this can greatly complicate processing the data and hinder the training process. The traditional way of generating system logs is shown in figure 1. In this case, the R&D teams have full freedom to generate logs to optimally support their needs. These approaches often have developed over time to optimally support developers. The challenge is that when the same logs are used for machine learning, the data science team is required to spend significant effort on pre-processing, the pre-processed data then used to manually (re)train the model which is, subsequently, manually deployed. As part of our case study research at the primary and secondary case companies as well as based on the literature that we reviewed and reported in section II, we have identified eight significant challenges associated with applying ML to system logs.
Modified on	11/02/2022 14:27

Name	Software Logs for Machine Learning in a DevOps Environment
Number of Coding References	10
Number of Codes Coding	2
Coverage	11.63%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	04/02/2022 21:53
Coded Text	In a DevOps environment, especially, unmanaged evolution in log data structure causes frequent disruption of operations in automated data pipelines, dashboards and analytics.
Modified on	11/02/2022 14:26

Name	Software Logs for Machine Learning in a DevOps Environment
Number of Coding References	10
Number of Codes Coding	2
Coverage	11.63%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	04/02/2022 21:53
Coded Text	Our research shows that source code is often not available for analysis and many logs are highly unstructured and consequently difficult to parse. The problems concerning access to source code [9] and lack of structure [4] have been acknowledged in research and automated log parsers, such as MoLFI [9], Drain [7], and Spell [5], have been developed to some success. However, even the state-of-the-art log parser, Drain [7], struggles with state identification and dealing with log messages of variable length, which leads to varying and unpredictable performance based on the type of log [18]. In practice, many companies have logs with less standardization and automated log parsing does not provide the desired results. Finally, some logging standards have been proposed but these are often domain specific or insufficient. For example, the XES standard [6] is simple to parse, but the transformation
Modified on	11/02/2022 14:27

Name	Software Logs for Machine Learning in a DevOps Environment
Number of Coding References	10
Number of Codes Coding	2
Coverage	11.63%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	04/02/2022 21:53
Coded Text	System logs perform a critical function in software- intensive systems as logs record the state of the system and significant events in the system at important points in time. Unfortunately, log entries are typically created in an ad-hoc, unstructured and uncoordinated fashion, limiting their usefulness for analytics and machine learning.
Modified on	11/02/2022 14:25

Name	Software Quality for AI~ Where We Are Now~
Number of Coding References	7
Number of Codes Coding	3
Coverage	14.62%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:24
Coded Text	Murphy et al. [22] proposed a testing framework for Machine Learning (ML), introducing the concept of regression testing and an approach for ranking the correctness of new versions of ML algorithms.
Modified on	03/02/2022 15:31

Name	Software Quality for AI~ Where We Are Now~
Number of Coding References	7
Number of Codes Coding	3
Coverage	14.62%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:27
Coded Text	
Modified on	03/02/2022 15:33

Name	Software Quality for AI~ Where We Are Now~
Number of Coding References	7
Number of Codes Coding	3
Coverage	14.62%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:27
Coded Text	
Modified on	03/02/2022 15:33

Name	Software Quality for AI~ Where We Are Now~
Number of Coding References	7
Number of Codes Coding	3
Coverage	14.62%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:27

Coded Text	AI Software Quality: Key Issues and Comments Based on the collective experience of our groups and through simple selfethnography [9], we elicited different code quality issues commonly faced by all sorts of stakeholders (e.g., our research assistant working for consultancy projects in AI, our colleagues, and our students as developers of AI Software) working with AI software. In this Section, we describe the aforementioned elicitation, also discussing possible solutions that might be adopted to solve the emerged issues. – Developers Skills and Training. Once the suitable data has been chosen and proven to be compliant with our requirements the next step involves coding. The machine learning engineer profession was born less than a decade ago and therefore, no training guidelines have been outlined yet. Most of the professionals that occupy this position have been moving from a similar field such as mathematics, physics, or computer vision. This grouping of different backgrounds generated “communication problems” which reflected in the way the code was written. We identify four open problems nested in this macro area: code understandability, code quality guidelines, training problems
Modified on	03/02/2022 15:32

Name	Software Quality for AI~ Where We Are Now~
Number of Coding References	7
Number of Codes Coding	3
Coverage	14.62%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:27
Coded Text	However, a large number of Artificial Intelligence applications are produced by developers without proper training on software quality practices or processes, and in general, lack in-depth knowledge regarding software engineering processes. The main reason is due to the fact that the machine-learning engineer profession has been born very recently, and currently there is a very limited number of training or guidelines on issues (such as code quality or testing) for machine learning and applications using machine learning code. In this work, we aim at highlighting the main software quality issues of Artificial Intelligence systems, with a central focus on machine learning code, based on the experience of our four research groups. Moreover, we aim at defining a shared research road map, that we would like to discuss and to follow in collaboration with the workshop participants.
Modified on	03/02/2022 15:30

Name	Software Quality for AI~ Where We Are Now~
Number of Coding References	7
Number of Codes Coding	3
Coverage	14.62%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:27
Coded Text	The need for considering the quality of AI-enabled systems was highlighted already even more than 30 years ago [26,31]. For the time being, different approaches have been proposed to evaluate the quality of the AI-models but little in the way of AI code quality itself (e.g. [15] and [8]). Conversely, as already mentioned, the overall quality of the AI-enabled sys- tems, and in particular the ML code has never been investigated systematically so far if not anecdotally. For example, a report from Informatics Europe1 and the ACM Europe Council2, as well as the Networked European Software and Services Initiative3, highlighted the importance of assessing the quality of AI-related
Modified on	03/02/2022 15:30

Name	Software Quality for AI~ Where We Are Now~
Number of Coding References	7
Number of Codes Coding	3
Coverage	14.62%
Folder Location	Codes\\Snowballing
Parent Name	Codes\\Snowballing\\Snowballing
Created on	21/06/2022 14:00
Coded Text	Murphy et al. [22] proposed a testing framework for Machine Learning (ML), introducing the concept of regression testing and an approach for ranking the correctness of new versions of ML algorithms. Besides the proposed model, they also highlighted conflicting technical terms with very different meanings to machine learning experts than they do to software engineers (e.g. “testing”, “regression”, “validation”, “model”). Moreover, they raised the problem of code quality, reporting that future works should address it. Related to the matter of ML testing, Zhang et al. provided a comprehensive survey research [30]. In this work with the term ML testing, they refer to “any activity aimed at detecting differences between existing and required behaviors of machine learning systems.” The work comprehends a section related to fundamental terminology in ML which will be referred to in Table 1. Nakajima, in his invited talk, call the attention the product quality of ML- based systems, identifying new challenges for quality assurance methods. He proposed to identify new testing methods for ML-based systems, proposing to adopt Metamorphic testing [10] is a pseudo oracle approach and uses golden outputs as testing values. Pimentel et al. [10] investigated the reproducibility of Jupyter notebooks, showing that less than 50% of notebooks are reproducible, opening new questions to our community to propose advanced approaches for analyzing Jupyter notebooks. Wang et al. [29] analyzed 2.5 Million of Jupiter notebooks investigating their code quality reporting that notebooks are inundated with poor quality code, e.g., not respecting recommended coding practices, or containing unused variables and deprecated functions. They also report that poor coding practices, as well as the lack of quality control, might be propagated into the next generation of developers. Hence, they argue that there is a strong need to identify quality practices, but especially a quality culture in the ML community. The vast majority of grey literature also focuses on the quality of the ML models or on the data. Only a limited number of authors raised the problem of the overall product quality or of the quality of the ML code. Vignesh [24] proposed to continuously validate the quality of ML models considering black boxes techniques and evaluating the performance of model post-deployment on test data sets and new data from production scenarios. He also proposes to adopt metamorphic testing, involving data mutation to augment test data sets used for evaluating model performance and accuracy. It is interesting to note that Vignesh recommends exposing models being tested as RESTful service, instead of testing internally or manually. As for the quality of the code of ML Models, Dhaval [20] proposes to introduce code review processes for ML developers, adopting code reviewing techniques traditionally adopted in SW Engineering. Besides the model itself, the essence of a good machine learning-based soft- ware relies on the data used to train the network. It is therefore vital to take into account the characteristics and features that mark a specific application and
Modified on	22/06/2022 10:39

Name	Sources of Risk of AI Systems
Number of Coding References	13
Number of Codes Coding	2
Coverage	5.28%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	3.4. Complexity ofthe Intended Task and Usage Environment AI is sensibly used for tasks for which there are no classic technologies as alternatives. Such tasks are usually characterised by a high degree of complexity. This complexity can arise from the task itself, as is the case, for example, with the classification of people, or from the complexity of the environment in which it is used, as is the case, for example, in the field of self-driving vehicles. Often, both apply evenly, which makes the task even more difficult. This complexity gives rise to a certain level of uncertainty in the system's behaviour, which is often perceived as non-deterministic, but whose cause lies in the fact that a complex task and/or environment can only be analysed and described completely by a human with great difficulty. This is mainly due to the large state space of such an environment, which can also be subject to constant change, which, in turn, continuously enlarges the state space, whereby it can be assumed that even a model that generalises the state space very well will not react appropriately to every possible state of the environment.
Modified on	16/02/2023 09:49

Name	Sources of Risk of AI Systems
Number of Coding References	13
Number of Codes Coding	2
Coverage	5.28%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	3.5. Degree ofTransparency and Explainability Often, aspects of traceability, explainability, reproducibility and general transparency are summarised under the term "transparency". However, these terms must be clearly distinguished from one another. Transparency is the characteristic of a system that describes the degree to which appropriate information about the system is communicated to relevant stakeholders, whereas explainability describes the property of an AI system to express important factors influencing the results of the AI system in a way that is understandable for humans. For this reason, the transparency of a system is often considered a prerequisite for an explainable AI system. Even if this statement is not entirely correct, in relation to existing model-agnostic methods for increasing the explainability of neural networks, for example, a high degree of transparency nevertheless has a positive effect on the explainability of an AI system.
Modified on	16/02/2023 09:49

Name	Sources of Risk of AI Systems
Number of Coding References	13
Number of Codes Coding	2
Coverage	5.28%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	3.8. Technological Maturity The technological maturity level describes how mature and error-free a certain tech- nology is in a certain application context. If new technologies with a lower level of maturity are used in the development of the AI system, they may contain risks that are still unknown or difficult to assess. Mature technologies, on the other hand, usually have a greater variety of empirical data available, which means that risks can be identified and assessed more easily. However, with mature technologies, there is a risk that risk awareness de-
Modified on	16/02/2023 09:49

Name	Sources of Risk of AI Systems
Number of Coding References	13
Number of Codes Coding	2
Coverage	5.28%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	cording to the “Ethics guidelines for trustworthy AI” of the AI HILEG, trustworthy AI comprises three components, entailing that the actors and processes involved in AI systems (including their development, deployment and use) should be: I. II. Lawful—complying with all applicable laws and regulations; Ethical—ensuring adherence to ethical principles and values; III. Robust—both from a technical and social perspective. As mentioned, this work deals with purely technical sources of risk or those sources of risk that entail a technical implementation and not with legal issues. On this basis, the taxonomy of different sources of risk that can influence the trustworthiness of an AI system presented in Figure 1 was drawn up.
Modified on	16/02/2023 09:48

Name	Sources of Risk of AI Systems
Number of Coding References	13
Number of Codes Coding	2
Coverage	5.28%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	Degree ofAutomation and Control The degree of automation and control describes the extent to which an AI system functions independently of human supervision and control. It thus determines not only how much information about the tactile behaviour of the system is available to the operator, but also defines the control and intervention options of the human. On the one hand, an assessment is made with regard to how high the degree of automation must be for the respective application, but on the other hand, an assessment is also made with regard to whether the human is adequately supported by the AI application and is given appropriate room for manoeuvring in interactions with the AI application. Systems with a high degree of automation may exhibit unexpected behaviour that can be
Modified on	16/02/2023 09:48

Name	Sources of Risk of AI Systems
Number of Coding References	13
Number of Codes Coding	2
Coverage	5.28%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51

Coded Text	Fairness The general principle of equal treatment requires that an AI system upholds the principle of fairness, both ethically and legally. This means that the same facts are treated equally for each person unless there is an objective justification for unequal treatment. AI systems used for automated decision-making pose a particular risk for the unfair treatment of specific persons or groups of persons.
Modified on	16/02/2023 09:48

Name	Sources of Risk of AI Systems
Number of Coding References	13
Number of Codes Coding	2
Coverage	5.28%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	In order to be able to classify the identified sources of risk in a taxonomy, the com- ponents of a trustworthy AI, according to the High-Level Expert Group on Artificial Intelligence (AI HLEG) of the European Commission, should be used as a guideline. Ac-
Modified on	16/02/2023 09:47

Name	Sources of Risk of AI Systems
Number of Coding References	13
Number of Codes Coding	2
Coverage	5.28%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	Privacy Privacy is related to the ability of individuals to control or influence what information related to them may be collected and stored and by whom that information may be disclosed. Due to their characteristics, it is possible for AI applications to interfere with a variety of legal positions. Often, these are encroachments on privacy or the right to informational self-determination. Many AI methods process a variety of different data. Machine learning methods and deep learning methods are especially dependent on large amounts of data, as they need sufficient data to train their models. Ultimately, their accuracy often correlates with the amount of data used. The misuse or disclosure of some data, particularly personal and sensitive data (e.g., health records), could have harmful effects on data subjects. For example, AI applications often process sensitive information, such as personal or private data, including voice recordings, images or videos. Therefore, it must be ensured that the respective local data protection regulations, such as the General Data Protection Regulation (GDPR) [66] in Europe, are observed and complied with.
Modified on	16/02/2023 09:48

Name	Sources of Risk of AI Systems
Number of Coding References	13
Number of Codes Coding	2
Coverage	5.28%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	Security To assess the trustworthiness of an AI-based system, traditional IT security require- ments also need to be considered. ISO/IEC 27001 [119], ISO/IEC 18045 [120] and ISO/IEC 62443 [121] already provide processes for the audit and certification of horizontal IT security requirements that are also applicable to AI-based systems. In addition to following the best practices and observing existing standards for con- ventional systems, artificial intelligence comes with an intrinsic set of challenges that need to be considered when discussing trustworthiness, especially in the context of functional safety. AI models, especially those with higher complexities (such as neural networks), can exhibit specific weaknesses not found in other types of systems and must, therefore, be subjected to higher levels of scrutiny, especially when deployed in a safety-critical context. One class of attacks on AI systems in particular has recently garnered interest: adver- sarial machine learning. Here, an attacker tries to manipulate an AI model to either cause it to malfunction, change the expected model output or obtain information about the model that would otherwise not be available to them.
Modified on	16/02/2023 09:49

Name	Sources of Risk of AI Systems
Number of Coding References	13
Number of Codes Coding	2
Coverage	5.28%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	These can be roughly divided into two different blocks. The first block deals with ethical aspects. These include fairness, privacy and the degree of automation and control. The second block deals with various aspects that can influence the reliability and robustness of the AI system, and thus have a direct influence on the safety of the system. Generally, robustness relates to the ability of a system to maintain its level of performance under any circumstances of its usage [4]. Robustness differs from reliability in that a reliable system only needs to maintain its level of performance under the specified conditions for a specific period of time [4]. Robustness, on the other hand, also includes stability against bias or errors and, therefore, represents an extension of the concept of reliability. In the case of AI, robustness properties demonstrate the ability of the system to maintain the same level of performance when using new data as it achieves when using the data with which it was trained or data for typical operations. Robustness is a new challenge in the context of AI systems, as these systems are used for very complex tasks in complex usage environments, which involve a certain degree of uncertainty. Neural network architectures represent a particularly difficult challenge, as they are both hard to explain and sometimes have unexpected behaviour due to their nonlinear nature. Furthermore, some machine learning methods offer new attack vectors that can reduce the security of the system against external attacks. It is also important to consider the multiple influences of hardware failures, as well as the specific aspects related to them, which can also have a negative effect. Finally, the technological maturity of the AI method used is another important aspect to consider. Details of the above properties and risk factors, along with their related aspects and challenges, are discussed below.
Modified on	16/02/2023 09:48

Name	Sources of Risk of AI Systems
Number of Coding References	13
Number of Codes Coding	2
Coverage	5.28%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	Artificial intelligence can be used to realise new types of protective devices and assistance systems, so their importance for occupational safety and health is continuously increasing. However, established risk mitigation measures in software development are only partially suitable for applications in AI systems, which only create new sources of risk. Risk management for systems that for systems using AI must therefore be adapted to the new problems.
Modified on	16/02/2023 09:51

Name	Sources of Risk of AI Systems
Number of Coding References	13
Number of Codes Coding	2
Coverage	5.28%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	Conclusions Artificial intelligence is still a very agile field of research that is making great progress. Essentially, we envisage three pillars derived from the rapid progress in the field of artificial intelligence within the last few years: The objective for this was the technical but also economic availability of a high computing power, which made it possible to significantly reduce the training times for deep neural networks [67,141,142]. The second pillar is the availability of large amounts of data, which makes the meaningful training of these deep neural networks possible [47,67–69], and the third pillar is the spread of the open-source idea [143–145]. This has not only made new methods, but also entire training algorithms or even complete models, quickly accessible to a broader public audience, which can thus be easily taken up by other working groups and directly used or further optimised. All of these factors are still in place, which means that it can be expected that major advances will continue to be made in this field, resulting in the continued rapid market growth in artificial intelligence. Even though this technology has already established itself permanently on the market in some areas, the fields of application of artificial intelligence will be expanded in the future through the realisation of new innovative applications. However, care must be taken to ensure that a human-centred approach is always adopted in the development of such systems. For this, compliance with basic safety principles is essential and must fulfil all the framework conditions for trustworthy AI. This requires a precise understanding of the specific aspects of the individual artificial intelligence processes and their impact on the overall quality of the system in general, as well as its safety. In particular, AI systems based on machine learning present new challenges for the security integrity of the system. Since their models are not developed directly based on the interpretation of a specification by human developers, but are indirectly derived from data, major difficulties exist, especially in creating the specification. Ashmore et al. (Ashmore, 2019) derived the risk source of an incomplete specification from this. Other works also name these and point out the problem of interpretability [146–148]. However, it can be stated here that the problem of incomplete specification is a consequence of the complexity of the task and operational environment of AI systems, which can thus be regarded as the actual original source of risk. Furthermore, the term interpretability is not defined in the basic standard for AI terminology [4], which instead defines the concept of explainability, also being reflected in the scientific discipline of XAI (explainable AI).
Modified on	16/02/2023 09:49

Name	Sources of Risk of AI Systems
Number of Coding References	13
Number of Codes Coding	2
Coverage	5.28%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	Protective devices and control systems based on artificial intelligence have already enabled fully automated vehicles and robots to be created [8,9]. Furthermore, they enable accidents to be prevented by assistance systems capable of recognising hazardous situations [10,11]. However, for the benefit of human safety and health, safe and trustworthy artificial intelligence is required. This is because, despite the rapid, positive progression of this technology and the new prospects for occupational safety, the increasing application of this technology will also produce new risks [12]. Even today, we already face an increasing number of accidents in systems that utilise artificial intelligence [13], including various reports on fatal accidents due to AI-related failures in automated vehicles [14,15]. Established measures of risk reduction in the development of software are limited in their ability to mitigate these risks, and existing safety standards are hardly applicable to AI systems as they do not take into account their technical peculiarities [16]. For example, during verification and validation activities in the software life cycle, the influences of different input values in the system are investigated, but these can be relatively easily mapped by boundary value analyses. In the field of artificial intelligence, however, this is difficult due to the extensive and complex possible state space. These applications have to deal with the influence of many different biases [17], some of which are specific to AI systems and are therefore not considered in the required verification and validation activities of existing software safety standards. For this reason, the development of safe AI systems requires a good understanding of the components of trustworthy artificial intelligence [18,19] as risk management for systems that utilise AI must be carefully adapted to the new problems associated with this technology. Some research proposes assurance cases to support the quality assurance and certi- fication of AI applications. These must provide assessable and structured arguments to achieve a certain quality standard [20–22]. However, these works lack a detailed list of concrete criteria. We propose to define these criteria based on the sources of risk for AI systems. These can then be analysed and evaluated within the risk assessment to derive appropriate risk mitigation measures; this proves to be necessary.
Modified on	16/02/2023 09:51

Name	Sources of Risk of AI Systems Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Studying the Practices of Deploying Machine Learning Projects on Docker
Number of Coding References	9
Number of Codes Coding	1
Coverage	5.36%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	• Logging and Monitoring: This category is about using Docker to identify suspicious activities in any privileged operating of MLbased software projects through logging and monitoring. Logging includes tracking and storing records related to the evens, data input, processes, data output, and final results of the running MLbased software projects. On the other hand, monitoring is a diagnostic tool used for alerting ML engineers (visualization) of the
Modified on	18/02/2023 16:02

Name	Studying the Practices of Deploying Machine Learning Projects on Docker
Number of Coding References	9
Number of Codes Coding	1
Coverage	5.36%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	based on their domains and types, into six (6) different categories of ‘Application System’ (42%), ‘AIOps’ (23%), ‘ToolKit’ (16%), ‘DL Frameworks’ (15%), ‘Models’ (13%), and ML-based Tutorials/ Documentation (1%).
Modified on	18/02/2023 16:02

Name	Studying the Practices of Deploying Machine Learning Projects on Docker
Number of Coding References	9
Number of Codes Coding	1
Coverage	5.36%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	Composition ofBase Image As mentioned in Section 2, the base image is the initial point to understand what the project is using Docker for. This SubSection reports the base images and the base image types used to build the Docker images for deploying ML-based software projects. Figure 4a present the composition of the most used base images extracted from the Dockerfiles of the selected ML software projects.
Modified on	18/02/2023 16:04

Name	Studying the Practices of Deploying Machine Learning Projects on Docker
Number of Coding References	9
Number of Codes Coding	1
Coverage	5.36%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	Composition ofthe File in layers and their size Here we are interested in characterizing the files contained in the image layers in terms of their size and types. As the initial step to understanding the contained files in the layers, we observed that only a small percentage of file types are the most used files within the image layers by analyzing the whole dataset. For example, in the MLOps/ AIOps category, only 3% (135) different file types (e.g., files related to script/ source code, Executable, Object code, Libraries including the ELF/COFF, or files associated with Python local packages storage (site-packages directory)) take up 98% as the most occurring types of files.
Modified on	18/02/2023 16:04

Name	Studying the Practices of Deploying Machine Learning Projects on Docker
Number of Coding References	9
Number of Codes Coding	1
Coverage	5.36%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	Docker and its related container technologies have become a prominent solution for automating the deployment ofmodern software systems. This is true due to its numerous desirable features such as isolation, low overhead, and efficient packaging of the runtime environment. Docker container management framework consists of images containing applications and their required runtime dependencies [24] and can be easily versioned, stored, and shared via centralized registry services (e.g., Docker Hub1). Researchers [4, 8, 16, 22, 23, 25] have extensively studied the use of Docker for the deployment process of general software systems. In contrast, we could not find any study focusing on understanding how Docker is being used to deploy machine learning based (ML-based) projects (i.e., Projects using machine learning). This information could help the software engineering community understand the emerging practice of deploying ML applications using containers and identify aspects that should be improved.
Modified on	18/02/2023 16:00

Name	Studying the Practices of Deploying Machine Learning Projects on Docker
Number of Coding References	9
Number of Codes Coding	1
Coverage	5.36%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	Docker is a containerization service that allows for convenient deployment ofwebsites, databases, applications' APIs, and machine learning (ML) models with a few lines ofcode. Studies have recently explored the use of Docker for deploying general software projects with no specific focus on how Docker is used to deploy ML-based projects. In this study, we conducted an exploratory study to understand how Docker is being used to deploy ML-based projects. As the initial step, we examined the categories of ML-based projects that use Docker. We then examined why and how these projects use Docker, and the characteristics of the resulting Docker images. Our results indicate that six categories of ML-based projects use Docker for deployment, including ML Applications, MLOps/ AIOps, Toolkits, DL Frameworks, Models, and Documentation.
Modified on	18/02/2023 16:00

Name	Studying the Practices of Deploying Machine Learning Projects on Docker
Number of Coding References	9
Number of Codes Coding	1
Coverage	5.36%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	related issues by analyzing metrics. Usually, Logging and monitoring help ensure application availability and assess the impact of state transformations on performance. • Cloud-based Development: This category is about using Docker to automate the process of setting the software development environment on the server-side (i.e., accessible through the browser) connected to cloud-based infrastructure (e.g., CI/CD and versioncontrolled system [9, 14, 19, 21]) and other services such as a database. More specifically Docker is used in the setting up of the workspace that is ready-to-code where all the dependencies needed by the source code are compiled, running build tools and automated testing (e.g., on git push), and live sharing of code. • Interactive Development: This category encompasses the use of Docker for deploying and distributing the interactive development environment such as Jupiter notebook or RStudio that allows data scientists or ML Engineers to create and share documents that integrate live code, equations, visualizations, computational output, and other multimedia resources. • Model management: In this category, Docker is used to handle the different activities related to the trained ML algorithms (ML model) that can generate predictions using patterns in the input data. This process includes managing data flow to ML models, working with multiple models, and collecting and analyzing metrics throughout the life cycle ofmodels. For example, in Figure 2 we highlight the different scenarios where Docker is used when working with ML models in the studied ML-based software projects, such as exposing the API for prediction/ inference, using Docker for Model training/ testing, Model export, model evaluation, and model storage. • Manage Users: In this category, the ML engineers use Docker to create and manage users, such as granting varying levels of access based on requirements. • CI/CD: This category combined using Docker to automate the continuous integration and delivery or deployment of ML-based software projects, usually to bridge the gaps between development, teams, and operation activities by enforcing automation in building, testing, and deploying ML-based software projects.
Modified on	18/02/2023 16:02

Name	Studying the Practices of Deploying Machine Learning Projects on Docker
Number of Coding References	9
Number of Codes Coding	1
Coverage	5.36%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	This subsection reports our discovered purposes of using Docker in ML-based software projects. Figure 2 presents 21 high-level categories of the purposes of using Docker (in light grey color) observed in the studied ML-based software projects (identified in Step 5o of our analysis methodology Section 3). The purposes reported on the left side of Figure 2 are more general, while those on the right are more related to ML components. In the following, we describe some of the categories and sub-categories of the obtained taxonomy in details, highlighting the examples.
Modified on	18/02/2023 16:02

Name	Studying the Practices of Deploying Machine Learning Projects on Docker
Number of Coding References	9
Number of Codes Coding	1
Coverage	5.36%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	What kind ofML-based software projects use Docker? This question aims to understand the types ofML-based software projects that use Docker in their deployment process. This information will help us understand ifDocker is only being adopted by some specific ML-based software projects or by ML-based software projects in general. Through manual analysis, we grouped the studied ML-based software projects
Modified on	18/02/2023 16:02
Name	Studying the Practices of Deploying Machine Learning Projects on Docker Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Switch Transformers~ Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
Number of Coding References	11
Number of Codes Coding	1
Coverage	9.37%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	4.8 5.0 5.2 5.4 5.6 5.8 6.0 2e 4e 8e 16e 32e 64e 128e 256e 1 09 Sparse Model Parameters 1 01 0 2.0 1.9 1.8 1.7 1.6 1.5 1.4 1.3 1.2 0 Switch-Base: 128e Switch-Base: 64e Switch-Base: 32e Switch-Base: 16e T5-Base 1 2 Training Step 3 4 1e5 Figure 1: Scaling and sample efficiency of Switch Transformers. Left Plot: Scaling properties for increasingly sparse (more experts) Switch Transformers. Right Plot: Negative log perplexity comparing Switch Transformers to T5 (Raffel et al., 2019) models using the same compute budget. Sparse training is an active area of research and engineering (Gray et al., 2017; Gale et al., 2020), but as of today, machine learning libraries and hardware accelerators still cater to dense matrix multiplications. To have an efficient sparse algorithm, we start with the Mixture-of-Expert (MoE) paradigm (Jacobs et al., 1991; Jordan and Jacobs, 1994; Shazeer et al., 2017), and simplify it to yield training stability and computational benefits. MoE models have had notable successes in machine translation (Shazeer et al., 2017, 2018; Lepikhin et al., 2020), however, widespread adoption is hindered by complexity, communication costs, and training instabilities. We address these issues, and then go beyond translation, to find that these class of algorithms are broadly valuable in natural language. We measure superior scaling on a diverse set of natural language tasks and across three regimes in NLP: pre-training, finetuning and multi-task training. While this work focuses on scale, we also show that the Switch Transformer architecture not only excels in the domain of supercomputers, but is 3 Test Loss Neg Log Perplexity Fedus, Zoph and Shazeer beneficial even with only a few computational cores. Further, our large sparse models can be distilled (Hinton et al., 2015) into small dense versions while preserving 30% of the sparse model quality gain. Our contributions are the following: • The Switch Transformer architecture, which simplifies and improves over Mixture of Experts. • Scaling properties and a benchmark against the strongly tuned T5 model (Raffel et al., 2019) where we measure 7x+ pre-training speedups while still using the same FLOPS per token. We further show the improvements hold even with limited computational resources, using as few as two experts. • Successful distillation of sparse pre-trained and specialized fine-tuned models into small dense models. We reduce the model size by up to 99% while preserving 30% of the quality gains of the large sparse teacher. • Improved pre-training and fine-tuning techniques: (1) selective precision training that enables training with lower bfloat16 precision (2) an initialization scheme that allows for scaling to a larger number of experts and (3) increased expert regularization that improves sparse model fine-tuning and multi-task training. • A measurement of the pre-training benefits on multilingual data where we find a universal improvement across all 101 languages and with 91% of languages benefiting from 4x+ speedups over the mT5 baseline (Xue et al., 2020). • An increase in the scale of neural language models achieved by efficiently combining data, model, and expert-parallelism to create models with up to a trillion parameters. These models improve the pre-training speed of a strongly tuned T5-XXL baseline by 4x. 2. Switch Transformer The guiding design principle for Switch Transformers is to maximize the parameter count of a Transformer model (Vaswani et al., 2017) in a simple and computationally efficient way. The benefit of scale was exhaustively studied in Kaplan et al. (2020) which uncovered powerlaw scaling with model size, data

	set size and computational budget. Importantly, this work advocates training large models on relatively small amounts of data as the computationally optimal approach. Heeding these results, we investigate a fourth axis: increase the parameter count while keeping the floating point operations (FLOPs) per example constant. Our hypothesis is that the parameter count, independent of total computation performed, is a separately important axis on which to scale. We achieve this by designing a sparsely activated model that efficiently uses hardware designed for dense matrix multiplications such as GPUs and TPUs. Our work here focuses on TPU architectures, but these class of models may be similarly trained on GPU clusters. In our distributed training setup, our sparsely activated layers split unique weights on different devices. Therefore, the weights of the model increase with the number of devices
Modified on	28/02/2023 14:38
Name	Switch Transformers~ Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
Number of Coding References	11
Number of Codes Coding	1
Coverage	9.37%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Data Parallelism When training data parallel models, which is the standard for distributed training, then all cores are allocated to the data-parallel dimension or n = N,m = 1. This has the advantage that no communication is needed until the entire forward and backward pass is finished and the gradients need to be then aggregated across all cores. This corresponds to the left-most column of Figure 9. 5.2 Model Parallelism We now consider a scenario where all cores are allocated exclusively to the model-parallel dimension and so n = 1,m = N. Now all cores must keep the full B tokens and each core will contain a unique slice of the weights. For each forward and backward pass, a communication cost is now incurred. Each core sends a tensor of [B, dmodel] to compute the second matrix multiplication ReLU(h)Wout because the dff dimension is partitioned and must be summed over. As a general rule, whenever a dimension that is partitioned across cores must be summed, then an all-reduce operation is added for both the forward and backward pass. This contrasts with pure data parallelism where an all-reduce only occurs at the end of the entire forward and backward pass.
Modified on	28/02/2023 14:42

Name	Switch Transformers~ Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
Number of Coding References	11
Number of Codes Coding	1
Coverage	9.37%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Discussion We pose and discuss questions about the Switch Transformer, and sparse expert models generally, where sparsity refers to weights, not on attention patterns. Isn't Switch Transformer better due to sheer parameter count? Yes, and by design! Parameters, independent of the total FLOPs used, are a useful axis to scale neural language models. Large models have been exhaustively shown to perform better (Kaplan et al., 2020). But in this case, our model is more sample efficient and faster while using the same computational resources. I don't have access to a supercomputer—is this still useful for me? Though this work has focused on extremely large models, we also find that models with as few as two experts improves performance while easily fitting within memory constraints of commonly available GPUs or TPUs (details in Appendix D). We therefore believe our techniques are useful in small-scale settings. Do sparse models outperform dense models on the speed-accuracy Pareto curve? Yes. Across a wide variety of different models sizes, sparse models outperform dense models per step and on wall clock time. Our controlled experiments show for a fixed amount of computation and time, sparse models outperform dense models. I can't deploy a trillion parameter model—can we shrink these models? We cannot fully preserve the model quality, but compression rates of 10 to 100x are achievable by distilling our sparse models into dense models while achieving ≈30% of the quality gain of the expert model. Why use Switch Transformer instead of a model-parallel dense model? On a time basis, Switch Transformers can be far more efficient than dense-models with sharded parameters (Figure 6). Also, we point out that this decision is not mutually exclusive—we 25 Fedus, Zoph and Shazeer can, and do, use model-parallelism in Switch Transformers, increasing the FLOPs per token, but incurring the slowdown of conventional model-parallelism. Why aren't sparse models widely used already? The motivation to try sparse models has been stymied by the massive success of scaling dense models (the success of which is partially driven by co-adaptation with deep learning hardware as argued in Hooker (2020)). Further, sparse models have been subject to multiple issues including (1) model complexity, (2) training difficulties, and (3) communication costs. Switch Transformer makes strides to alleviate these issues.
Modified on	28/02/2023 14:42

Name	Switch Transformers~ Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
Number of Coding References	11
Number of Codes Coding	1
Coverage	9.37%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Efficient Sparse Routing We use Mesh-Tensorflow (MTF) (Shazeer et al., 2018) which is a library, with similar semantics and API to Tensorflow (Abadi et al., 2016) that facilitates efficient distributed data and model parallel architectures. It does so by abstracting the physical set of cores to a logical mesh of processors. Tensors and computations may then be sharded per named dimensions, facilitating easy partitioning of models across dimensions. We design our model with TPUs in mind, which require statically declared sizes. Below we describe our distributed Switch Transformer implementation. 3. See Section 2.2 for a technical description. 6 Switch Transformers Distributed Switch Implementation. All of our tensor shapes are statically deter- mined at compilation time, but our computation is dynamic due to the routing decisions at training and inference. Because of this, one important technical consideration is how to set the expert capacity. The expert capacity—the number of tokens each expert computes—is set by evenly dividing the number of tokens in the batch across the number of experts, and then further expanding by a capacity factor, expert capacity = < tokens per batch number of experts > × capacity factor. (3) A capacity factor greater than 1.0 creates additional buffer to accommodate for when tokens are not perfectly balanced across experts. If too many tokens are routed to an expert (referred to later as dropped tokens), computation is skipped and the token representation is passed directly to the next layer through the residual connection. Increasing the expert capacity is not without drawbacks, however, since high values will result in wasted computation and memory. This trade-off is explained in Figure 3. Empirically we find ensuring lower rates of dropped tokens are important for the scaling of sparse expert-models. Throughout our experiments we didn't notice any dependency on the number of experts for the number of tokens dropped (typically < 1%). Using the auxiliary load balancing loss (next section) with a high enough coefficient ensured good load balancing. We study the impact that these design decisions have on model quality and speed in Table 1. A Differentiable Load Balancing Loss. To encourage a balanced load across experts we add an auxiliary loss (Shazeer et al., 2017, 2018; Lepikhin et al., 2020). As in Shazeer et al. (2018); Lepikhin et al. (2020), Switch Transformers simplifies the original design in Shazeer et al. (2017) which had separate load-balancing and importance-weighting losses. For each Switch layer, this auxiliary loss is added to the total model loss during training. Given N experts indexed by i = 1 to N and a batch B with T tokens, the auxiliary loss is computed as the scaled dot-product between vectors f and P,
Modified on	28/02/2023 14:40

Name	Switch Transformers~ Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
Number of Coding References	11
Number of Codes Coding	1
Coverage	9.37%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	In deep learning, models typically reuse the same parameters for all inputs. Mixture of Experts (MoE) models defy this and instead select different parameters for each incoming example. The result is a sparsely-activated model—with an outrageous number of parameters—but a constant computational cost. However, despite several notable successes ofMoE, widespread adoption has been hindered by complexity, communication costs, and training instability. We address these with the introduction of the Switch Transformer. We simplify the MoE routing algorithm and design intuitive improved models with reduced communication and computational costs. Our proposed training techniques mitigate the instabilities, and we show large sparse models may be trained, for the first time, with lower precision (bfloat16) formats. We design models based off T5-Base and T5-Large (Raffel et al., 2019) to obtain up to 7x increases in pre-training speed with the same computational resources. These improvements extend into multilingual settings where we measure gains over the mT5-Base version across all 101 languages. Finally, we advance the current scale of language models by pre-training up to trillion parameter models on the “Colossal Clean Crawled Corpus”, and achieve a 4x speedup over the T5-XXL model.12
Modified on	28/02/2023 14:37

Name	Switch Transformers~ Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
Number of Coding References	11
Number of Codes Coding	1
Coverage	9.37%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Model and Data Parallelism It is common to mix both model and data parallelism for large scale models, which was done in the largest T5 models (Raffel et al., 2019; Xue et al., 2020) and in GPT-3 (Brown et al., 2020). With a total of N = n ×m cores, now each core will be responsible for B/n tokens and dff/m of both the weights and intermediate activation. In the forward and backward pass each core communicates a tensor of size [B/n, dmodel] in an all-reduce operation. 21 Fedus, Zoph and Shazeer 5.4 Expert and Data Parallelism Next we describe the partitioning strategy for expert and data parallelism. Switch Transformers will allocate all of their cores to the data partitioning dimension n, which will also correspond to the number of experts in the model. For each token per core a router locally computes assignments to the experts. The output is a binary matrix of size [n, B/n, E, C] which is partitioned across the first dimension and determines expert assignment. This binary matrix is then used to do a gather via matrix multiplication with the input tensor of [n, B/n, dmodel]. einsum([n,B/n, dmodel], [n,B/n,E, C], dimension = [B/n]) (7) resulting in the final tensor of shape [n, E, C, dmodel], which is sharded across the first dimension. Because each core has its own expert, we do an all-to-all communication of size [E, C, dmodel] to now shard the E dimension instead of the n-dimension. There are additional communication costs of bfloat16 tensors of size E×C×dmodel in the forward pass to analogusly receive the tokens from each expert located on different cores. See Appendix F for a detailed analysis of the expert partitioning code. 5.5 Expert, Model and Data Parallelism In the design of our best model, we seek to balance the FLOPS per token and the parameter count. When we scale the number of experts, we increase the number of parameters, but do not change the FLOPs per token. In order to increase FLOPs, we must also increase the dff dimension (which also increases parameters, but at a slower rate). This presents a trade-off: as we increase dff we will run out of memory per core, which then necessitates increasing m. But since we have a fixed number of cores N, and N = n × m, we must decrease n, which forces use of a smaller batch-size (in order to hold tokens per core constant). When combining both model and expert-parallelism, we will have all-to-all communica- tion costs from routing the tokens to the correct experts along with the internal all-reduce communications from the model parallelism. Balancing the FLOPS, communication costs and memory per core becomes quite complex when combining all three methods where the best mapping is empirically determined. See our further analysis in section 5.6 for how the
Modified on	28/02/2023 14:42

Name	Switch Transformers~ Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
Number of Coding References	11
Number of Codes Coding	1
Coverage	9.37%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Scaling Properties We present a study of the scaling properties of the Switch Transformer architecture during pre-training. Per Kaplan et al. (2020), we consider a regime where the model is not bottlenecked by either the computational budget or amount of data. To avoid the data bottleneck, we use the large C4 corpus with over 180B target tokens (Raffel et al., 2019) and we train until diminishing returns are observed. The number of experts is the most efficient dimension for scaling our model. Increasing the experts keeps the computational cost approximately fixed since the model only selects one expert per token, regardless of the number of experts to choose from. The router must compute a probability distribution over more experts, however, this is a lightweight computation of cost O(dmodel × num experts) where dmodel is the embedding dimension of
Modified on	28/02/2023 14:41
Name	Switch Transformers~ Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
Number of Coding References	11
Number of Codes Coding	1
Coverage	9.37%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Scaling Results on a Time-Basis Figure 4 demonstrates that on a step basis, as we increase the number of experts, the performance consistently improves. While our models have roughly the same amount of FLOPS per token as the baseline, our Switch Transformers incurs additional communication costs across devices as well as the extra computation of the routing mechanism. Therefore, the increased sample efficiency observed on a step-basis doesn't necessarily translate to a better model quality as measured by wall-clock. This raises the question: For a fixed training duration and computational budget, should one train a dense or a sparse model?
Modified on	28/02/2023 14:41

Name	Switch Transformers~ Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
Number of Coding References	11
Number of Codes Coding	1
Coverage	9.37%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Scaling Versus a Larger Dense Model The above analysis shows that a computationally-matched dense model is outpaced by its Switch counterpart. Figure 6 considers a different scenario: what if we instead had allocated our resources to a larger dense model? We do so now, measuring Switch-Base against the next strong baseline, T5-Large. But despite T5-Large applying 3.5x more FLOPs per token,
Modified on	28/02/2023 14:41

Name	Switch Transformers~ Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
Number of Coding References	11
Number of Codes Coding	1
Coverage	9.37%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Simplifying Sparse Routing Mixture of Expert Routing. Shazeer et al. (2017) proposed a natural language Mixture- of-Experts (MoE) layer which takes as an input a token representation x and then routes this to the best determined top-k experts, selected from a set {Ei(x)}N The router variable Wr produces logits h(x) = Wr · x which are normalized via a softmax distribution over the available N experts at that layer. The gate-value for expert i is given by, pi(x) = N eh(x)i j eh(x)j . (1) The top-k gate values are selected for routing the token x. If T is the set of selected top-k indices then the output computation of the layer is the linearly weighted combination of each expert's computation on the token by the gate value, y = i∈T pi(x)Ei(x). (2) Switch Routing: Rethinking Mixture-of-Experts. Shazeer et al. (2017) conjec- tured that routing to k > 1 experts was necessary in order to have non-trivial gradients to the routing functions. The authors intuited that learning to route would not work without the ability to compare at least two experts. Ramachandran and Le (2018) went further to 5 i=1 of N experts.
Modified on	28/02/2023 14:40

Name	Switch Transformers~ Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
Number of Coding References	11
Number of Codes Coding	1
Coverage	9.37%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	study the top-k decision and found that higher k-values in lower layers in the model were important for models with many routing layers. Contrary to these ideas, we instead use a simplified strategy where we route to only a single expert. We show this simplification preserves model quality, reduces routing computation and performs better. This k = 1 routing strategy is later referred to as a Switch layer. Note that for both MoE and Switch Routing, the gate value pi(x) in Equation 2 permits differentiability of the router. The benefits for the Switch layer are three-fold: (1) The router computation is reduced as we are only routing a token to a single expert. (2) The batch size (expert capacity) of each expert can be at least halved since each token is only being routed to a single expert.3 (3) The routing implementation is simplified and communication costs are reduced. Figure 3 shows an example of routing with different expert capacity factors.
Modified on	28/02/2023 14:40

Name	Switch Transformers~ Scaling to Trillion Parameter Models with Simple and Efficient Sparsity Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	TensorFI~ A Configurable Fault Injector for TensorFlow Applications
Number of Coding References	6
Number of Codes Coding	2
Coverage	3.74%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	E. Implementation TensorFI supports the following features: • Launching multiple FI runs with support for comparing each FI result with the golden run • Launching multiple FI runs in parallel (multi-threading) • Support for visualizing the modified TensorFlow graphs • Ability to specify fault type etc. in a configuration file • Automated logging of fault injection runs • Support for statistics collection and analysis
Modified on	03/02/2022 15:07
Name	TensorFI~ A Configurable Fault Injector for TensorFlow Applications
Number of Coding References	6
Number of Codes Coding	2
Coverage	3.74%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	In this paper, we build a fault injector for ML applications written using specialized frameworks. Because TensorFlow is the most widely used, publicly available software framework for writing ML applications today, we only support TensorFlow and we call our injector TensorFI. TensorFI has three main features. First, it does not rely on the internal implementation of TensorFlow, aiding its portability to different platforms and TensorFlow versions. Second, it requires minimal modifications for programmers to make to their applications and is hence easy to use. Third, it allows programmers to configure the injection process through an external interface without modifying the application (flexible).
Modified on	03/02/2022 15:07

Name	TensorFI~ A Configurable Fault Injector for TensorFlow Applications
Number of Coding References	6
Number of Codes Coding	2
Coverage	3.74%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	TensorFlow is a high-level dataflow framework for building ML applications and has become the most popular one in the recent past. ML applications are also being increasingly used in safety-critical systems such as self-driving cars and home robotics. Therefore, there is a compelling need to evaluate the resilience of ML applications built using frameworks such as TensorFlow. In this paper, we build a high-level fault injection framework for TensorFlow called TensorFI for evaluating the resilience of ML applications. TensorFI is flexible, easy to use, and portable. It also allows ML application programmers to explore the effects of different parameters and algorithms on error resilience.
Modified on	03/02/2022 15:05

Name	TensorFI~ A Configurable Fault Injector for TensorFlow Applications
Number of Coding References	6
Number of Codes Coding	2
Coverage	3.74%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	Based on our results, we find that the error resilience of ML applications can be very different under different algorithms and input datasets. Hence, ML applications need to be evaluated on a per application basis before their deployment, in order to benchmark their operational resilience. Further, we find that the error resilience (i.e., accuracy drops) of ML applications depends on the amount of output classes available in the input dataset used. This should be taken into consideration when designing resilient ML applications.
Modified on	03/02/2022 15:08

Name	TensorFI~ A Configurable Fault Injector for TensorFlow Applications
Number of Coding References	6
Number of Codes Coding	2
Coverage	3.74%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	Fault Injection (FI) is a widely used technique to evaluate the resilience of software applications to faults. While FI has been extensively used in general purpose applications, its use in ML applications presents three main challenges. First, because ML applications are often written using specialized infrastructures, it is difficult to inject faults at the level of individual program statements or variables as these are hidden inside the framework. Second, it is difficult to interpret the results of the FI experiments as they are dependent on the application and the inputs as well as the framework being deployed. Finally, performing FI in ML applications requires the programmer to understand where faults are likely to occur in the application and map them to its implementation.
Modified on	03/02/2022 15:06

Name	TensorFI~ A Configurable Fault Injector for TensorFlow Applications
Number of Coding References	6
Number of Codes Coding	2
Coverage	3.74%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	TensorFlow is a high-level dataflow framework for building ML applications and has become the most popular one in the recent past. ML applications are also being increasingly used in safety-critical systems such as self-driving cars and home robotics. Therefore, there is a compelling need to evaluate the resilience of ML applications built using frameworks such as TensorFlow. In this paper, we build a high-level fault injection framework for TensorFlow called TensorFI for evaluating the resilience of ML applications. TensorFI is flexible, easy to use, and portable. It also allows ML application programmers to explore the effects of different parameters and algorithms on error resilience.
Modified on	03/02/2022 15:05

Name	Testing machine learning based systems~ a systematic mapping
Number of Coding References	12
Number of Codes Coding	2
Coverage	7.69%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	08/02/2022 13:55
Coded Text	a relatively wide adoption of the proposed adequacy criteria. Indeed, availability of MLspecific ways to measure the adequacy of the test data is crucial for MLS testing. Only a few papers adopted adequacy criteria for black box testing, e.g., scenario coverage, that is useful when we do not have white box access and we are interested in the behaviour of the whole system
Modified on	28/02/2023 15:22

Name	Testing machine learning based systems~ a systematic mapping
Number of Coding References	12
Number of Codes Coding	2
Coverage	7.69%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Compared to input generation, the oracle problem in MLS testing has received substan- tially less attention, indicating the need for further approaches to produce effective MLS oracles. System level oracles are particularly difficult to define, being extremely domain specific (e.g., in the self-driving car domain, they require the definition of safe driving conditions and thresholds). Moreover, they often take the form of continuous quality functions (e.g., quality of driving metrics) rather than binary ones (e.g., the car crashing or not).
Modified on	28/02/2023 15:22

Name	Testing machine learning based systems~ a systematic mapping
Number of Coding References	12
Number of Codes Coding	2
Coverage	7.69%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	domain-specific deviations from the expected behaviour are adopted, such as collisions with pedestrians or other vehicles, not stopping at the stop sign, or exceeding the speed limit. Differential or cross-referencing oracle is a type of “pseudo-oracle” (Davis and Weyuker 1981) in which multiple implementations of the same algorithm are compared against each other. If the results are not the same, then one or more of the implementations may contain a defect. This type of oracle was used in six analysed papers (12%) (Murphy et al. 2007b; 2007a; Pei et al. 2017; Sekhon and Fleming 2019; Udeshi and Chattopadhyay 2019;Qin et al. 2018). While the work by Qin et al. (2018) proposes a program synthesis approach that constructs twin “oracle-alike mirror programs”, the remaining papers find different implementations for the MLS under test and use those to cross check the results. A drawback of this type of oracle is the attribution of the fault when the considered implementations produce different results. This was the case in the work by Murphy et al. (2007b)and the authors commented that “there was no way to know which output was correct”. On the other hand, three papers from our pool (Pei et al. 2017; Udeshi and Chattopadhyay 2019; Sekhon and Fleming 2019) take advantage of such a situation, as getting different outputs in each of the different implementations makes the inputs rather interesting and worth further investigation by the developers. Pei et al. (2017) and Sekhon and Fleming (2019)use such differential behaviour along with a coverage criterion as part of a joint optimisation problem aimed to generate erroneous corner case scenarios for MLSs. Similarly, Udeshi and Chattopadhyay (2019) propose an approach that, given a pair ofML models and a grammar encoding their inputs, searches the input space for inputs that expose differential behaviours. Another commonly used oracle for classifiers (6 papers out of 50, 12%) is the misclas- sification of manually labeled inputs (Gopinath et al. 2018; Fremont et al. 2019;Maetal. 2018c, 2019; Zhang et al. 2019; Shi et al. 2019). While using human labels as an oracle is a pretty straightforward approach (especially for data-driven systems such as MLS), it may also require substantial effort. Another type of oracle observed during our analysis is mutation killing, which is used in three different papers (6%) that either propose (Ma et al. 2018d;Shenetal. 2018)orevaluate(Chengetal. 2018a) mutation operators for MLS.
Modified on	28/02/2023 15:23

Name	Testing machine learning based systems~ a systematic mapping
Number of Coding References	12
Number of Codes Coding	2
Coverage	7.69%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Figure 11 shows how often each adequacy criterion was used. Overall, the data indicate a relatively wide adoption of the proposed adequacy criteria. Indeed, availability of MLspecific ways to measure the adequacy of the test data is crucial for MLS testing. Only a few papers adopted adequacy criteria for black box testing, e.g., scenario coverage, that is useful when we do not have white box access and we are interested in the behaviour of the whole system.
Modified on	28/02/2023 15:22

Name	Testing machine learning based systems~ a systematic mapping
Number of Coding References	12
Number of Codes Coding	2
Coverage	7.69%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Figure 13 provides an overview of the types of oracles that have been adopted with MLSs. The most popular type of oracle is the metamorphic oracle, used in 22 out of 50 (44%) papers (Aniculaesei et al. 2018;Dingetal. 2017;Duetal. 2019; Dwarakanath et al. 2018; Guo et al. 2018; Murphy et al. 2008, b, 2009; Nakajima and Bui 2016, 2018, 2019;Saha and Kanewala 2019; Tian et al. 2018;Udeshietal. 2018; Xie et al. 2011, 2018, 2019; Zhang et al. 2016, 2018b; Sun et al. 2018a;Tuncalietal. 2018, 2019). A central element of a metamorphic oracle is a set of metamorphic relationships that are derived from the innate characteristics of the system under test. The new test inputs are generated from the existing ones using MRs so that the outputs for these inputs can be predicted. Out of 22 papers adopting a metamorphic oracle, 11 focus on proposing and evaluating novel MRs for different kinds of MLS. However, these papers mostly consider classical supervised learning algorithms, such as k-nearest neighbours, naive Bayes classifier, support vector machine, and ranking algorithms. The work by Xie et al. (2018) proposes MRs for unsupervised machine learning algorithms, such as clustering algorithms. The remaining papers (11 out of 22) use MRs that are already available in the literature or that encode well-known domainspecific properties of the system. In 10 out of 50 (20%) papers, domain-specific failure of the MLS under test was used as an oracle (Abdessalem et al. 2016, 2018a, b; Abeysirigoonawardena et al. 2019;Beglerovic et al. 2017;B¨uhler and Wegener 2004; Odena et al. 2019; Rubaiyat et al. 2018; Uesato et al. 2019;Lietal. 2016). In general, failure is denoted as the weakest form of an oracle. However, only one of the analysed papers (Odena et al. 2019) conforms to such definition, i.e., the crash of the system under test. In all remaining cases, more complicated and
Modified on	28/02/2023 15:23

Name	Testing machine learning based systems~ a systematic mapping
Number of Coding References	12
Number of Codes Coding	2
Coverage	7.69%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	input that cannot occur in practice is not a real fault. Udeshi et al. (Udeshi and Chattopadhyay 2019) propose a test input generation approach that mutates inputs in a way that makes the result conform to a given grammar, which characterises the validity domain. Tian et al. (2018) produce artificial inputs that represent real driving scenes in different conditions. A further challenge is assessing whether the results obtained in a simulated environment would also scale to the real world (de Oliveira Neves et al. 2016; Wolschke et al. 2018;Li et al. 2016). Two works propose to generate realistic test scenarios from in-field data (de Oliveira Neves et al. 2016), or by mining test cases from real-world traffic situations or traffic simulators (Wolschke et al. 2018). Test Adequacy Criteria Twelve papers define metrics to measure how a test suite is adequate for assessing the quality of an MLS. They often exploit them to drive test input generation. Since classical adequacy criteria based on the code's control flow graph are ineffective with NNs, as typically 100% control flow coverage of the code of an NN can be easily reached with few inputs, researchers have defined novel test adequacy criteria specifically targeted to neural networks (Kim et al. 2019;Maetal. 2018b, 2019; Sekhon and Fleming 2019; Sun et al. 2018a, b; Pei et al. 2017;Shenetal. 2018; Guo et al. 2018;Xie et al. 2019). Behavioural Boundaries Identification Similar inputs may unexpectedly trigger different behaviours of an MLS. A major challenge is identifying the boundaries between different behaviours in the input space (Mullins et al. 2018; Tuncali and Fainekos 2019), which is related to boundary-value analysis in software testing (Young and Pezz`e 2005). For instance, Tuncali and Fainekos (2019) investigate similar scenarios that trigger different behaviours of autonomous vehicles in safety critical settings, e.g., nearly avoidable vehicle collisions. Scenario Specification and Design For scenario-based test cases, one fundamental challenge is related to the specification and design of the environment in which the MLS operates. In fact, only a high fidelity simulation of the environment can produce realistic and meaningful synthetic data (Klueck et al. 2018; Fremont et al. 2019; Majumdar et al. 2019). Oracle Overall, we found 13 papers in our pool that tackle the oracle problem for MLSs (Zheng et al. 2019; Xie et al. 2011; Nakajima and Bui 2016, 2018, 2019; Qin et al. 2018; Cheng et al. 2018b; Ding et al. 2017; Gopinath et al. 2018; Murphy et al. 2007a, 2008;Saha and Kanewala 2019; Xie et al. 2018). The challenge is to assess the correctness of MLSs' behaviour, which is possibly stochastic, due to the non-deterministic nature of training (e.g., because of the random initialisation of weights or the use of stochastic optimisers) and which depends on the choice of the training set. The vast majority of the proposed oracles leverages metamorphic relations among input data as a way to decide if the execution with new inputs is a pass or a fail, under the assumption that such new inputs share similarities with inputs having known labels (Xie et al. 2011; Cheng et al. 2018b;Dingetal. 2017;Saha and Kanewala 2019). Faults and Debugging Eight works considered in our mapping are related to faults in MLSs. Six of them address the problems of studying and defining the spectrum of bugs in MLSs, and automating the debugging of MLSs (Cheng et al. 2018a; Zhang et al. 2018a; Ma et al. 2018c; Odena et al. 2019; Dwarakanath et al. 2018;Eniseretal. 2019). Concerning the former, two studies in our pool present an empirical study on the bugs affecting MLSs (Cheng et al. 2018a; Zhang et al. 2018a). Indeed, the very notion of a fault for an 5216 Empirical Software Engineering (2020) 25:5193–5254 MLS is more complex than in traditional software. The code that builds the MLS may be bug-free, but it might still deviate from the expected behaviour due to faults introduced in the training phase, such as the misconfiguration of some learning parameters or the use of an unbalanced/non-representative training set (Humbatova et al. 2020; Islam et al. 2019). Regarding debugging automation, four studies address the problem of debugging an MLS (Ma et al. 2018c; Odena et al. 2019), or localising the faults within an MLS (Dwarakanath et al. 2018;Eniseretal. 2019). The challenge in this case is to unroll the hidden decision-making policy of the ML model, which is driven by the data it is fed with. Other two papers (Li et al. 2018; Rubaiyat et al. 2018) investigate how to inject faults in MLSs in order to obtain faulty versions of the system under test. Regression Testing Five papers deal with the regression testing problem in the context of MLSs (Byun et al. 2019; Shi et al. 2019; Zhang et al. 2019; Groce et al. 2014; Wolschke et al. 2017), i.e., the problem of selecting a small set of test scenarios that ensure the absence of mis-behaviours on inputs that were

	managed correctly by the previous version of the MLS. The works by Byun et al. (2019)and by Shietal. (2019) both propose a test prioritisation technique to reduce the effort of labelling new instances of data. Groce et al. (2014) deal with test selection for MLSs, whereas Wolschke et al. (2017) perform test minimisation by identifying nearly-similar (likely redundant) behavioural scenarios in the training set. Online Monitoring and Validation Eight works address the problem of online monitoring for validating the input at runtime. Since during development/training it is impossible to foresee all potential execution contexts/inputs that the MLSmay be exposed to, it is likewise essential to keep monitoring the effectiveness of the systems after they are deployed “in the field”, possibly preventing mis-behaviours when an anomalous/invalid input is being processed by the MLS. Six of them leverage anomaly detection techniques to identify unexpected execution con- texts during the operation of MLSs (Henriksson et al. 2019; Patel et al. 2018; Aniculaesei et al. 2018;Wangetal. 2019; Bolte et al. 2019; Zhang et al. 2018b), whereas two papers are related to online risk assessment and failure probability estimation for MLSs (Strickland et al. 2018; Uesato et al. 2019). Cost of Testing The cost of performing MLS testing is particularly challenging, especially in resource-constrained settings (e.g., during system or in-field testing) and in the presence of high dimensional data. Eight papers tackle this problem in the automotive domain (Abdessalem et al. 2016, 2018a; Beglerovic et al. 2017; Zhao and Gao 2018;B¨uhler and Wegener 2004; Murphy et al. 2009; Abeysirigoonawardena et al. 2019;Tuncalietal. 2018). In this domain, comprehensive in-field testing is prohibitively expensive in terms of required time and resources. Therefore, simulation platforms are typically used to test MLSs since they allow re-testing new system releases on a large number of conditions, as well as in challenging and dangerous circumstances (e.g., adverse weather, or adversarial pedestrians suddenly crossing the road) (Stocco et al. 2020). Integration of ML Models Two papers in our pool test the interplay of different ML models within the same system (Abdessalem et al. 2018b; Zhang et al. 2016). Abdessalem et al. (2018b) address the functional correctness of multiple ML models interacting within autonomous vehicles. Differently, Zhang et al. (2016) focus on different levels of metamorphic testing applied to two different computer vision components within a pipeline. Empirical Software Engineering (2020) 25:5193–5254 5217 Data Quality Assessment MLSs may exhibit inadequate behaviours due to poor training data, i.e., inputs that are not representative of the entire input space. At the same time, low quality test data may produce misleading information about the quality of the MLS under test. Hence, the key step towards improving the MLS quality is by achieving high training/test data quality (Ma et al. 2018d;Udeshietal. 2018).
Modified on	28/02/2023 15:21
Name	Testing machine learning based systems~ a systematic mapping
Number of Coding References	12
Number of Codes Coding	2
Coverage	7.69%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Realism of Test Input Data Input generation should be targeted towards creating input data that can expose faults in the considered system, yet being representative of real-world scenarios (Udeshi and Chattopadhyay 2019;Tianetal. 2018). Indeed, a fault exposed by a test
Modified on	28/02/2023 15:21

Name	Testing machine learning based systems~ a systematic mapping
Number of Coding References	12
Number of Codes Coding	2
Coverage	7.69%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Testing MLSs vs Programmed Software Systems Testing software systems is a complex task aiming to detect and prevent unintended behavior of the software. Besides the well known challenges of classical (i.e., non-ML based) software systems, testing MLSs poses additional challenges. First, the behaviour of an MLS is largely impacted by factors such as the available training data sets, the choice of hyperparameters, the model architecture, algorithm and optimiser. On the other hand, the source code is usually very succinct and less prone to errors, because it consists of a plain sequence of API function invocations, and the decision-making policy (i.e., the actual algorithm) is inferred from the training data. Additionally, the exhibited behaviour, encoded by the learned weights within the model, is very difficult to interpret and debug for humans. Classical testing techniques are not easily applicable to any of the above mentioned artefacts, except for the source code. In this section, we describe the main differences between testing classical software systems vs MLS. 2.4.1 Fault and Failure When considering unintended behavior in software systems, we distinguish between faults and failures. According to their definitions in the IEEE Standard Glossary (IEEE 1990): Definition 3 (Fault) [...] An incorrect step, process, or data definition in a computer program. A typical example of a fault in a classical software system is a wrong instruction in a line of code. Definition 4 (Failure) The inability of a system or component to perform its required functions within specified performance requirements. Empirical Software Engineering (2020) 25:5193–5254 5199 To prevent failures, testing of classical software systems attempts to identify and eliminate faults. Since uncertainty is unavoidable when using ML techniques and mis-predictions are hopefully rare, but definitely possible, a robust MLS should capture such uncertainty and take suitable countermeasures to prevent failures even when an internal ML model fails to make a correct prediction. The inability of an MLS to do so can be considered as a data sensitive fault, according to the IEEE Standard Glossary definition (IEEE 1990): Definition 5 (Data Sensitive Fault) A fault that causes a failure in response to some particular pattern of data. Exposing all data sensitive faults is a difficult task in ML applications, given their intangibly big input space. It is the main goal of MLS testing to find ML mis-predictions that give raise to MLS failures. 2.4.2 Test Input Generation The large input space is the main root cause for faults in MLS. Thus, generating test data that represent the input space properly is an important, yet difficult task. The generated data has to fulfil various criteria, such as diversity and realism, and must be equipped with a suitable oracle. The generation should also scale well, allowing for the creation of sufficient testing data to reliably analyse the robustness of the MLS. Depending on the domain, in practice, this may be a major challenge (Campos et al. 2016) (consider, e.g., crashing a self-driving car for testing purposes Cerf 2018). Therefore, tests are conducted within simulated environments, which enable the detection of failures in a scalable and reproducible manner (Stocco et al. 2020;Cerf 2018). The generation of input data for MLS is particularly challenging since the validity domain (i.e., the subset of the input space that is considered a valid input) is often ambiguous and has blurred borders. 2.4.3 Adequacy Criteria Test Adequacy Criteria aim to evaluate whether the suite of implemented tests is comprehensive enough to test the software thoroughly. Classical metrics such as code coverage are severely limited with respect to ML components, primarily as they saturate with any test suite, since the ML code is usually just a sequence of library function invocations that is fully covered when at least one test input is processed. Mutation Adequacy offers an interesting alternative and is more adequate to MLSs. It is measured as the proportion of mutations—artificial faults injected into the code—that the test suite detects (i.e., kills), provided ML-specific mutation operators are defined and implemented. Hence, for MLS new ML-specific adequacy criteria must be introduced, which

typically either analyse the used test data or the internal states of the ML component.
2.4.4 Oracle
The effectiveness of a set of tests depends strongly on their oracle, i.e., on a way to determine whether the tested behavior is correct or a problem is observed. In the absence of oracles, tests can only detect crashes (smoke testing). While conceptually oracles are the same for both classical software systems as well as MLSs (i.e., they encode the intended system behaviour), for complex domains the correct
5200 Empirical Software Engineering (2020) 25:5193–5254
behaviour of an MLS can be challenging to define even for humans. Moreover, the nondeterministic behaviour of MLSs (i.e., repeated training on the same data may result in different behaviours) require novel notions of statistical correctness. In the context ofMLS, Metamorphic Oracles have emerged as a viable approach to infer oracle information from data. Metamorphic oracles take advantage of metamorphic relations between input values: if a metamorphic relation holds between the inputs, the corresponding MLS outputs must necessarily satisfy a known relation (usually equality or equivalence). For instance, an image representing a handwritten digit classified as a “5” should remain in such class upon minimal addition of noise to a small number of pixels.
2.5 MLS Testing Levels
Test levels define the granularity of the tested entity, e.g., whether the tested entity is a small unit or the system as a whole. Typical test levels in classical software engineering include unit, integration,and system level testing. All such testing levels can also be used for MLS, but they may require further specialisation. In this paper, we distinguish between input testing, model testing, integration testing and
system testing, as defined below. Please note that while the definition of integration testing and system testing are taken from the IEEE Standard Glossary (IEEE 1990), input testing and model testing are new test levels we introduce to better capture the specific properties of MLS testing.
Definition 6 (Input Testing) Input level tests analyse the training data used to train the ML components as well as the input data used at prediction time.
Input level tests do not attempt to find faults in an MLS directly, but rather identify
potential reasons for unsuccessful training in the used data, thus minimising the risk of faults due to prediction uncertainty. Referring to our example of self-driving MLS in Fig. 2, input level testing could either
be performed offline or online. In the former case, it allows e.g. the detection of unbalanced training data, a common issue detrimental to the ML training process. In the latter case, online input validation can be used to identify out-of-distribution inputs, i.e., input images of driving conditions that are underrepresented in the initial training set (e.g., being taken in extreme weather conditions). The latter is a form of input validation.
Definition 7 (Model Testing) Model level tests consider an ML model in isolation, e.g., without taking any of the other components of the MLS into account.
Model level tests aim at finding faults due to, e.g., suboptimal model architectures,
training process or model hyper-parameters. Typical metrics used when performing model level testing include the classical measures
of accuracy (for classifiers) or mean squared error (for regressors) given a labelled test dataset. Model level testing can be considered as the equivalent of unit testing for units that rely on training (i.e., models). In our self-driving car example, model testing can be used to identify inputs for which the lane keeping assistant model produces wrong steering angle predictions, i.e., predictions that deviate substantially from the given ground truth.
Empirical Software Engineering (2020) 25:5193–5254 5201
Definition 8 (Integration Testing) Testing in which software components (including ML models), hardware components, or both are combined and tested to evaluate the interaction between them.
Integration testing focuses on issues that emerge when multiple models and components
are integrated, although each of them behaves correctly in isolation (i.e., for which input or model level tests pass). For example, in our self-driving car, a critical integration scenario would be the case in which the decision making component must decide between possibly hitting a pedestrian suddenly crossing the road (resulting in a failure of the pedestrian protection component), or swerving and possibly crashing the car (i.e., a failure of the lane keeping assistant component).
Definition 9 (System Testing) Testing conducted on a complete, integrated ML based system to evaluate the system’s compliance with its specified requirements.
System level testing aims to test the MLS as a whole in the environment in which it is
supposed to operate. For our self-driving car example, system testing would involve running the car in realistic conditions, for instance by finding tracks and environmental conditions that are critical to handle, or by checking the behaviour in relation to other vehicles and obstacles. System testing can be either simulated, in which synthetic input data and MLS outputs are processed in a simulation environment, or performed in-field, where the system is tested in the same runtime conditions experienced by the end-users.
2.5.1 Access to the Tested Component

	Tests that rely only on the system's inputs and outputs are called black-box tests. They usually generalise well when the software architecture changes from classical software to MLS, but are limited as they cannot rely on coverage of the internal system states. Techniques that instead directly observe the program execution and the execution states are called whitebox tests. Traditional white-box techniques relying on code coverage are quite ineffective on MLS, as coverage of the code is often trivially achieved by any test suite, regardless of its quality, because the behaviour is not encoded in the code's conditional logic (usually, the code for an ML component is just a plain sequence of instructions). For the aim of this systematic mapping, we found it useful to introduce a new class of tests called data-box tests, which can be regarded as intermediate between black-box and white-box tests. The relation between black-box, data-box, and white-box testing levels is illustrated in Fig. 3. More precisely, we adopt the following definitions of testing accesses: Definition 10 (Black-Box Testing) A black-box test has only access to the MLS inputs and outputs. This definition is conceptually equivalent to black-box testing in traditional software testing, as black-box tests do not rely on the internal architecture or state of the system under test. Of course, in their execution the tested systems may use an ML model to make predictions, which are then reflected in the tested output. Inputs for black-box tests can be selected e.g. by equivalence class partitioning, boundary value analysis, or diversity. In MLS testing, the data used to train the model (training set) and to assess its performance after training (test set) determine the behaviour exhibited by the MLS at runtime. 5202 Empirical Software Engineering (2020) 25:5193–5254 White-box Model's Architecture & Weights Data-box Hyper parameters Neuron Activations Training set Test set Black-box Trained Model Fig. 3 Access Levels during MLS Testing Correspondingly, several ML testing techniques are based on the training/test data, rather than the code implementing the ML model. This introduces a new access level to the system under test that fits neither the definitions of classical black-box nor white-box testing. Therefore, in this paper, we introduce a novel testing access for MLS called data-box testing. Definition 11 (Data-Box Testing) A data-box test has access to everything a black-box test has access to. In addition, a data-box test makes use of the training/test data that was originally used to train/assess the ML component. Data-box testing is more permissive than black-box testing, even though they both do not rely on the internal architecture or state of the system under test. Typical use-cases of data-box tests include the modification of the training set followed by a re-training of the model, in order to observe its change in prediction capability. Training sets may also be considered in isolation, to check the heterogeneity of the data and its representativeness of the complete input space. The test set can be also checked for representativeness and completeness. Referring to Fig. 2, data-box testing could for instance identify underrepresented weather conditions in the training set in order to exercise more extensively the self-driving car when running in such conditions. Our definition of white-box testing for MLS is consistent with the classical testing literature, where the test method has access to everything within the tested component. Definition 12 (White-Box Testing) Awhite-box test has access to the internals of the tested component. This includes the ML model, its code, its hyper-parameters, its training/test data and its runtime state (e.g., neuron activations). Typical use-cases of ML specific white-box testing include the analysis of the diver- sity of the states of a model under test given some inputs (e.g., the analysis of the neuron activations in an NN when presented with the validation or test set) as well as mutation testing, achieved through ML specific mutation operators, such as changes of the model's architecture, weights, hyper-parameters, or training data. As shown in Fig. 3, black-box testing is subsumed by data-box testing, which in turn is subsumed by white-box testing, because black-box testing has access only to inputs and outputs; data-box testing has access to the training/test data in addition to inputs and outputs; white-box testing has access to the model's architecture, hyper-parameters, runtime state, in addition to the features accessible to data-box testing.
Modified on	28/02/2023 15:20

Name	Testing machine learning based systems~ a systematic mapping
Number of Coding References	12
Number of Codes Coding	2
Coverage	7.69%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	The majority of the works that use adversarial input generation (5 papers out of 52, 10%) employ the existing state-of-the-art attacking methods to generate such inputs (Cheng et al. 2018a; Kim et al. 2019;Wangetal. 2019; Zhang et al. 2019). In contrast, the work by Abeysirigoonawardena et al. (2019) has a more targeted approach which aims to create adversarial self-driving scenarios that expose poorly-engineered or poorly-trained self-driving policies, and therefore increase the risk of collision with simulated pedestrians and vehicles. While adversarial inputs can successfully trigger misbehaviours of the MLS under test, they are often very unlikely or impossible to occur in reality, unless the system is under the attack of a malicious user. However, security verification and validation of MLS is a different research area on its own and the present systematic mapping does not cover it.
Modified on	28/02/2023 15:22

Name	Testing machine learning based systems~ a systematic mapping
Number of Coding References	12
Number of Codes Coding	2
Coverage	7.69%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	The prevalence of autonomous systems and in particular autonomous driving cars indi- cate that safety critical domains are those in highest demand of techniques to ensure the dependability and reliability of such systems, with testing approaches specifically designed for their peculiar features.
Modified on	28/02/2023 15:21

Name	Testing machine learning based systems~ a systematic mapping
Number of Coding References	12
Number of Codes Coding	2
Coverage	7.69%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	The prevalence of NNs matches the growing popularity and success of this approach to machine learning. Since NNs are general function approximators, they can be applied to a wide range of problems. Hence, testing techniques that prove to be effective on NNs will exhibit an incredibly wide range of application scenarios.
Modified on	28/02/2023 15:22

Name	Testing machine learning based systems~ a systematic mapping
Number of Coding References	12
Number of Codes Coding	2
Coverage	7.69%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	This result indicates that ML models are mostly tested “in isolation”, whereas it would be also important to investigate how failures of these components affect the behaviour of the whole MLS (i.e., whether model level faults propagate to the system level)
Modified on	28/02/2023 15:21

Name	Testing machine learning based systems~ a systematic mapping Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Testing machine learning systems in industry~ an empirical study
Number of Coding References	12
Number of Codes Coding	4
Coverage	11.09%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	03/02/2022 10:47
Coded Text	Furthermore, even if the input is generated from scratch, there is no explicit oracle for the input, especially when evaluating the model behaviors. While the behavior of source code in ML systems can be analyzed with traditional methods, the behavior ofML models making predictions or decisions is difficult to predict and analyze. The latter involves much more external knowledge on matrix operations, statistics, and data science, and thus could be challenging in practice even for senior practitioners. Open Challenge: Practitioners highly expect to reduce the cost of labor-intensive test data collection process, but existing research work on automated test input generation has not addressed practitioners' pain points.
Modified on	18/02/2023 16:07

Name	Testing machine learning systems in industry~ an empirical study
Number of Coding References	12
Number of Codes Coding	4
Coverage	11.09%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	04/06/2022 17:30
Coded Text	Data Quality Assurance. 91% of our respondents sug- gested that their teams performed activities to ensure the data quality after collection, but they often struggled in highdimensional, discrete data. For example, it is relatively easy to evaluate the skewness in data with continuous distributions via techniques like kernel density estimation or KL divergence [5, 16], but it is challenging on sequential discrete data such as natural language. One of our respondents complained that "As we work with natural language, it is really hard to quantify howmany samples we need for train and test, and how test data varies with training data". Open Challenge: Data quality assurance is very important in practice, due to the "garbage in garbage out" characteristic of ML. However, it is challenging to test the data quality, especially for high-dimensional, discrete data, e.g., natural language utterances.
Modified on	18/02/2023 16:08

Name	Testing machine learning systems in industry~ an empirical study
Number of Coding References	12
Number of Codes Coding	4
Coverage	11.09%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	Entanglement Problem inTestingWorkflow. All interviewees find that the entanglement among data, source code and ML model makes the testing and debugging highly inefficient and challenging. Specifically, whenever a change is made in data or code, not only does the changed component needs testing, but also the ML model needs re-training and re-testing, due to their entanglement. Bug triage can also be hard since a defect can be caused by insufficient training data, buggy implementations ofML models, improper design ofML models, or just the limitation of the ML algorithm. In addition, some respondents added that the ML models can also be entangled, making the testing and debugging more challenging. Fig. 3 presents three architectures ofML systems. (1) Single-model (39%, Fig. 3a). There is only one ML model in the system, and this model solves the problem in an end-to-end manner. 39% of our respondents suggested that they used the single-model architecture. (2) Pipeline (38%, Fig. 3b). An ML problem is decomposed into several subproblems, each of which is solved by a distinct model. For example, a speech translation problem can be decomposed
Modified on	18/02/2023 16:08

Name	Testing machine learning systems in industry~ an empirical study
Number of Coding References	12
Number of Codes Coding	4
Coverage	11.09%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	Evaluation Metrics Selection forMLmodels. As suggested by 98% ofour respondents, it is a sophisticated task to design and select metrics for evaluating an ML model's properties, such as correctness [41], robustness [40], and fairness [17]. Specifically, there are a lot of metrics for different types ofML (e.g., supervised learning and unsupervised learning), different models, and different model properties. Take one property of ML models, robustness, as an example. Although researchers have proposed various metrics to evaluate the robustness [2, 4, 40], there is currently no consensus on which metric to use. Consequently, all of our interviewees suggested that their teams did not adopt these metrics proposed in the literature because they are not aware of any metrics that are suitable for their models, and the robustness of their ML models are still tested and evaluated in
Modified on	18/02/2023 16:09

Name	Testing machine learning systems in industry~ an empirical study
Number of Coding References	12
Number of Codes Coding	4
Coverage	11.09%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	Given the wealth of metrics proposed in the literature, it may be the time to ask how far are we from effective ML model evaluations. The answer to this question, on the one hand, can help the industry to provide more reliable ML systems. On the other hand, it can make researchers better positioned to design practical metrics. Open Challenge: Selecting evaluation metrics for ML models is a sophisticated task, because there is currently no consensus on which metrics to use, especially for the nonfunctional properties ofML models.
Modified on	18/02/2023 16:09

Name	Testing machine learning systems in industry~ an empirical study
Number of Coding References	12
Number of Codes Coding	4
Coverage	11.09%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	Implications Based on our study, we summarize some inspirations from practice, which can serve as an important complement to the discussion on ML testing in literature [41]. 4.1 Implications for ML Testing Specification of requirements: The specification of re- quirements plays a significant role in the whole lifecycle of ML systems, including testing. In traditional software systems, the requirements are extensively studied and are usually converted to design specifications that can decouple the requirement analysis from the later development and testing process. But in ML systems, things get different. The behavior of an ML system usually can not be completely and formally specified by traditional methods as it constantly updates with the new data. Thus, the specification of requirements is important in the whole process ofML system development, including testing. In our study, we actually find that practitioners use the information from the application context, such as the user scenario described in Section 3.1, to improve their testing practices. However, just as mentioned in [29], how to effectively use the specification of requirements in the ML systems needs more research. Test case collection: It is difficult to formally define the valid input space ofML programs (e.g., how to define a human face image). So it remains an open question how to measure the test coverage in ML systems. The oracle problem is also severe while the ML systems are called non-testable in prior work [41]. It is non-trivial to specify the expected results before running the ML models. In practice, we find that the activity to prepare partial oracles for the ML systems, i.e., data labeling, is often very expensive. Assumptions changed: The incorporation of ML with software comprehensively affects the testing activities on each level while it challenges some assumptions in traditional software systems, such as encapsulation and modular design. In traditional software testing, we often use unit tests to test whether a code module faithfully implements its design specification, because the behavior of the module solely 270 depends on its code. But in an ML system, data, code, and model are entangled together, so unit tests cannot effectively answer the above question. In addition, the testing activities, from unit test to regression test, are now conducted in a statistical manner due to the statistical nature of ML. 4.2 Research Opportunity We have touched on some research opportunities in the preceding sections. Herewe discuss three promising research avenues in ML testing. Test input generation with contextual constraints: As mentioned, the natural inputs for ML systems are difficult to generate. Contextual constraints, such as application scenarios, play a significant role in reducing the search space and generating natural inputs. While current research mainly aims to generate inputs in specific fields, such as computer vision and natural language, more contextual constraints can be taken into account to achieve further efficiency. We believe contextual information could be helpful and deserve more attention in research on test input generation for ML systems. More broadly, the specification of requirements, except for the application context, can possibly help on testing, and there are some preliminary attempts in research [29]. Regression testing for MLmodels: As discussed in Sec- tion 3.2, the regression on model performance is severe in ML systems and can be covered under the overall performance improvement. While the practitioners conduct regression testing in ML systems by retesting all, exposing all regression cases is somehow costly. And the primary concern in ML systems shifts to explain or avoid the regression, which is even more significant than test selection or prioritization. Moreover, the regression in hierarchical ML systems can be more complicated due to the inter-correlation among models. These problem are challenging and remain to be studied. Efficient testing workflow or paradigms: The tradi- tional testing workflow is inefficient in ML systems while some traditional assumptions are challenged. Generally, the statistical and unexplainable behaviors of ML systems make the testing process repetitive and even uncontrollable. What’s worse, the non-determinism ofML causes test flakiness [27]. While the efficient testing workflow of ML systems remains an open challenge, we believe systematic research in this field could be promising and very

	beneficial. First, the traditional testing activities need to be adjusted to adapt to the ML properties. Meanwhile, the new testing activities introduced by data and ML model need to be integrated into the process. Finally, some fundamental problems on testing, such as regression, are also worth more in-depth study to improve the efficiency of the overall ML testing process. During the survey, we also got some encouraging feed- back from our respondents that they learned many testing practices from our questionnaire, and they would like to acquire the advanced testing techniques to examine whether techniques can be applied to their ML systems.
Modified on	18/02/2023 16:10
Name	Testing machine learning systems in industry~ an empirical study
Number of Coding References	12
Number of Codes Coding	4
Coverage	11.09%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	Nevertheless, how to improve the testing efficiency and the debuggability ofML systems remains challenging. There is no good standard or guidance for practitioners. As suggested by one of our respondents, one promising research direction is to provide better test tool support for industry, e.g., automating the testing workflow and easing integration between model production and evaluation. Also, we notice that prior research work for ML testing mainly focuses on single-model ML systems. But given the high proportion of pipeline and hierarchical ML systems in the industry, it is necessary for the research community to put more research efforts on these ML systems. Open Challenge: Dealing with the entanglement among the ML components and the various architecture ofML systems is challenging, which hinders efficient test execution.
Modified on	18/02/2023 16:08
Name	Testing machine learning systems in industry~ an empirical study
Number of Coding References	12
Number of Codes Coding	4
Coverage	11.09%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	Prior research work on regression testing focuses on prioritizing and selecting test cases to improve the test efficiency [18, 26]. Thanks to the advanced parallel computing techniques, the cost of running test on ML models is usually affordable. So the focus may be shifted to regression reduction, termed catastrophic forgetting in neural networks [20], which still remains an open question. Open Challenge: Due to the statistical nature of ML, the regression problem of ML systems becomes more severe and requires new techniques to solve.
Modified on	18/02/2023 16:09

Name	Testing machine learning systems in industry~ an empirical study
Number of Coding References	12
Number of Codes Coding	4
Coverage	11.09%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	Regression Problem on MLmodel. 65% ofour respondents agreed that due to the statistical nature ofML, the regression problem ofML systems becomes more severe than
Modified on	18/02/2023 16:08

Name	Testing machine learning systems in industry~ an empirical study
Number of Coding References	12
Number of Codes Coding	4
Coverage	11.09%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	traditional software systems and requires new techniques to solve. The remainder of respondents did not consider the partial regression ofML model as a significant problem since their team only focused on the overall metrics, such as accuracy and recall, due to their business decisions. Fig. 4 visualizes the regression problem in ML system. Although the new ML model (version 2) passes more test cases than its previous version (version 1), it fails on some test cases passed in its previous version. Practitioners prefer a new ML model that outperforms its previous version and passes all test cases passed by its previous version.
Modified on	18/02/2023 16:08

Name	Testing machine learning systems in industry~ an empirical study
Number of Coding References	12
Number of Codes Coding	4
Coverage	11.09%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	04/02/2022 21:53
Coded Text	All interviewees consider it important to measure the cor- relation between offline metrics and online metrics, i.e., examining whether the improvements in offline metrics translate to online improvements. 65% of respondents in our survey agree with it. The good practice from P2's team is to conduct rounds of small-scale online testing with intentionally degraded or upgraded models to test the relationships between online and offline metrics. Open Challenge: Designing online metrics for ML models is challenging, as it closely relates to how users interact with the system, and it is also important to measure the correlations between different metrics.
Modified on	18/02/2023 16:09

Name	Testing machine learning systems in industry~ an empirical study
Number of Coding References	12
Number of Codes Coding	4
Coverage	11.09%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	04/02/2022 21:53
Coded Text	In our study, an important practice to assess the testing results is data slicing. 69% of respondents noted that their teams slice data along multiple dimensions based on application context information to understand the source of improvement or degradation of ML models. For example, as P6 suggested, his team grouped natural language queries in the test set by the frequency of usage and examine from which group(s) the performance changes originate. Then, they can identify the strength or the weakness of the model and come up with solutions accordingly. Open Challenge: Since it is impossible to fully decouple the components in ML systems, gaining the interpretability of test results is complicated and challenging in practice.
Modified on	18/02/2023 16:10

Name	Testing machine learning systems in industry~ an empirical study Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	TF-DM~ Tool for Studying ML Model Resilience to Data Faults
Number of Coding References	4
Number of Codes Coding	2
Coverage	15.79%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	TensorFlow is one of the most popular open source frame- works for writing and deploying ML applications. So we want TF-DM users to be able to comprehensively characterize any ML model implemented in TensorFlow 2 with respect to different data level faults. In this section, we describe some of the implementation challenges we faced during development. We then discuss the data mutators we evaluate in the next section along with their potential use cases. A. Implementation challenges 1) Handling different datasets: We use the datasets avail- able through the Keras and TensorFlow libraries. The datasets from Keras are smaller and can be loaded on the fly through the load data function. In TF-DM, we use TensorFlow APIs and the NumPy library to mutate the parameters according to configured faults. Once the training or test data is passed to the mutator, it checks for the amount of mutation specified. For example, if it is specified to be 50, then a random 50% of the loaded data is sliced. Then depending on the mutator, it is further operated on by flattening the elements, performing the mutation and reshaping the data back to the original dimensions. The challenge comes with larger datasets such as ImageNet, as they are usually downloaded beforehand and due to the sheer amount of data samples, mutation on them as a whole entity is infeasible because of system memory constraints. These are usually loaded using Keras preprocessing APIs to generate tf.data.Dataset objects from image files from the directory. So we make use of the ImageDataGenerator class available in TensorFlow to operate on smaller batches of the training data at a time and retrain them with the fit generator function exported by the Keras Model APIs. 2) Using TensorFlow 2: In TensorFlow 1, the session objects contained all the information regarding the graph operations and model parameters. In TensorFlow 2, there is no graph duplication as the eager execution model is adopted. This meant that previous fault injectors or data mutators developed for TensorFlow 1 cannot be used for the programs written in the current version, and so we need to develop a framework that addresses this fundamental design change. We do so in TF-DM by taking advantage of eager execution, directly operating on a copy of the mutation artifact rather than the entire graph, and returning it to the ML program to observe the faulty inference runs. This implementation is fast, and is general enough to work with programs that build models using any of the sequential, functional or subclassing model APIs that are available for TensorFlow 2. Finally, since the TensorFlow operations are not modified, this method also guarantees compatibility with future API changes. B. Data mutators Mutating the data samples can help determine the model robustness, how well it can handle adversarial examples and improve resilience with adversarial training. 1) Data removal: The data removal mutators remove an amount of training or test data specified by the user. This helps users to determine the minimal training data required for correct prediction for the chosen model and dataset. 2) Feature noise addition: The feature noise addition mu- tator can add different types of noise into the pixels of images to mimic noisy data. Both DeepMutation and the TensorFlow framework let users add Gaussian noise to layers to mitigate overfitting, color augmentations and image transformations. Although real world noise can widely differ on account of weather conditions or camera capture related noise, artificial noise can nevertheless help simulate the model resiliency to real noise. For this reason, we provide additional noise types such as speckle, salt and pepper, poisson and random noise. Users can configure both the amount of noise in each image and the total amount of noisy images for testing. Varying the noise type and intensity helps users understand the properties and resilience of the features in the training data. 3) Label error: The label error mutator involves modifying the labels of training or test data either randomly or to a particular class of the dataset. By varying the amount of mislabelled data, users can determine how susceptible the model is to (i) targeted or (ii) untargeted misclassifications.
Modified on	28/02/2023 12:06

Name	TF-DM~ Tool for Studying ML Model Resilience to Data Faults
Number of Coding References	4
Number of Codes Coding	2
Coverage	15.79%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	we present our tool TensorFlow Data Mutator (TF-DM), which supports three types of data mutators, so users can perform a holistic assessment of their ML models to the effects of different data faults. TF-DM allows users to (1) remove parts of the training or test data to understand the minimal amount of data that is required for their model, (2) mislabel parts of the data to see the consequences of both targeted and untargeted misclassifications from adversarial attacks, and (3) add different kinds of noise to the data to emulate the effects of noisy inputs. Currently, TF-DM has the capability to mutate different datasets from the Keras and TensorFlow libraries, including support for large scale datasets such as ImageNet. In summary, our main contributions in this paper are: • Present the background of the different faults and attacks at the data level in ML applications, • Discuss the challenges in modeling the various fault models to corresponding data mutators in a framework, • Implement three mutators in an automated tool TF-DM, • Perform detailed evaluation of TF-DM on 6 ML models and 3 datasets commonly used in the literature
Modified on	28/02/2023 12:06

Name	TF-DM~ Tool for Studying ML Model Resilience to Data Faults
Number of Coding References	4
Number of Codes Coding	2
Coverage	15.79%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	It has thus become a necessity to assess the dependabil- ity of the ML models before they are deployed in safetycritical applications. To that end, research applying conventional software techniques have seen various forms of success in evaluating different aspects of the ML model. Mutation testing [8], differential fuzzing [9], metamorphic testing [10]– [12], whitebox [13] and blackbox testing tools [14] for ML have been developed. Fault injection is another well known technique to evaluate the reliability of applications and tools such as Ares [15], TensorFI [16], PyTorchFI [17] have been developed for different ML frameworks with the goal of resiliency analysis. These tools are typically used for studying the impact of hardware faults on ML models such as bit-flips. However, there is no comprehensive tool that produces an endto-end evaluation of an ML application for faults in input data.
Modified on	28/02/2023 12:05

Name	TF-DM~ Tool for Studying ML Model Resilience to Data Faults
Number of Coding References	4
Number of Codes Coding	2
Coverage	15.79%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	Machine learning (ML) is widely deployed in safety- critical systems (e.g. self-driving cars). Failures can have disastrous consequences in these systems, and hence ensuring the reliability of its operations is important. Mutation testing is a popular method for assessing the dependability of applications and tools have recently been developed for ML frameworks. However, the focus has been on improving the quality of test data. We present an open source data mutation tool, TensorFlow Data Mutator (TF-DM), which targets different kinds of data faults for any ML program written in TensorFlow 2. TF-DM supports different types of data mutators so users can study model resilience to data faults. We explain how different fault models are mapped to mutators in TF-DM, and present a detailed evaluation and resiliency analysis of 6 ML models and 3 datasets
Modified on	28/02/2023 12:05

Name	TF-DM~ Tool for Studying ML Model Resilience to Data Faults Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	The Machine Learning Bazaar~ Harnessing the ML Ecosystem for Effective System Development
Number of Coding References	8
Number of Codes Coding	4
Coverage	4.07%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	
Modified on	03/02/2022 14:42

Name	The Machine Learning Bazaar~ Harnessing the ML Ecosystem for Effective System Development
Number of Coding References	8
Number of Codes Coding	4
Coverage	4.07%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	
Modified on	03/02/2022 14:43

Name	The Machine Learning Bazaar~ Harnessing the ML Ecosystem for Effective System Development
Number of Coding References	8
Number of Codes Coding	4
Coverage	4.07%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	To address these problems, we introduce the Machine Learning Bazaar, a new framework for developing machine learning and automated machine learning software systems. First, we introduce ML primitives, a unified API and specification for data processing and ML components from different software libraries. Next, we compose primitives into usable ML pipelines, abstracting away glue code, data flow, and data storage. We further pair these pipelines with a hierarchy ofAutoML strategies — Bayesian optimization and bandit learning. We use these components to create a general-purpose, multi-task, end-to-end AutoML system that provides solutions to a variety ofdata modalities (image, text, graph, tabular, relational, etc.) and problem types (classification, regression, anomaly detection, graph matching, etc.). We demonstrate 5 real-world use cases and 2 case studies of our approach
Modified on	03/02/2022 14:38
Name	The Machine Learning Bazaar~ Harnessing the ML Ecosystem for Effective System Development
Number of Coding References	8
Number of Codes Coding	4
Coverage	4.07%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	07/02/2022 14:10
Coded Text	. To address these problems, we introduce the Machine Learning Bazaar, a new framework for developing machine learning and automated machine learning software systems. First, we introduce ML primitives, a unified API and specification for data processing and ML components from different software libraries. Next, we compose primitives into usable ML pipelines, abstracting away glue code, data flow, and data storage. We further pair these pipelines with a hierarchy ofAutoML strategies — Bayesian optimization and bandit learning. We use these components to create a general-purpose, multi-task, end-to-end AutoML system that provides solutions to a variety ofdata modalities (image, text, graph, tabular, relational, etc.) and problem types (classification, regression, anomaly detection, graph matching, etc.). We demonstrate 5 real-world use cases and 2 case studies of our approach.
Modified on	25/02/2022 10:00

Name	The Machine Learning Bazaar~ Harnessing the ML Ecosystem for Effective System Development
Number of Coding References	8
Number of Codes Coding	4
Coverage	4.07%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	07/02/2022 14:10
Coded Text	AutoML libraries. AutoML research has often been lim- ited to solving sub-problems of an end-to-end ML workflow, such as data cleaning [14], feature engineering [28, 30], hyperparameter tuning [18, 21, 25, 27, 33, 40, 46, 48], or algorithm selection [25, 50]. Thus AutoML solutions are often not widely applicable or deployed in practice without human support.
Modified on	06/06/2022 11:21

Name	The Machine Learning Bazaar~ Harnessing the ML Ecosystem for Effective System Development
Number of Coding References	8
Number of Codes Coding	4
Coverage	4.07%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	07/02/2022 14:10
Coded Text	AutoML systems. These AutoML libraries, if deployed, are typically one component within a larger system that aims to manage several practical aspects such as parallel and distributed training, tuning, and model storage, and even serving, deployment, and graphical interfaces for model building. These include ATM [47], Vizier [20], and Rafiki [52], as well as commercial platforms like Google AutoML, DataRobot, and Azure Machine Learning Studio. While these systems provide many benefits, they have several limitations. First, they each focus on a specific subset ofML use cases, such as computer vision, NLP, forecasting, or hyperparameter tuning. Second, these systems are designed as proprietary applications and do not support community-driven integration of new innovations.
Modified on	06/06/2022 11:21

Name	The Machine Learning Bazaar~ Harnessing the ML Ecosystem for Effective System Development
Number of Coding References	8
Number of Codes Coding	4
Coverage	4.07%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications\Architecture ML system
Created on	03/02/2022 14:39
Coded Text	As machine learning is applied more widely, data scientists often struggle to find or create end-to-end machine learning systems for specific tasks. The proliferation of libraries and frameworks and the complexity of the tasks have led to the emergence of “pipeline jungles” — brittle, ad hoc ML systems.
Modified on	06/06/2022 10:51

Name	The Machine Learning Bazaar~ Harnessing the ML Ecosystem for Effective System Development
Number of Coding References	8
Number of Codes Coding	4
Coverage	4.07%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56
Coded Text	AutoML libraries. AutoML research has often been lim- ited to solving sub-problems of an end-to-end ML workflow, such as data cleaning [14], feature engineering [28, 30], hyperparameter tuning [18, 21, 25, 27, 33, 40, 46, 48], or algorithm selection [25, 50]. Thus AutoML solutions are often not widely applicable or deployed in practice without human support. In contrast, ML Bazaar integrates many of these existing approaches and designs one coherent and configurable structure for joint tuning and selection of end-to-end pipelines. AutoML systems. These AutoML libraries, if deployed, are typically one component within a larger system that aims to manage several practical aspects such as parallel and distributed training, tuning, and model storage, and even serving, deployment, and graphical interfaces for model building. These include ATM [47], Vizier [20], and Rafiki [52], as well as commercial platforms like Google AutoML, DataRobot, and Azure Machine Learning Studio. While these systems provide many benefits, they have several limitations. First, they each focus on a specific subset ofML use cases, such as computer vision, NLP, forecasting, or hyperparameter tuning. Second, these systems are designed as proprietary applications and do not support community-driven integration of new innovations. ML Bazaar provides a new approach to
Modified on	03/02/2022 14:45

Name	The security of machine learning in an adversarial setting~ A survey
Number of Coding References	9
Number of Codes Coding	2
Coverage	13.87%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	06/01/2022 15:37
Coded Text	
Modified on	28/02/2023 12:29

Name	The security of machine learning in an adversarial setting~ A survey
Number of Coding References	9
Number of Codes Coding	2
Coverage	13.87%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	06/01/2022 15:37
Coded Text	• Gradient masking method Gradient masking methods enhance ML robustness by modifying the gradients of the input data, loss/activation function, etc. For example, Lyu et al. proposed to mitigate against L-BFGS and FGSM attacks by penalizing the gradient of the loss function of the networks [57]. Shaham et al. attempted to increase the local stability of neural networks by minimizing the loss function over adversarial examples when the model parameters are updating [84]. Moreover, Nguyen and Sinha developed a gradient masking method to defend against C&W attacks, in which the noise is appended to the network logit layer [62]. In addition, Ross and Doshi-Velez analyzed the gradient regularization of inputs and then proposed a method for improving network robustness [78]. Because small adversarial perturbations cannot drastically vary the outputs of the deployed model, this method trains the model by penalizing the input variation degree. • Defensive distillation method Hinton et al. observed that the knowledge of a large DDN model can be transferred to a small model by the distillation technique [36]. This means that distillation can decrease the dimensionality of the DNN. Motivated by this work, Papernot et al. formulated a distillation variant, defensive distillation, to resist against adversarial perturbations used to attack DNN models [69]. In contrast to the knowledge transferring pattern of the original distillation, the defensive distillation method extracts the knowledge from its own structures to enhance resistance to adversarial attacks. Fig. 9 demonstrates the framework of the defensive distillation method. As depicted in Fig. 9, the DNN model first derives the knowledge based on its outputs, and then, it retrains the model using the obtained knowledge. • DeepCloak method The DeepCloak method enhances the robustness of the DNN model by removing useless features [27]. The flowchart of DeepCloak is illustrated in Fig. 10. First, this method inserts a masking layer before the network decision layer. Then, the added layer is trained using the original and adversarial image pairs. Finally, the feature differences of the decision layer and added layer are encoded. Because the most prominent features have the dominant weights, the prominent features can be removed by masking the dominant weights for the added layer. Compared to the above-mentioned defense algorithms, DeepCloak method is more efficient because it eliminates the unnecessary features without retraining the model.
Modified on	28/02/2023 12:30

Name	The security of machine learning in an adversarial setting~ A survey
Number of Coding References	9
Number of Codes Coding	2
Coverage	13.87%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	06/01/2022 15:37
Coded Text	• Universal perturbation defense method The goal of a universal perturbation defense strategy is to rectify the contaminated inputs by a perturbation rectifying network (PRN) before the input layer. Depending upon the PRN, the rectified input data and the origin input data are characterized by the same distributions [2]. The PRN is a pre-input layer and is trained without having to change the parameters of the targeted network. Fig. 13 demonstrates the framework of the universal perturbation defense method based on the PRN. As depicted in Fig. 13, during the data rectification process, this method first feeds the input images into the PRN, and then, it detects possible perturbations based on the output features of the PRN. In summary, the existing defense measures are reported to be successful to resist the adversarial examples on ML prediction phase to some extend. Some approaches (e.g., adversarial training [30,60,80], data compression [17,21,33,59], defensive distillation [36,69]) validate they can defend against different kinds of adversarial attacks via carrying out extensive experiments. However, these methods are probably to decrease the predictive accuracy rate on the benign examples. Some approaches (e.g., [57,62,78,84]) can provide favorable results on a kind of specific adversarial attacks, whereas these methods are usually vulnerable to the same adversarial examples due to their transferability. Therefore, devising robust and universal defense strategies in the absence of reducing the performance of the origin ML model is a challenging task.
Modified on	28/02/2023 12:30

Name	The security of machine learning in an adversarial setting~ A survey
Number of Coding References	9
Number of Codes Coding	2
Coverage	13.87%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	06/01/2022 15:37
Coded Text	Defense strategy The defense strategy under the adversarial setting includes four main parts: (1) the security evaluation mechanism for the ML model, (2) the defense strategy during the training phase, (3) the defense strategy during the prediction/test phase, and (4) the privacy protection method for the ML system. Moreover, Table 2 summarizes the main attributes of the defense methods presented in this paper. 5.1. Security evaluation mechanism Traditional security evaluation methods primarily focus on the classification accuracy of the ML model, and they fail to consider the security of such model. To address this problem, various studies have proposed a number of methods for evaluating the security and robustness of ML models [7,8]. The motivation of the security evaluation is to simulate various realistic attack scenarios and highlight the most critical vulnerabilities by analyzing the influence of the adversarial attacks on a given ML model. Characterized by the defense pattern, the mechanisms underlying these evaluation methods can be categorized into reactive defense and proactive defense, as shown in Fig. 8. Reactive defense refers to an arms race in which both the ML model designer and the adversary are involved. For each iteration, the adversary first analyzes the ML defenses strategy and develops an attack approach against the model. Then, the designer of the ML model responds to this action based on this proposed attack type. Finally, the designer adds features that can defend from the novel attack to update the ML system (Fig. 8(a)). In contrast to reactive defense, the main agent of this arms race in proactive defense is the designer of the ML model. Before deploying the ML system, the designer develops the defense strategies by analyzing the possible deficiencies and threats to the ML model based on the adversarial capability, attack strategy, etc. (Fig. 8(b)). 5.2. Defense strategy during training phase During the ML training phase, the adversary can deploy a poisoning attack to modify the statistical characteristics of the training data. To resist poisoning attacks, a common method to enhance ML model robustness is to sanitize the training data. The data sanitization technique is an immediate method, therein aiming to filter the adversarial examples applied to the training dataset. Nelson et al. proposed a novel method to reject the negative impact on spam filtering systems via data sanitization [61]. This method first builds a new dataset by appending a candidate sample to the original training dataset and then trains the ML model using this new dataset; finally, the ML model
Modified on	28/02/2023 12:29

Name	The security of machine learning in an adversarial setting~ A survey
Number of Coding References	9
Number of Codes Coding	2
Coverage	13.87%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	06/01/2022 15:37
Coded Text	Machine learning (ML) methods have demonstrated impressive performance in many application fields such as autopilot, facial recognition, and spam detection. Traditionally, ML models are trained and deployed in a benign setting, in which the testing and training data have identical statistical characteristics. However, this assumption usually does not hold in the sense that the ML model is designed in an adversarial setting, where some statistical properties of the data can be tampered with by a capable adversary. Specifically, it has been observed that adversarial examples (also known as adversarial input perambulations) elaborately crafted during training/test phases can seriously undermine the ML performance. The susceptibility of ML models in adversarial settings and the corresponding countermeasures have been studied by many researchers in both academic and industrial communities. In this work, we present a comprehensive overview of the investigation of the security properties of ML algorithms under adversarial settings. First, we analyze the ML security model to develop a blueprint for this interdisciplinary research area. Then, we review adversarial attack methods and discuss the defense strategies against them. Finally, relying upon the reviewed work, we provide prospective relevant future works for designing more secure ML models.
Modified on	28/02/2023 12:27

Name	The security of machine learning in an adversarial setting~ A survey
Number of Coding References	9
Number of Codes Coding	2
Coverage	13.87%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	06/01/2022 15:37
Coded Text	ML methods are widely employed in various fields such as the Internet of Things, medical imaging, computer vision, and social networking [106]. Despite the impressive performance ML methods have achieved, it is observed that ML methods are highly vulnerable to adversarial attacks during both the training and prediction phases. This paper presented a comprehensive survey on the security of ML models under adversarial settings. Future work on ML security may include the following aspects. (1) The defense strategy against adversarial examples for DL methods should include more attention. In recent years, many studies have demonstrated that DNNs are vulnerable to subtle input perturbations that result in substantial changes in outputs. Although many defense strategies have been proposed to resist such attacks, we cannot find a fundamental solution to these problems. Therefore, developing a secure DL model under adversarial settings is a crucial issue. (2) Data privacy is important in ML security. Although the DP and HE techniques provide a powerful guarantee for ML methods, they remain inefficient because of the high computational complexity. Hence, it is worth designing effective and efficient encryption methods to ensure ML model privacy. (3) The security evaluation of ML methods under adversarial settings should be highlighted in research. Because of the continuous emergence of new security threats, ML designers should place greater importance on the security evaluation of ML models. However, most evaluation methods are
Modified on	28/02/2023 12:30

Name	The security of machine learning in an adversarial setting~ A survey
Number of Coding References	9
Number of Codes Coding	2
Coverage	13.87%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	06/01/2022 15:37
Coded Text	tivation functions. These methods change the ML model parameters. (2) For external models, these methods use additional modules to detect or suppress adversarial attacks, thereby keeping the ML model parameters unchanged. 5.3.1. Modifying ML model • Adversarial training method Adversarial training was the first approach against adversarial attacks. Since the discovery of adversarial attacks on ML models, there has been a general consensus that adversarial training can enhance the robustness against adversarial examples. In contrast to the general ML training process, adversarial training incorporates a hybrid dataset including adversarial examples and the original samples. In essence, adversarial training is a robust generalization method for the ML training phase [30,60]. However, it is a non-adaptive strategy in the sense that the ML model is required to be trained on relevant adversarial examples to resist a particular type of attack. Moreover, a new study suggests that the adversarially trained model can be again attacked by effective adversarial examples [80]. Fig. 8. A conceptual representation of reactive defense and proactive defense [13], where (a) is reactive defense and (b) is proactive defense. classification performances trained on the original dataset and on the new dataset are evaluated. If the error rates on the two datasets have considerable differences, the candidate samples are identified as having malicious examples and are required to be removed. The relevant experiments are conducted on a spam filter system, and the corresponding results illustrate that the proposed approach can effectively detect malicious data. One limitation of this work is that this method is not appropriate for large-scale data because of the heavy computational cost. Another defense strategy mitigates the impacts of adversarial examples by improving the generalization capability of the ML model. The representation methods include the Bagging (bootstrap aggregating algorithm) and RSM (random subspace method) methods proposed by Biggio et al. [5,6] and the ANTIDOTE algorithm presented by Rubinstein et al. [79]. These methods employ median absolute deviation and Laplace truncated domains to minimize the influence of malicious training data. The results of the experiments show that these methods are effective at suppressing malicious data. 5.3. Defense strategy on prediction/test phase Defenses against adversarial attacks during the prediction/test phase follow two main directions: (1) Modifying ML model, e.g. via transforming the training data/ process, adjusting the network layers, and changing ac- • Data compression method Ghahramani et al. observed that most widely used ML model datasets are comprised as JPG format images. They performed research with regard to the impact of JPG compression [21]. According to their studies, JPG compression can suppress the decrease in performance to a large extent in terms of classification accuracy under FGSM perturbations. Moreover, Guo et al. and Das et al. studied the image JPEG compression effect to mitigate the impact of adversarial images [17,33]. These methods employ an ensemblebased technique to counter FGSM attacks by removing the high-frequency components from the images. In addition, Moosavi et al. proposed to use Discrete Cosine Transform (DCT) compression to defend against universal perturbations [59]. The main limitation of the data compression strategy is that a higher compression rate leads to a loss of ML classification accuracy, whereas a low compression rate cannot effectively counter adversarial attacks. • Foveation-based method Boix et al. demonstrated that the influence of adversarial attacks can be mitigated by a foveation mechanism in different image regions [56]. They observed that the DNN model is robust to scale and transformation changes over the original images enforced by the foveation approach, whereas this property does not generalize to adversarial patterns. As a result, the foveation can be used as an alternative solution for alleviating the impact of adversarial attacks. However, the effectiveness of the foveation against more powerful attacks has not been validated.
Modified on	28/02/2023 12:29

Name	The security of machine learning in an adversarial setting~ A survey
Number of Coding References	9
Number of Codes Coding	2
Coverage	13.87%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	06/01/2022 15:37
Coded Text	To construct an effective ML model, a large amount of training data is required. Generally, most such training data are collected using crowd sourcing techniques. Because these training data usually contain a large amount of private and sensitive information with regard to the users, ensuring training data privacy has become a crucial issue for ensuring the security of ML technologies. In addition, some investigations have demonstrated that the important parameters in the ML model may also be hidden. For instance, the adversary can perform an inversion attack against the ML system to obtain private data inside the ML model [97]. To ensure the data privacy of the ML model, two popular defense strategies are usually employed, i.e., deferential privacy (DP)-based methods and homomorphic encryption (HE)-based methods. 5.4.1. Deferential-privacy-based methods Dwork et al. proposed the Deferential privacy (DP) technique in 2006 and then proved its robustness and security in theory [20]. The DP technique is widely accepted as a means of ensuring model privacy, therein aiming at obscuring the inputs by adding noise to the original data model. Based on DP, Erlingsson et al. proposed a randomized aggregative privacy-preserving ordinal response (RAPPOR) method to achieve strong security in a crowd sourcing process from a user terminal. This method combines a randomized response mechanic and DP technique to obtain a high performance. Recently, Papernot et al. proposed a promising approach, PATE (Private Aggregation of Teacher Ensembles) [70], to ensure the privacy of the ML model. This method constructs a teacher model by training some disjoint subsets of the data and then formulating a student model based on the outputs of the teacher system. Note that the training process of the student model depends on the prediction results of the teacher model, instead of the intrinsic parameters. Fig. 14 presents an overview of PATE. 5.4.2. Homomorphic-encryption-based methods Another important technique for ML data privacy protection is Homomorphic encryption (HE) based methods. HE is a special cryptography technology that allows certain algebraic operations over ciphertexts [76]. Specifically, the encryption value of the algebraic operation results over plaintexts is consistent with the algebraic operation results over the corresponding ciphertexts.
Modified on	28/02/2023 12:30

Name	The security of machine learning in an adversarial setting~ A survey
Number of Coding References	9
Number of Codes Coding	2
Coverage	13.87%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	06/01/2022 15:36
Coded Text	In summary, the existing attack methods prohibit impres- sive results in fooling the DNN models under various datasets, e.g., MNIST [43], ImageNet [18] and CIFAR-10 [47]. It demonstrates that the adversarial attack is a general phenomenon in the field of ML. Many literature ascertains that DL methods are also vulnerable to adversarial attacks in the real physical world. Moreover, the experimental results demonstrate that one common property of adversarial examples is that they transfer well between different neural networks of relatively similar architecture. However, although many hypotheses have been developed to explain the existence of adversarial examples [65,82,90], the underlying reason for this fact is still an open question.
Modified on	28/02/2023 12:28

Name	The security of machine learning in an adversarial setting~ A survey Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	The State of the ML-universe~ 10 Years of Artificial Intelligence & Machine Learning Software Development on GitHub
Number of Coding References	2
Number of Codes Coding	2
Coverage	0.81%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	31/01/2022 14:42
Coded Text	In this paper, we contribute additional insights into AI & ML development and triangulate results from existing studies. We characterize the landscape of AI & ML repositories on GitHub in order to understand the AI & ML boom in recent years and the differences between AI & ML and traditional software development. Specifically, we conduct a large-scale empirical study of GitHub to characterize and compare software development across three types of repositories (Section 2): (1) AI & ML Tools: 700 AI & ML frameworks & libraries (2) Applied AI & ML: 4,524 repositories using AI & ML (3) Comparison: 4,101 repositories unrelated to AI & ML
Modified on	03/02/2022 14:06

Name	The State of the ML-universe~ 10 Years of Artificial Intelligence & Machine Learning Software Development on GitHub
Number of Coding References	2
Number of Codes Coding	2
Coverage	0.81%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	31/01/2022 14:42
Coded Text	As a result, applying AI & ML to solve existing and emergent problems is an increasingly popular practice. However, little is known about this domain from the software engineering perspective. Many AI & ML tools and applications are open source, hosted on platforms such as GitHub that provide rich tools for large-scale distributed software development. Despite widespread use and popularity, these repositories have never been examined as a community to identify unique properties, development patterns, and trends.
Modified on	03/02/2022 14:03

Name	ThunderML~ A Toolkit for Enabling AI~ML Models on Cloud for Industry 4.0
Number of Coding References	10
Number of Codes Coding	3
Coverage	9.73%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 11:15
Coded Text	
Modified on	03/02/2022 11:16

Name	ThunderML~ A Toolkit for Enabling AI~ML Models on Cloud for Industry 4.0
Number of Coding References	10
Number of Codes Coding	3
Coverage	9.73%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 11:15
Coded Text	
Modified on	03/02/2022 11:16

Name	ThunderML~ A Toolkit for Enabling AI~ML Models on Cloud for Industry 4.0
Number of Coding References	10
Number of Codes Coding	3
Coverage	9.73%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 11:15
Coded Text	
Modified on	03/02/2022 11:17

Name	ThunderML~ A Toolkit for Enabling AI~ML Models on Cloud for Industry 4.0
Number of Coding References	10
Number of Codes Coding	3
Coverage	9.73%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 11:15
Coded Text	
Modified on	03/02/2022 11:17

Name	ThunderML~ A Toolkit for Enabling AI~ML Models on Cloud for Industry 4.0
Number of Coding References	10
Number of Codes Coding	3
Coverage	9.73%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 11:15
Coded Text	
Modified on	03/02/2022 11:17

Name	ThunderML~ A Toolkit for Enabling AI~ML Models on Cloud for Industry 4.0
Number of Coding References	10
Number of Codes Coding	3
Coverage	9.73%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 11:15
Coded Text	
Modified on	03/02/2022 11:18

Name	ThunderML~ A Toolkit for Enabling AI~ML Models on Cloud for Industry 4.0
Number of Coding References	10
Number of Codes Coding	3
Coverage	9.73%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 11:15
Coded Text	In order to address these issues, we have developed ThunderML, a Python- based toolkit that makes the creation and deployment of purpose built AI models for industrial applications easier. ThunderML leverages many open source frameworks such as scikit-learn, Tensorflow, and Keras. The extension points are predominantly in terms of how we have built out a series of useful modeling functions and industrial solution templates to expedite the task of building and deploying AI for industrial applications. ThunderML is flexible enough to run on local hardware as well as providing an easier path to using common cloud service provider platforms for enhanced scalability in training and convenient model deployment services. Before we proceed, it's worth briefly giving a few examples of purpose built industrial solution templates available in ThunderML: – Time Series Prediction (TSPred): Flexible solution for forecasting time series from historical data in industries. – Failure Pattern Analysis (FPA): Predicting imminent failures for assets using IoT sensor data and past failure history data; – Root Cause Analysis (RCA): Building interpretable models to assist plant operators track down the root causes for product quality deviances on batch or continuous process lines; – Anomaly Analysis: Building unsupervised/semi-supervised models to identify anomalous behaviors of manufacturing assets; – Cognitive Plant Advisor (CPA): Combines advanced AI to build a predictive model of one or more key process outputs such as throughput and yield and uses these models within a business objective optimization problem to suggest optimal process settings to plant operators. In summary, ThunderML can also help alleviate the skills gap issue that has hampered AI adoption in many industries. In our experience, technically adept (but not necessarily experts in AI personnel) can use ThunderML's industry templates and programming interface to enable industry AI models.
Modified on	03/02/2022 11:15

Name	ThunderML~ A Toolkit for Enabling AI~ML Models on Cloud for Industry 4.0
Number of Coding References	10
Number of Codes Coding	3
Coverage	9.73%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 11:15
Coded Text	Our contribution in this paper is the design and implementation of ThunderML. We elaborately discuss how ThunderML expedites the AI modeling workflow by giving practitioners an easier path for doing advanced modeling work leveraging cloud-based platforms for training and deployment. We then provide a use case to demonstrate the benefits of ThunderML in practice for a very general and widely applicable problem.
Modified on	03/02/2022 11:16

Name	ThunderML~ A Toolkit for Enabling AI~ML Models on Cloud for Industry 4.0
Number of Coding References	10
Number of Codes Coding	3
Coverage	9.73%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications\Architecture ML system
Created on	03/02/2022 11:15
Coded Text	Challenges of Using Existing Cloud-Based AI Platforms While cloud-based AI platforms have done much to facilitate adoption of AI by alleviating many of the infrastructure provisioning and maintenance challenges associated with on-premises enterprise AI initiatives, they have not done enough to abstract away some of the complexity of running AI workflows in vendor agnostic ways. Current platforms expect practitioners to know a given vendor's means and methods of interacting with the computing resources without consideration given to providing a common programming model that makes the job of an AI practitioner easier. Cloud-based AI environments, by their very nature, push users towards batch training modes to facilitate data center resource management via a queued execution model. Such batch training modes are problematic for many data scientists who wish to see errors or results in real or near real time in order to make their modeling workflow more efficient.1 Another issue is that cloud-offerings typically approach AI from either a black-box perspective which offers users simplicity at the cost of flexibility or through a more complex runtime environment that requires users maintain code artifacts that often have nothing to do with the actual AI tasks at hand2.Even with a diverse set of offerings in the market, we feel a gap remains for the AI practitioner community. Cloud AI offerings should be easy to learn and use and provide the right level of complexity and flexibility AI practitioners need.
Modified on	03/02/2022 11:15

Name	ThunderML~ A Toolkit for Enabling AI~ML Models on Cloud for Industry 4.0
Number of Coding References	10
Number of Codes Coding	3
Coverage	9.73%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Research gap in Maintaining ML systems
Created on	03/02/2022 11:14
Coded Text	There are three fundamentals gaps. The first is the lack of purpose built solution pipelines designed for common industrial problem types, the second is the lack of tools for automating the learning from noisy sensor data and the third is the lack of platforms which help practitioners leverage cloud-based environment for building and deploying custom modeling pipelines.
Modified on	03/02/2022 11:14

Name	To Seed or Not to Seed~ An Empirical Analysis of Usage of Seeds for Testing in Machine Learning Projects
Number of Coding References	5
Number of Codes Coding	2
Coverage	6.08%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	in two modes: with seeds and without seeds, and generates a report summarizing the failed tests, the different types of failures, and the failure rates. Overall, we find 461 unique tests across 32 projects that fail when seeds are removed but always pass when seeds are present. Although developers commonly try to mitigate the effects of randomness by setting seeds [25], this is often a “workaround” rather than an actual solution. Instead, it may be possible to limit randomness (and test flakiness) and alleviate the negative effects of setting seeds using statistical techniques. To evaluate this intuition, we select a subset of tests that only fail when seeds are removed and attempt to fix them, i.e., minimize their flakiness, using alternative techniques such as tuning the hyperparameters used for the algorithm under test [24], adjusting the bounds of the assertions in the test [26], or refactoring the test/assertion in other ways. We sent 16 Pull Requests and 7 Issues (for cases where we were unable to fix the test) to the developers. These covered a total of 42 tests. At the time of writing this paper, developers have accepted our changes or fixed 26 tests, while the rest are pending resolution. We further analyze a subset of 56 tests, discuss and categorize various characteristics such as nature of test oracles, source(s) of randomness, evolution of seeds relative to the tests, and how the seeds are set. We also provide a general set of recommendations based on our experience for using (or not using) seeds (more details in Section VI): • Use fixed seeds only for tests checking for exact reproducibility of some functionality in their code. • Randomize and log the seeds in other tests for nondeterministic algorithms to allow both reproducibility of failures and diverse executions. • Use Test Re-runs on failure instead of setting seeds to mitigate random failures. • Determine optimal test settings such as hyper-parameters for the algorithm(s) under test and/or assertion bounds to minimize flakiness. Finally, we discuss the impact of setting (or not setting) seeds and alternative fixes on the fault-detecting ability of tests on several examples (Sections VII-B,VII-C). Contributions. We make the following contributions: • We conduct the first large-scale empirical study of the usage of seeds in tests in 114 Machine Learning projects. • We analyze the tests that fail without seeds and study important aspects related to the nature of such tests and the root causes for flakiness. • We apply various alternative strategies (instead of setting seeds) to fix the root causes for 42 tests and mitigate flakiness. • We provide several insights and implications related to usage of seeds and a general set of recommendations for both developers and researchers.
Modified on	16/02/2023 11:50

Name	To Seed or Not to Seed~ An Empirical Analysis of Usage of Seeds for Testing in Machine Learning Projects
Number of Coding References	5
Number of Codes Coding	2
Coverage	6.08%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Our Work. In this work, we conduct the first large-scale and systematic study of the usage of seeds (for random number generators) and its implications for testing in Machine Learning projects. We study several research questions and provide insights that can be useful for both developers and researchers: 1) How prevalent are tests that fail non-deterministically without seeds and how often they fail? (Section IV), 2) Can we use alternative strategies to mitigate flakiness instead of setting seeds for such tests? (Section V), and 3) What are the common characteristics of these failing tests? (Section VI). We conduct an empirical study on a corpus of 114 Python projects from the Machine Learning domain. We develop a tool, XSEED, which automatically installs each project, runs each test a pre-specified number of times (500 in our evaluation) 2022
Modified on	16/02/2023 11:50

Name	To Seed or Not to Seed~ An Empirical Analysis of Usage of Seeds for Testing in Machine Learning Projects
Number of Coding References	5
Number of Codes Coding	2
Coverage	6.08%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Should Seeds be Used in Tests? Based on our experience with the tests and projects that we study, we develop a set of general recommendations or best practices for using (or not using) seeds for testing. When to use fixed seeds. Developers should ideally use seeds when testing for exact reproducibility of some functionality in their code. For instance, this may include APIs that implement functionality/wrappers related to random number generators (such as Listing 2 from google/TensorNetwork). Another example is a test in TensorFlow/ranking [7] that tests a sorting algorithm that randomly shuffles ties. The test uses two different seeds to check whether the algorithm outputs two sequences with two different orderings of tied elements. Randomize and Log seeds for variability and reproducibility. During our discussions with developers, some mentioned that they use fixed seeds in their tests for better reproducibility of test failures. A better approach might be to randomize and log the seed. This would ensure that the code under test exhibits different sequences of computations and test failures can still be reproduced. Interestingly, we find one such example in pytorch/serve [5]: random.seed(datetime.datetime.now()). Further, in one case, the developers randomized the seed after our bug report [64]. However, the test may also randomly fail (not due to a bug). We next discuss a strategy for mitigating this risk. Use Test Re-run on failure instead of setting seeds. Instead of setting seeds, developers can choose to re-run the test on failure (e.g., using Python's flaky plugin [28]). This has a few distinct advantages: 1) CI builds will not be blocked due to intermittent failures, reducing the burden on developers, 2) intermittent failures can still be logged allowing developers to investigate them later, and 3) if test still fails after re-run(s), developers can use it as signal for immediate investigation. The expected cost of re-runs will be low if the test rarely fails [24]. Finding optimal test settings to minimize flakiness. In general, developers can use the available tools: TERA and FLEX, to find both optimal hyper-parameters and assertion bounds for their tests. In our study, we were able to fix a majority (more than 78%) of the selected tests using these two techniques (or their combination). The positive response for our fixes also demonstrates that developers welcome such changes. Future research can perhaps look into making these tools more approachable and cost-efficient for developers so that they can easily integrate them into their workflow.
Modified on	16/02/2023 11:51

Name	To Seed or Not to Seed~ An Empirical Analysis of Usage of Seeds for Testing in Machine Learning Projects
Number of Coding References	5
Number of Codes Coding	2
Coverage	6.08%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	passes and fails non-deterministically for the same version of code. Test flakiness undermines the reliability of test results and puts additional burden on developers for investigating test failures (even in absence of bugs). Flakiness makes it difficult for developers to distinguish real test failures due to programming errors (bugs) from noisy executions due to randomness. To minimize test flakiness, developers need to make vari- ous non-trivial choices such as choosing the optimal hyperparameters for the ML algorithm under test [24] and a reasonable assertion bound [26]. However, without a systematic approach, it is hard for a developer to get these settings right. Hence, to mitigate flakiness, they tend to set the seeds for the random number generators that are used by the code under test. Setting the seeds can make the test execution deterministic and alleviate the developer from dealing with randomness. However, it is unknown whether this is always the best approach or if there are alternative ways to mitigate flakiness. Setting seeds can also lead to unintended consequences. For instance, fixing the seed(s) limits the sequence of computations exercised by the code under test. Hence, developers may potentially miss bugs in code under test that are triggered by other sequences, thus reducing the fault-detecting effectiveness of the test [25]. Prior work [25] has shown that algorithmic randomness is a major contributor to flakiness in ML projects. Further, it is known that tests for ML algorithms (prone to flaky failures) are typically more time-consuming than other tests in the suite, often consuming more than 80% of test time [24]. This makes it important to study such tests – and the role of seeds – in greater depth and scale than previous works.
Modified on	16/02/2023 11:49

Name	To Seed or Not to Seed~ An Empirical Analysis of Usage of Seeds for Testing in Machine Learning Projects
Number of Coding References	5
Number of Codes Coding	2
Coverage	6.08%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	Testing implementations of ML algorithms is challenging. Many ML algorithms are inherently random in nature – multiple executions of the algorithm with same inputs and configurations may often lead to varying results. Moreover, the lack of proper test oracles further complicates the testing scenario. A natural consequence of randomness is test flakiness, i.e., when a test
Modified on	16/02/2023 11:49

Name	To Seed or Not to Seed~ An Empirical Analysis of Usage of Seeds for Testing in Machine Learning Projects Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Towards a Common Testing Terminology for Software Engineering and Data Science Experts
Number of Coding References	4
Number of Codes Coding	2
Coverage	6.27%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	
Modified on	03/02/2022 11:10

Name	Towards a Common Testing Terminology for Software Engineering and Data Science Experts
Number of Coding References	4
Number of Codes Coding	2
Coverage	6.27%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	We see the potential to exploit experiences and concepts from the field of classical software testing for the QA of AI-based systems and components. To this end, collaboration and direct exchange between experts from both fields are important. This is, however, impeded by different terminologies and meaning of terms, which leads to misunderstandings and makes it more difficult to relate to work from the respective other field. Contribution: In this paper, we make a first step towards a common terminology. We use established terms from classical software testing as a basis to map corresponding concepts from the field of AI to it, pointing out differences and key challenges in transferring known concepts. The proposed mapping was developed in an interdisciplinary collaboration among the authors, who have many years of experience in at least one of the two fields, partly in both. We intend this to be a stimulus and a basis for discussions aimed at building a common understanding between experts of both fields.
Modified on	03/02/2022 11:09

Name	Towards a Common Testing Terminology for Software Engineering and Data Science Experts
Number of Coding References	4
Number of Codes Coding	2
Coverage	6.27%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	Analytical quality assurance, especially testing, is an integral part of software-intensive system development. With the increased usage of Artificial Intelligence (AI) and Machine Learning (ML) as part of such systems, this becomes more difficult as well-understood software testing approaches cannot be applied directly to the AI-enabled parts of the system.
Modified on	03/02/2022 11:07

Name	Towards a Common Testing Terminology for Software Engineering and Data Science Experts
Number of Coding References	4
Number of Codes Coding	2
Coverage	6.27%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	31/01/2022 15:14
Coded Text	Due to the increasing relevance of Artificial Intelligence (AI) and Machine Learning (ML) as part of software systems, the question arises how AI/ML-enabled systems, and especially their AI/ML-based components, should be tested. The functionality of such components, which we refer to as data-driven components (DDCs), is not explicitly defined by a specification and implemented by a programmer within the code. Instead, it is given by a – usually complex and not human-understandable – model that is automatically derived from a data sample via a learning algorithm. Due to properties such as limited specification and understandability, the transfer of classical test approaches is not trivial.
Modified on	03/02/2022 11:08

Name	Towards Accountability for Machine Learning Datasets~ Practices from Software Engineering and Infrastructure
Number of Coding References	6
Number of Codes Coding	2
Coverage	2.39%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	03/02/2022 10:47
Coded Text	
Modified on	03/02/2022 10:52

Name	Towards Accountability for Machine Learning Datasets~ Practices from Software Engineering and Infrastructure
Number of Coding References	6
Number of Codes Coding	2
Coverage	2.39%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	03/02/2022 10:47
Coded Text	
Modified on	03/02/2022 10:55

Name	Towards Accountability for Machine Learning Datasets~ Practices from Software Engineering and Infrastructure
Number of Coding References	6
Number of Codes Coding	2
Coverage	2.39%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	03/02/2022 10:47

| **Coded Text** | Datasets that power machine learning are often used, shared, and reused with little visibility into the processes of deliberation that led to their creation. As artificial intelligence systems are increasingly used in high-stakes tasks, system development and deployment practices must be adapted to address the very real consequences of how model development data is constructed and used in practice. This includes greater transparency about data, and accountability for decisions made when developing it. |
| **Modified on** | 03/02/2022 10:47 |

Name	Towards Accountability for Machine Learning Datasets~ Practices from Software Engineering and Infrastructure
Number of Coding References	6
Number of Codes Coding	2
Coverage	2.39%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	03/02/2022 10:47
Coded Text	Despite rapid growth, the disciplines of data-driven decision making—including ML—have come under sustained criticism in recent years due to their tendency to perpetuate and amplify social inequality [13, 44]. Data is frequently identified as a key source of these failures through its role in "bias-laundering" [40, 51, 54, 119, 125]. For example, recent studies have uncovered widespread prevalence of undesirable biases in ML datasets, such as the underrepresentation of minoritized groups [27, 40, 131] and stereotype aligned correlations [28, 51, 72, 155]. Datasets also frequently reflect historical patterns of social injustices, which can subsequently be reproduced by ML systems built from the data. For example, in a recent study examining the datasets underlying predictive policing models deployed in police precincts across the US, the underlying data source was found to reflect racially discriminatory and corrupt policing practices [119]. The norms and standards of data collection within ML have themselves been subject to critique, with scholars identifying insufficient documentation and transparency regarding processes of dataset construction [52, 53, 126], as well as problematic consent practices [114]. The lack of accountability to datafied and surveilled populations as well as groups impacted by data-driven decisions [32] has been further critiqued.
Modified on	03/02/2022 10:51

Name	Towards Accountability for Machine Learning Datasets~ Practices from Software Engineering and Infrastructure
Number of Coding References	6
Number of Codes Coding	2
Coverage	2.39%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Research gap in Maintaining ML systems
Created on	03/02/2022 10:50
Coded Text	as paving the way for achieving human-level artificial general intelligence [7]. However the datasets which machine learning (ML) critically depends on—and which frequently contribute to errors— are often poorly documented, poorly maintained, lacking in answerability, and have opaque creation processes. This paper argues that the development ofML datasets should embrace engineering best practices around visibility and ownership, as a necessary (but not sufficient) requirement for accountability, and as a prerequisite for mitigating harmful impacts.
Modified on	03/02/2022 10:50

Name	Towards Accountability for Machine Learning Datasets~ Practices from Software Engineering and Infrastructure
Number of Coding References	6
Number of Codes Coding	2
Coverage	2.39%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Research gap in Maintaining ML systems
Created on	03/02/2022 10:50
Coded Text	Despite rapid growth, the disciplines of data-driven decision making—including ML—have come under sustained criticism in recent years due to their tendency to perpetuate and amplify social inequality [13, 44]. Data is frequently identified as a key source of these failures through its role in “bias-laundering” [40, 51, 54, 119, 125]. For example, recent studies have uncovered widespread prevalence of undesirable biases in ML datasets, such as the underrepresentation of minoritized groups [27, 40, 131] and stereotype aligned correlations [28, 51, 72, 155]. Datasets also frequently reflect historical patterns of social injustices, which can subsequently be reproduced by ML systems built from the data. For example, in a recent study examining the datasets underlying predictive policing models deployed in police precincts across the US, the underlying data source was found to reflect racially discriminatory and corrupt policing practices [119]. The norms and standards of data collection within ML have themselves been subject to critique, with scholars identifying insufficient documentation and transparency regarding processes of dataset construction [52, 53, 126], as well as problematic consent practices [114]. The lack of accountability to datafied and surveilled populations as well as groups impacted by data-driven decisions [32] has been further critiqued.
Modified on	03/02/2022 10:51

Name	Towards Building Robust DNN Applications~ An Industrial Case Study of Evolutionary Data Augmentation
Number of Coding References	8
Number of Codes Coding	2
Coverage	12.17%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	10/02/2022 12:05
Coded Text	For the image classification task, theWorst ofk and Sensei methods achieved a better robustness for the image classification model than the Random approach. Therefore, we can answer ‘yes’ to RQ1 and confirm the effectiveness of the Worst ofk and Sensei approaches regarding the robustness of the image classification model with both open and industrial data. For the object detection task, the mAP and robustness of the object detection model did not show any significant differences. Therefore, we must answer ‘no’ to RQ2 and cannot state that the Worst of k and Sensei methods contribute to the robustness in the object recognition task when applying the industrial GUI data used in this study. Through a manual investigation using our eyes, we found that some of the bounding boxes did not be learned properly by the object detection model when some or all parts of the bounding boxes extended outside the bounds of the image due to a translation or zoom in. In this case, the implicit conditions by which the human eye can correctly classify a sample were not satisfied. We believe we can solve this problem through one of the following two ways: (1) restricting the range of translation to prevent missing the bounding boxes or (2) removing the bounding boxes that are partially missing from the dataset.
Modified on	11/02/2022 14:41

Name	Towards Building Robust DNN Applications~ An Industrial Case Study of Evolutionary Data Augmentation
Number of Coding References	8
Number of Codes Coding	2
Coverage	12.17%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	10/02/2022 12:05
Coded Text	The major findings in our research are as follows. • DA techniques can improve the robustness of a model in image classification tasks by up to 0.73 pts for industrial GUI datasets. In particular, Sensei improves the robustness of the model by nearly 0.09 pts compared to the other techniques. • There are many challenges in applying DA techniques to object detection tasks (e.g., some realistic variations for image classification incorrectly exclude the bounding boxes); they are highlighted in this study. • Through feedback from developers, we identified two types of demand for a DA technique: (1) maintaining the training speed to avoid slowing the system development and (2) the extensibility for a variety of tasks (e.g., an anomaly detection
Modified on	11/02/2022 14:41

Name	Towards Building Robust DNN Applications~ An Industrial Case Study of Evolutionary Data Augmentation
Number of Coding References	8
Number of Codes Coding	2
Coverage	12.17%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	10/02/2022 12:05
Coded Text	The Worst ofk approach chooses a transformation having the largest value for its loss function among k randomly selected transformations in the training loop. Gao et al. proposed a search-based method called Sensei, which finds an effective transformation with a genetic algorithm [4]. Their evaluation reported that Sensei achieves a higher effectiveness than the Random and Worst ofk approaches in maximizing the robustness of the image
Modified on	11/02/2022 14:40

Name	Towards Building Robust DNN Applications~ An Industrial Case Study of Evolutionary Data Augmentation
Number of Coding References	8
Number of Codes Coding	2
Coverage	12.17%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	10/02/2022 12:05
Coded Text	We asked the developers involved in the in-house ML system development for their feedback. Their opinions are as follows. • A developer involved in creating an anomaly detection system from waveform data stated that such DA techniques can effectively generate anomaly data that are rarely observed in the real world. • A developer who has already utilized the Random DA approach for improving the accuracy of a handwriting recognition system claimed that these DA techniques are expected to go beyond the Random method. • A developer who uses an existing DA technique included in a deep learning framework is concerned that the improvement in robustness will result in a slower learning. Through an in-house trial and interviews with in-house developers, we learned following lessons: • The results of a long training time and feedback regarding the concerns of a comparatively slow system development indicate that DA techniques such as the Worst of k and Sensei approaches should not only to improve the robustness of the classifier but also to maintain the training speed. • The findings regarding the difficulties in extending the DA techniques for an image classification task to an object detection task, and the feedback regarding the need for DA techniques in various domains indicate that the extensibility of the augmentation technique for a variety of tasks is important when designing a DA library.
Modified on	11/02/2022 14:41

Name	Towards Building Robust DNN Applications~ An Industrial Case Study of Evolutionary Data Augmentation
Number of Coding References	8
Number of Codes Coding	2
Coverage	12.17%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	10/02/2022 12:05
Coded Text	We evaluate data augmentation techniques in image classification and object detection tasks using an industrial in-house graphical user interface dataset. As the results indicate, the genetic algorithm-based data augmentation technique outperforms two random-based methods in terms of the robustness of the image classification model. In addition, through this evaluation and interviews with the developers, we learned following two lessons: data augmentation techniques should (1) maintain the training speed to avoid slowing the development and (2) include extensibility for a variety of tasks.
Modified on	11/02/2022 14:40

Name	Towards Building Robust DNN Applications~ An Industrial Case Study of Evolutionary Data Augmentation
Number of Coding References	8
Number of Codes Coding	2
Coverage	12.17%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	03/02/2022 10:47
Coded Text	Data augmentation techniques that increase the amount of training data by adding realistic transformations are used in machine learning to improve the level of accuracy. Recent studies have demonstrated that data augmentation techniques improve the robustness of image classification models with open datasets; however, it has yet to be investigated whether these techniques are effective for industrial datasets. In this study, we investigate the feasibility of data augmentation techniques for industrial use.
Modified on	11/02/2022 14:39

Name	Towards Building Robust DNN Applications~ An Industrial Case Study of Evolutionary Data Augmentation
Number of Coding References	8
Number of Codes Coding	2
Coverage	12.17%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	03/02/2022 10:47
Coded Text	methods perform well for open benchmark datasets, their performance in industrial systems has yet to be evaluated. Therefore, in this study, we investigate the effectiveness of the Worst ofk and Sensei approaches using our industrial graphical user interface (GUI) recognition system and determine the feasibility of these techniques in cases of real industrial use. We evaluate the DA techniques in a stepwise manner using image classification and object detection models because there are differences not only in the data domains but also in the target ML tasks between the existing studies and our proposed approach. The existing studies on Worst ofk and Sensei target image classification tasks using photographic images (e.g., an animal and vehicle image dataset [6] and a traffic sign image dataset [14]), whereas our GUI recognition system targets an object detection task using GUI screenshot images.
Modified on	11/02/2022 14:41

Name	Towards Building Robust DNN Applications~ An Industrial Case Study of Evolutionary Data Augmentation
Number of Coding References	8
Number of Codes Coding	2
Coverage	12.17%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	03/02/2022 10:47
Coded Text	To improve the robustness of an ML model, a number of studies have focused on data augmentation (DA) techniques [3, 4, 12, 17]. DA is a technique for providing data with realistic variations to ML models during the training phase. The variations in such transformations vary by domain. For example, photo images have variations such as an inversion, translation, rotation, zoom, occlusion, brightness, and contrast [4, 12]. In DA, these transformations are often applied randomly to a dataset (hereinafter referred to as the Random approach). Such techniques are implemented in major deep learning frameworks including PyTorch [8] and Keras [2]. Engstrom et al. showed that the 'Worst ofk' method outperforms the Random method in terms of improvement to the robustness of ML models [3].
Modified on	11/02/2022 14:40

Name	Towards CRISP-ML(Q)~ A Machine Learning Process Model with Quality Assurance Methodology
Number of Coding References	11
Number of Codes Coding	3
Coverage	25.29%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	. Modeling The choice of modeling techniques depends on the ML and the business objectives, the data and the boundary conditions of the project the ML application is contributing to. The requirements and constraints that have been defined in Section 3.1 are used as inputs to guide the model selection to a subset of appropriate models. The goal of the modeling phase is to craft one or multiple models that satisfy the given constraints and requirements. Literature research on similar problems: It is best practice to screen the literature (e.g., publications, patents, internal reports) for a comprehensive overview on similar ML tasks, since ML has become an established tool for a wide number of applications. New models can be based on published insights, and previous results can serve as performance baselines. Define quality measures of the model: The modeling strategy is required to have multiple objectives in mind [96]. At least six complementary properties of a model (Table 2) should be evaluated. Besides a performance metric, soft measures such as robustness, explainability, scalability, resource demand, and model complexity need to be evaluated (Table 2). In practical application, explainability [51,97,98] or robustness might be valued more than accuracy. In such cases, the measures can be weighted differently depending on the application. The models could be ranked either by summing up the weighted quality measures to a scalar or finding Pareto optimal models in a multi-objective optimization process [99]. Additionally, the model's fairness [55,56] or trust might have to be assessed and mitigated. Model Selection: There are plenty of ML models and introductory books on classical methods [62,100], and Deep Learning [91] can be used to compare and understand their characteristics. The model selection depends on the data and has to be tailored to the problem. There is no such model that performs the best on all problem classes (No Free Lunch Theorem for ML [101]). It is best practice to start with models of lower capacity, which can serve as baseline, and gradually increase the capacity. Validating each step assures its benefit and avoid unnecessary complexity of the model. Incorporate domain knowledge: In practice, a specialized model for a specific task performs better than a general model for all possible tasks. However, adapting the model to a specific problem involves the risk of incorporating false assumptions and could reduce the solution space to a non-optimal subset. Therefore, it is best practice to validate the incorporated domain knowledge in isolation against a baseline. Adding domain knowledge should always increase the quality of the model, otherwise, it should be removed to avoid false bias. Model training: The trained model depends on the learning problem, and as such, they are tightly coupled. The learning problem contains an objective, optimizer, regularization, and cross-validation [62,91]. The objective of the learning problem depends on the application. Different applications value different aspects and need to be tweaked in alignment with the business success criteria. The objective is a proxy to evaluate the performance of the model. The optimizer defines the learning strategy and how to adapt the parameters of the model to improve the objective. Regularization, which can be incorporated in the objective, optimizer, and in the model itself, is needed to reduce the risk of overfitting and can help to find unique solutions. Cross-validation is performed for feature selection, to optimize the hyperparameters of the model and to test its generalization property to unseen data [102]. Crossvalidation [62] is based on a splitting of historical data in training, validation, and testing Mach. Learn. Knowl. Extr. 2021, 3 402 data, where the latter is used as a proxy for the target environment [103]. Frameworks such as Auto-ML [104,105] or Neural Architecture Search [106] enable the hyper-parameters optimization and the architecture search to be automated. Using unlabeled data and pre-trained models: Labeling data could be very expensive and might limit the available data set size. Unlabeled data might be exploited in the training process, e.g., by performing unsupervised pre-training [107] and semi-supervised learning algorithms [108–110].Complementary transfer learning could be used to pre-train the network on a proxy data set (e.g., from simulations) that resembles the original data to extract common features [111]. Model Compression: Compression or pruning methods could be used to obtain a more compact model. In kernel methods, low rank approximation of the kernel matrix is an essential tool to tackle large scale learning problems [112,113].

	Neural Networks use a different approach [114] by either pruning the network weights [115] or applying a compression scheme on the network weights [116]. Ensemble methods: Ensemble methods train multiple models to perform the decision based on the aggregate decisions of the individual models. The models could be of different types or multiple instantiations of one type. This results in a more fault-tolerant system as the error of one model could be absorbed by the other models. Boosting, Bagging, or Mixture of Experts are mature techniques to aggregate the decision of multiple models [117–119]. In addition, ensemble models are used to compute uncertainty estimates and can highlight areas of low confidence [120,121].
Modified on	28/02/2023 15:36

Name	Towards CRISP-ML(Q)~ A Machine Learning Process Model with Quality Assurance Methodology
Number of Coding References	11
Number of Codes Coding	3
Coverage	25.29%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	3.2. Data Preparation Building on the experience from the preceding data understanding phase, data prepa- ration serves the purpose of producing a data set for the subsequent modeling phase. However, data preparation is not a static phase and backtracking circles from later phases are necessary if, for example, the modeling phase or the deployment phase reveal erroneous data. To path the way towards ML life-cycle in a later phase, methods for data preparation that are suitable for automation as demonstrated by [8] are preferable. 3.2.1. Select Data Feature selection: Selecting a good data representation based on the available measure- ments is one of the challenges to assure the quality of the ML application. It is best practice to discard under-utilized features, as they provide little to no modeling benefit, but offer possible loopholes for errors, i.e., instability of the feature during the operation of the ML application [36]. In addition, the more features are selected, the more samples are necessary. Intuitively, an exponentially increasing number of samples for an increasing number of features is required to prevent the data from becoming sparse in the feature space. This is termed as the curse of dimensionality [61,62]. Thus, it is best practice to select just necessary features. A checklist for the feature selection task is given in [63]. Note that data often forms a manifold of lower dimensions in the feature space and models have to learn this, respectively, [64]. Feature selection methods can be separated into three categories: (1) filter methods select features from data without considering the model, (2) wrapper methods use a learning model to evaluate the significance of the features, and (3) embedded methods combine the feature selection and the classifier construction steps. A detailed explanation and in-depth analysis on the feature selection problem are given in [65–68]. Feature selection could carry the risk of selection bias, that could be reduced when the feature selection is performed within the cross-validation of the model (Section 3.3) to account for all possible combinations [69]. Surveys on the feature selection problem are given in [70,71]. However, the selection of the features should not be relied purely on the validation and test error, but should be analyzed by a domain expert as potential biases might occur due to spurious correlation in the data. Lapuschkin et al. [72,73] showed that classifiers could exploit spurious correlations, here the copyright tag on the horse class, to obtain a remarkable test performance and, thus, fake a false sense of generalization. In such cases, explanation methods [74] could be used to highlight the significance of features (Section 3.4) and analyzed from a human's perspective. Data selection: Discarding samples should be well documented and strictly based on objective quality criteria. However, certain samples might not satisfy the necessary quality, i.e., does not satisfy the requirements defined in Section 3.1.5 and are not plausible and, thus, should be removed from the data set. Mach. Learn. Knowl. Extr. 2021, 3 400 Unbalanced Classes: In cases of unbalanced classes, where the number of samples per class is skewed, different sampling strategies could improve the results. Over-sampling of the minority class and/or under-sampling of the majority class [75–78] have been used. Over-sampling increases the importance of the minority class, but could result in overfitting on the minority class. Under-sampling by removing data points from the majority class has to be done carefully to keep the characteristics of the data and reduce the chance of introducing biases. However, removing points close to the decision boundary or multiple data points from the same cluster should be avoided. Comparing the results of different sampling techniques reduces the risk of introducing bias to the model. 3.2.2. Clean Data Noise reduction: The gathered data often includes, besides the predictive signal, noise and unwanted signals from other sources. Signal processing filters could be used to remove the irrelevant signals from the data and improve the signal-to-noise ratio [79,80]. However, filtering the data should be documented and evaluated because of the risk that an erroneous filter could remove important parts of the signal in the data. Data imputation: To get a complete data set, missing, NaN (Not a Number), and special values could be imputed with a model readable value. Depending on the data and ML task, the values are imputed by mean or median values, interpolated, replaced by a special value symbol [81] (as the pattern of the values could be informative), substituted by model predictions [82], matrix factorization [83] or multiple imputations [84–86], or imputed based on a convex optimization problem [87]. To reduce the risk of introducing substitution artifacts, the performance of the model should be compared between different imputation techniques.

Modified on	28/02/2023 15:36

Name	Towards CRISP-ML(Q)~ A Machine Learning Process Model with Quality Assurance Methodology
Number of Coding References	11
Number of Codes Coding	3
Coverage	25.29%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	Assure Reproducibility A major principle of scientific methods and the characteristics of robust ML applica- tions is reproducibility. However, ML models are difficult to reproduce due to the mostly non-convex and stochastic training procedures and randomized data splits. It has been proposed to distinguish reproducibility on two different levels. First, one has to assure that the method itself is reproducible and secondly its results [122]. Method reproducibility: This task aims at reproducing the model from an extensive description or sharing of the used algorithm, data set, hyper-parameters, and run-time environment (e.g., software versions, hardware, and random seeds [123]). The algorithm should be described in detail, i.e., with (pseudo) code and on the meta-level including the assumptions. Result reproducibility: It is best practice to validate the mean performance and assess the variance of the model on different random seeds [124,125]. Reporting only the top performance of the model [124,126] is common but dubious practice. Large performance variances indicate the sensitivity of the algorithm and question the robustness of the model. Experimental Documentation: Keeping track of the changed model's performance and its causes by precedent model modifications allows model comprehension by addressing which modifications were beneficial and improve the overall model quality. The documentation should contain the listed properties in the method reproducibility task. Tool-based approaches on version control and meta-data handling while experimenting on ML models and hyper-parameters exist [127]. CRISP-ML(Q) example-step 'assure reproducibility': It is favorable to update a ML model every time a deficit in the performance is detected. There is a risk that the ML model cannot be reproduced and hence the deficit in performance cannot be resolved. To mitigated this risk, QA has to address the model reproducibility by tracking and storing relevant model information (e.g., ML source code to train the model, data sets, computation environment). Now, if the ML model requires an update, e.g., in the absence of the developing data scientist, another data scientist can first reproduce the last model and results (as code, data set, environment, etc., are reproducible) and then develop an improved model with a changed code, data set, or environment. Mach. Learn. Knowl. Extr. 2021, 3 403 3.4. Evaluation Validate performance: A risk occurs when information from a test set leak into the validation or even training set. Hence, it is best practice to hold back an additional test set, which is disjointed from the the validation and training set, stored only for a final evaluation and never shipped to any partner to be able to measure the performance metrics (blind-test). The test set should be assembled and curated with caution and ideally by a team of experts that are capable to analyze the correctness and ability to represent real cases. In general, the test set should cover the whole input distribution and consider all invariances, e.g., transformations of the input that do not change the label, in the data. Another major risk is that the test set cannot cover all possible inputs due to the large input dimensionality or rare corner cases, i.e., inputs with low probability of occurring [128–130]. Extensive testing reduces this risk [103]. It is recommended to separate the teams and the procedures collecting the training and the test data to erase dependencies and avoid methodology dependence. Additionally, it is recommended to perform a sliced performance analysis to highlight weak performance on certain classes or time slices. CRISP-ML(Q) example-step 'validate performance': A ML application is trained to detect ten types of noises in sound data. The number of samples per class vary as some noises have been recorded less frequently. A first ML model achieves an overall accuracy on a test set of 95%. After intensive tuning of hyper-parameters, a second model achieves 96%. A sliced performance analysis is used for QA to mitigate risks in model validation. The analysis reveals, that the second model while performing better on frequent classes has lower accuracy on less frequent classes, which results in a better overall accuracy. The second model is discarded, and the model objectives (Section 3.3) are modified to include a minimal accuracy per class. Determine robustness: A major risk occurs if ML applications are not robust (in terms of not being able to perform appropriately), if data is perturbed, e.g., noisy or wrong, or manipulated in advance to fool the system (e.g., adversarial

attacks) as shown by Chang [131]. This requires methods to statistically estimate the model's local and global robustness. One approach is adding different kinds of noisy or falsified input to the data or varying the hyper-parameters to characterize the model's generalization ability and then verify the quality metrics for robustness defined in Section 3.3. This could be for example the failure rate on adversarial attacks. Formal verification approaches [43] and robustness validation methods using cross-validation techniques on historical data [103] exist. For a comprehensive overview on how to assess robustness of neural networks, including the definition of robustness goals and testing, the ISO/IEC technical report 24029 [24] is recommend. Increase explainability for ML practitioner & end user: Explainability of a model
helps to find errors and allows strategies, e.g., by enriching the data set, to improve the overall performance [132]. In practice, inherently interpretable models are not necessary inferior to complex models in case of structured input data with meaningful features [97]. To achieve explainability and gain a deeper understanding of what a model has already learned and to avoid spurious correlations [73], it is best practice to carefully observe the features which impact the model's prediction the most and check whether they are plausible from a domain expert's point of view [133–135]. Moreover, case studies have shown that explainability helps to increase trust and users' acceptance [136,137] and could guide humans in ML-assisted decisions [98]. Reference [138] lever the value of an explainable framework in a real life example on time series data. Unified frameworks to explore model explainability are available (e.g., [139,140]). With emerging regulations for ML applications, explainability can be a prerequisite for the admission of a ML application in a specific domain. An ongoing topic of research is a standard procedure to measure and compare approaches for explainability [141]. Compare results with defined success criteria: Finally, domain and ML experts must decide if the model can be deployed. Therefore, it is best practice to document the results of the evaluation phase and compare them to the business and ML success criteria defined Mach. Learn. Knowl. Extr. 2021, 3 404
in Section 3.1.2. If the success criteria are not met, one might backtrack to earlier phases (modeling or even data preparation) or stop the project. Identified limitations of robustness and explainability during evaluation might require an update of the risk assessment (e.g., FMEA [40,41]) and might also lead to backtracking to the modeling phase or stopping the project.
3.5. Deployment The deployment phase of a ML model is characterized by its practical use in the
designated field of application. Define inference hardware: Choose the hardware based on the requirements defined
in Section 3.1.3 or align with an existing hardware. While cloud services offer scalable computation resources, embedded system have hard constraints. ML specific options are, e.g., to optimize towards the target hardware [142] regarding CPU and GPU availability, to optimize towards the target operation system (demonstrated for Android and iOS by [143]) or to optimize the ML workload for a specific platform [144]. Monitoring and maintenance (Section 3.6) need be considered in the overall architecture. Model evaluation under production condition: The risk persists that the production
data does not resemble the training data. Previous assumptions on the training data might not hold in production and the hardware that gathered the data might differ. Therefore, it is best practice to evaluate the performance of the model under incrementally increasing production conditions by iteratively running the tasks in Section 3.4. On each incremental step, the model has to be calibrated to the deployed hardware and the test environment. This allows identifying wrong assumptions on the deployed environment and the causes of model degradation. Domain adaptation techniques can be applied [145,146] to enhance the generalization ability of the model. This step will also give a first indication whether the business and economic success criteria defined in Section 3.1.2, could be met. CRISP-ML(Q) example-step 'model evaluation under production condition': A ML
model for an optical quality check in production is under development to detect several classes of defects. Training data with images from mobile devices is available, whereas in the target environment an industrial camera is used. The use of different cameras carry the risk of a performance gap in the target environment. A production test for QA is advanced with a reduced model trained on only one defect. As the performance does not meet the model objectives defined in Section 3.3, an iteration to the data preparation phase (Section 3.2) to align images from source and target domain is planned. Assure user acceptance and usability: Even after passing all evaluation steps, there
might be the risk that the user acceptance and the usability of the model is underwhelming. The model might be incomprehensible and or does not cover corner cases. It is best practice to build a prototype and run an field test with end users [103]. Examine the acceptance, usage rate, and the user experience. A user guide and disclaimer shall be provided to the end users to explain the system's functionality and limits. Minimize the risks of unforeseen errors: The risks of unforeseen errors and outage
times could cause system shutdowns and a temporary suspension of services. This could lead to user complaints and the declining of user numbers and could reduce the revenue. A fall-back plan, that is activated in case of, e.g., erroneous model updates or detected bugs, can help to tackle the problem. Options are to roll back to a previous version, a pre-defined baseline or a rule-based system. A second option to counteract unforeseen errors is to implement software safety cages that control and limit the outputs of the ML application [147] or even learn safe regions in the state space [148]. Deployment strategy: Even though the model is evaluated rigorously during each
previous step, there is the risk that errors might be undetected through the process. Before rolling out a model, it is best practice to setup an, e.g.,

incremental deployment strategy that includes a pipeline for models and data [96,149]. When cloud architectures are used, strategies can often be aligned on general deployment strategies for cloud software applica-
Mach. Learn. Knowl. Extr. 2021, 3 405
tions [150]. The impact of such erroneous deployments and the cost of fixing errors should be minimized.
3.6. Monitoring and Maintenance With the expansion from knowledge discovery to data-driven applications to infer
real-time decisions, ML models are used over a long period and have a life cycle which has to be managed. The risk of not maintaining the model is the degradation of the performance over time which leads to false predictions and could cause errors in subsequent systems. The main reason for a model to become impaired over time is rooted in the violation of the assumption that the training data and the input data for inference come from the same distribution. The causes of the violations are:
• Non-stationary data distribution: Data distributions change over time and result in a stale training set and, thus, the characteristics of the data distribution are represented incorrectly by the training data. Either a shift in the features and/or in the labels are possible. This degrades the performance of the model over time. The frequency of the changes depends on the domain. Data of the stock market are very volatile, whereas the visual properties of elephants will not change much over the next few years.
• Degradation of hardware: The hardware that the model is deployed on and the sensor hardware will age over time. Wear parts in a system will age and friction characteristics of the system might change. Sensors get noisier or fail over time. This will shift the domain of the system and has to be adapted by the model or by retraining it.
• System updates: Updates on the software or hardware of the system can cause a shift in the environment. For example, the units of a signal got changed during an update. Without notifications, the model would use this scaled input to infer false predictions.
After the underlying problem is known, the necessary methods to circumvent stale
models and assure the quality can be formulated. Two sequential tasks in the maintenance phase are proposed to assure or improve the quality of the model. In the monitor task, the staleness of the model is evaluated and returns whether the model has to be updated or not. Afterward, the model is updated and evaluated to gauge whether the update was successful.
Monitor: Baylor et al. [96] propose to monitor all input signals and notify when an
update has occurred. Therefore, statistics of the incoming data and the predicted labels can be compared to the statistics of the training data. The schema defined in Section 3.1.5 can be used to validate the correctness of the incoming data. Inputs that do not satisfy the schema can be treated as anomalies and denied by the model [96]. Libraries exist to help implement an automatic data validation system [60]. If the labels of the incoming data are known, e.g., in forecasting tasks, the performance of the model can be directly monitored and recorded. An equal approach can be applied to the outputs of the model that underlie a certain distribution if environment conditions are stable and can give an estimate on the number of actions performed when interacting with an environment [36]. The monitoring phase also includes a comparison of the current performance or a predicted future performance [46] with the defined success criteria. The future performance of the application can be predicted using ML. Based on the monitoring results, it can then be decided whether the model should be updated, e.g., if input signals change significantly, the number of anomalies reaches a certain threshold or the performance has reached a lower bound. The decision as to whether the model has to be updated should consider the costs of updating the model and the costs resulting from erroneous predictions due to stale models.
CRISP-ML(Q) example-step ‘monitor’: AML based recommendation system proposes
digital features as part of a software application. As the application programming interface, that provides input data to the model is frequently updated, the risk of model degradation due to changing input is eminent. Close monitoring of the input data of the ML model is
Mach. Learn. Knowl. Extr. 2021, 3 406
implemented as a QA measure. A counter is implemented to monitor input data rejections and if a predefined threshold is exceeded a data scientist is informed automatically. Update: In the updating step, new data is collected to re-train the model under the
changed data distribution. Consider that new data has to be labeled which could be very expensive. Instead of training a completely new model from scratch, it is advised to fine-tune the existing model to new data. It might be necessary to perform some of the modeling steps in Section 3.3 to cope with the changing data distribution. Every update step has to undergo a new evaluation (Section 3.4) before it can be deployed. The performance of the updated model should be compared against the previous versions and could give insights on the time scale of model degradation. It should be noted, that ML systems might influence their own behavior during updates due to direct, e.g., by influencing its future training data selection, or indirect, e.g., via interaction through the world, feedback loops [36]. The risk of positive feedback loops causing system instability has to be addressed, e.g., by not only monitoring, but limiting the actions of the model. In addition, as part of the deployment strategy, a module is needed that tracks the
application usage and performance and handles several deployment strategies like A/B testing [96,149]. The module can, e.g., be set up in form of a microservice [151] or a directed graph [152]. To reduce the risk of serving erroneous models, an automatic or human controlled fallback to a previous model needs to be implemented. The automation of the update strategy can be boosted up to a continuous training and continuous deployment of the ML

	application [96] while covering the defined QA methods. 4. Discussion While ML is a frequently tried technique to challenge established processes with the progress of digital transformation, industry practice experiences difficulties in efficient and effective ML project development, for which three reasons were identified: lack of guidance in the development process, poor data, and poor ML development execution. CRISP-ML(Q) is addressing these issues, covering all development phases that range from project idea formulation to maintaining and monitoring an existing ML application. After extensive review and comparison of existing processes for related development projects, the proposed process model is based on CRISP-DM, which was adapted to cover the application scenario of ML models inferring real-time decisions. Hence, CRISPML(Q) can draw from some advantages of CRISP-DM, in particular, a close relationship to established and proven industry standards that are included in the education of data mining practitioners. In line with the principles of CRISP-DM, the tasks devised by CRISPML(Q) were formulated with the goals of being stable enough to support future modeling techniques and general enough to cover many possible knowledge discovery scenarios. Whilst many tasks are shared with CRISP-DM, this paper contributes technical tasks needed to produce evidence that every step in the ML development process is of sufficient quality to warrant the adoption into business processes. The proposed steps/phases are incomplete for safety-critical applications, as their requirements are dependent on the application domain. Safety-critical systems, e.g., for safety in autonomous driving [153], might require different or additional processes and quality measures. However, as CRISP-ML(Q) is intended to be industry-, tool-, and application-neutral, the selection of quality assurance methodology was limited to rather generic tasks that cannot be a complete set for any ML application domain. The presented methods were also not chosen for their degree of novelty, but rather for having passed the test of time to become best practices in automotive industry projects and academia.
Modified on	28/02/2023 15:37

Name	Towards CRISP-ML(Q)~ A Machine Learning Process Model with Quality Assurance Methodology
Number of Coding References	11
Number of Codes Coding	3
Coverage	25.29%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	CRISP-DM names the data scientist responsible to define the scope of the project. However, in daily business, the separation of domain experts and data scientists carries the risk that the application will not satisfy the business needs. Moreover, the availability of training samples will to a large extent influence the feasibility of the data-based application [34]. It is, therefore, best practice to merge the requirements of the business unit with ML requirements while keeping in mind data related constraints in a joint step. 3.1.2. Success Criteria The success criteria of a ML project should be measured on three different levels: the business success criteria, the ML success criteria, and the economic success criteria. According to the IEEE standard for developing software life cycle processes [19] and Six Sigma [39], the requirement measurable is one of the essential principles of QA methodology. In addition, each success criterion has to be defined in alignment to each other and with respect to the overall system requirements [43] to prevent contradictory objectives. Aligned with the 'Define, Measure, Analyze, Design and Verify' methodology [19], this approach handles risk management in a preventative way to limit the risk of not meeting the success criteria to a reasonable level. Nevertheless, this process model can be applied in a reactive Mach. Learn. Knowl. Extr. 2021, 3 397 way whenever contingencies occur and offers iterative paths to, e.g., redefine objectives in an earlier phase, stop, or postpone the project. Business Success Criteria: Define the purpose and the success criteria of the ML application from a business point of view [10] and identify and classify risks. Dependent on the planned application, the business success criteria can include various perspectives on the project, e.g., operational, technical, and economic criteria [19]. In the sense of QA, the common denominator is to define measurable criteria and deduct ML success criteria in the next step. ML Success Criteria: Translate the business objective into ML success criteria (Table 2) as a measure of technical risks [19]. Unfortunately, there is no such metric that performs best on all ML applications, and the choice of a metric can even privilege one model over the other [44]. As there might be different and often concurrent success criteria, approaches like multi objectives or weighting of success criteria are applicable (Section 3.3). Still, success criteria defined in an theoretical approach can face the issue of a theory-practice gap [8] that might be hard to find in the early phases. It is advised to define a minimum acceptable level of performance to meet the business goals. As an option, a Minimal Viable Product (MVP, [45]) can be defined, and the learnings from the MVP can be used for the reduction of the theory -practice gap in the next iteration of CRISP-ML(Q).
Modified on	28/02/2023 15:35

Name	Towards CRISP-ML(Q)~ A Machine Learning Process Model with Quality Assurance Methodology
Number of Coding References	11
Number of Codes Coding	3
Coverage	25.29%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	Data Collection Costs and time are needed to collect a sufficient amount of consistent data by preparing and merging data from different sources and different formats (Section 3.2). A ML project might be delayed until the data is collected or could even be stopped if the collection of data of sufficient quality (Section 3.1.5) is not feasible. Data version control: Collecting data is not a static task, but rather an iterative task. Modification on the data set (Section 3.2) should be documented to mitigate the risk of obtaining irreproducible or wrong results. Version control on the data is one of the essential tools to assure reproducibility and quality, as it allows errors and unfavorable modifications to be tracked during the development. 3.1.5. Data Quality Verification The following three tasks examine whether the business and ML objectives can be achieved with the given quality of the available data. A ML project is doomed to fail if the data quality is poor. The lack of a certain data quality will trigger the previous data collection task (Section 3.1.4). Data description: The data description forms the basis for the data quality verifica- tion. A description and an exploration of the data is performed to gain insight about the underlying data generation process. The data should be described on a meta-level and by their statistical properties. Furthermore, a technically well-funded visualization of the data should help to understand the data generating process [58]. Information about format, units, and description of the input signals is expanded by domain knowledge. Data requirements: The data requirements can be defined either on the meta-level or directly in the data, and should state the expected conditions of the data, i.e., whether a certain sample is plausible. The requirements can be, e.g., the expected feature values (a range for continuous features or a list for discrete features), the format of the data and the maximum number of missing values. The bounds of the requirements has to be defined carefully to include all possible real world values but discard non-plausible data. Data that does not satisfy the expected conditions could be treated as anomalies and need to be evaluated manually or excluded automatically. To mitigate the risk of anchoring bias in the definition phase discussing the requirements with a domain expert is advised [35]. Documentation of the data requirements could be expressed in the form of a schema [59,60]. Mach. Learn. Knowl. Extr. 2021, 3 399 Data verification: The initial data, added data, but also the production data has to be checked according to the requirements (Section 3.6). In cases where the requirements are not met, the data will be discarded and stored for further manual analysis. This helps to reduce the risk of decreasing the performance of the ML application through adding low-quality data and helps to detect varying data distributions or unstable inputs. To mitigate the risk of insufficient representation of extreme cases, it is best practice to use data exploration techniques to investigate the sample distribution
Modified on	28/02/2023 15:36

Name	Towards CRISP-ML(Q)~ A Machine Learning Process Model with Quality Assurance Methodology
Number of Coding References	11
Number of Codes Coding	3
Coverage	25.29%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	Due to the lack of a process model for ML applications, many project organizations rely on alternative models that are closely related to ML, such as, the Cross-Industry Standard Process model for Data Mining (CRISP-DM) [10–12]. This model is grounded on industrial data mining experience [12] and is considered most suitable for industrial projects among related process models [13]. In fact, CRISP-DM has become the de facto industry standard [14] process model for data mining, with an expanding number of applications [15], e.g., in quality diagnostics [16], marketing [17], and warranty [18]. However, two major shortcomings of CRISP-DM are identified: First, CRISP-DM focuses on data mining and does not cover the application scenario of ML models inferring real-time decisions over a long period of time (Figure 1). The ML model must be adaptable to a changing environment or the model's performance will degrade over time, such that permanent monitoring and maintenance of the ML model is required after the deployment. A data B data mining process Training* data output (labels) Inference data model machine learning *supervised model inferred output machine learning application Figure 1. Difference between (A) data mining processes and (B) machine learning applications. Secondly and more concerning, CRISP-DM lacks guidance on Quality Assurance (QA) methodology ([11]). This deficit is particularly evident in comparison to standards in the area of information technology [19], but also apparent in alternative process models for data mining [20,21]. In the context of process models for ML, quality is not only defined by the product's fitness for purpose [14], but the quality of the task executions in any phase during the development of a ML application. This ensures that errors are caught as early as possible to minimize costs in the later stages of the development process. The paper provides two contributions addressing the mentioned shortcomings: In particular, the first shortcoming is addressed by deriving an end-to-end process model for the development of practical ML applications that covers all relevant phases in the life-cycle of a ML application, using CRISP-DM as a basis, but enlarging the scope with relevant phases supported by literature. The relevance for a process model is motivated by standards in the field of information technology that are proven in use, but do not cover ML specifics (e.g., IEEE 1074-1997 [19]). The model follows the principles of CRISP-DM, in particular by keeping the model industry- and application-neutral, but is modified to the particular requirements of ML applications. The second shortcoming is addressed by anchoring QA methodology in the proposed process model. The QA methodology is adopted from widespread standards for quality assurance (e.g., IEEE 730-1998 [22]), particularly building on the principle of 'risk based thinking' (DIN EN ISO 9001 [23]). The risk based process is kept generic to be industry and application-neutral and is summarized in a flow chart [24] to give a visual understanding. In this work, risk management is included early in the ML project as proposed by [25] in a preventative way (in contrast to reactive risk management that defines actions for 0 101 extract information patterns & knowledge Mach. Learn. Knowl. Extr. 2021, 3 394

	contingencies [19]). The focus of the QA methodology is primarily on the technical tasks needed to produce evidence that every step in the development process is of sufficient quality to warrant adoption into business processes. The necessary infrastructure is not covered in the process model to be tool, domain, and technology agnostic. Examples on possible infrastructure is given in [26,27]. The following second section describes related work and ongoing research in the development of process models for machine learning applications. In the third chapter, the tasks and QA methodology are introduced for each process phase. Finally, a conclusion and an outlook are given in the fourth chapter.
Modified on	28/02/2023 15:35
Name	Towards CRISP-ML(Q)~ A Machine Learning Process Model with Quality Assurance Methodology
Number of Coding References	11
Number of Codes Coding	3
Coverage	25.29%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	Feature engineering: New features could be derived from existing ones based on domain knowledge. This could be, for example, the transformation of the features from the time domain into the frequency domain, discretization of continuous features into bins or augmenting the features with additional features based on the existing ones. In addition, there are several generic feature construction methods, such as clustering [88], dimensional reduction methods such as Kernel-PCA [89], or auto-encoders [90]. Nominal features and labels should be transformed into a one-hot encoding, while ordinal features and labels are transformed into numerical values. However, the engineered features should be compared against a baseline to assess the utility of the feature. Underutilized features should be removed. Models that construct the feature representation as part of the learning process, e.g., neural networks, avoid the feature engineering steps [91]. Data augmentation: Data augmentation utilizes known invariances in the data to perform a label preserving transformation to construct new data. The transformations could either be performed in the feature space [76] or input space, such as applying rotation, elastic deformation, or Gaussian noise to an image [92]. Data could also be augmented on a meta-level, such as switching the scenery from a sunny day to a rainy day. This expands the data set with additional samples and allows the model to capture those invariances. CRISP-ML(Q) example-step 'data augmentation': A ML application is developed to optically detect the position and entity of a metal part on a conveyor and needs to be invariant to different illuminations and backgrounds. Therefore, a data set is synthetically generated, exposing the reflective part with different lighting, backgrounds, and noise levels in a digital tool [93]. The risk of introducing artifacts in the input space is eminent. To mitigate the risk, explanation methods like heat maps [72] are used for QA to verify the model's intended behavior. 3.2.4. Standardize Data File format: Some ML tools require specific variable or input types (data syntax). Indeed, in practice, the comma separated values (CSV) format is the most generic standard Mach. Learn. Knowl. Extr. 2021, 3 401 (RFC 4180). ISO 8000 recommends the use of SI units according to the International System of Quantities. Defining a fix set of standards and units, helps to avoid the risks of errors in the merging process and further in detecting erroneous data (Section 3.1.5). Normalization: Without proper normalization, the features could be defined on differ- ent scales and might lead to strong bias to features on larger scales. In addition, normalized features lead to faster convergence rates in neural networks than without [94,95]. Note that the normalization, applied to the training set has to be applied also to the test set using the same normalization parameters.
Modified on	28/02/2023 15:36

Name	Towards CRISP-ML(Q)~ A Machine Learning Process Model with Quality Assurance Methodology
Number of Coding References	11
Number of Codes Coding	3
Coverage	25.29%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	Machine learning is an established and frequently used technique in industry and academia, but a standard process model to improve success and efficiency of machine learning applications is still missing. Project organizations and machine learning practitioners face manifold challenges and risks when developing machine learning applications and have a need for guidance to meet business expectations. This paper therefore proposes a process model for the development of machine learning applications, covering six phases from defining the scope to maintaining the deployed machine learning application. Business and data understanding are executed simultaneously in the first phase, as both have considerable impact on the feasibility of the project. The next phases are comprised of data preparation, modeling, evaluation, and deployment. Special focus is applied to the last phase, as a model running in changing real-time environments requires close monitoring and maintenance to reduce the risk of performance degradation over time. With each task of the process, this work proposes quality assurance methodology that is suitable to address challenges in machine learning development that are identified in the form of risks. The methodology is drawn from practical experience and scientific literature, and has proven to be general and stable. The process model expands on CRISP-DM, a data mining process model that enjoys strong industry support, but fails to address machine learning specific tasks. The presented work proposes an industry- and applicationneutral process model tailored for machine learning applications with a focus on technical tasks for quality assurance.
Modified on	28/02/2023 15:35

Name	Towards CRISP-ML(Q)~ A Machine Learning Process Model with Quality Assurance Methodology
Number of Coding References	11
Number of Codes Coding	3
Coverage	25.29%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	04/02/2022 13:51
Coded Text	Quality Assurance in Machine Learning Projects A process model is proposed: CRoss-Industry Standard Process model for the devel- opment of Machine Learning applications with Quality assurance methodology (CRISPML(Q)). The name is chosen to highlight its compatibility to CRISP-DM. It is designed for the development of machine applications, i.e., application scenarios where a ML model is deployed and maintained as part of a product or service (Figure 1). CRISP-ML(Q) is an iterative model, consequently, the step back to a prior phase or step (e.g., from the modeling phase back to data collection) is an essential part of the process model. As a first contribution, CRISP-ML(Q) covers a monitoring and maintenance phase to address risks of model degradation in a changing environment. This extends the scope of the process model as compared to CRISP-DM and related process models [34,35] (Table 1). Moreover, business and data understanding are merged into a single phase because industry practice has taught that these two activities, which are separate in CRISP-DM, are strongly intertwined, since business objectives can be derived or changed based on available data (Table 1). A similar approach has been outlined in the W-Model [42]. As a second contribution, quality assurance methodology is introduced in each phase and task of the process model, giving a reasonable degree of confidence that the ML application acquires the expected quality throughout the development process [22]. The CRISP-ML(Q) approach for QA follows a generic quality assurance methodology (Figure 2) founded on 'risk based thinking' (DIN EN ISO 9001 [23]). For every CRISP-ML(Q) phase, requirements and constraints are defined supported by measurable metrics (Six Sigma [39]). Then, steps and tasks are instantiated for the specific application followed by the identification of taskspecific risks. Risks are checked for feasibility, e.g., by combining severity and probability of occurrence [40]. If risks are not feasible, appropriate QA tasks are chosen to mitigate the risks. While general risk management has diverse disciplines [19], this approach focuses on risks that affect the efficiency and success of the ML application and require technical Breck et al. [35] - Data Infrastructure Mach. Learn. Knowl. Extr. 2021, 3 396 tasks for risk mitigation. Whenever possible, measurable metrics (Six Sigma [39]) are implemented and checked for compliance with the defined objectives. In what follows, selected tasks from CRISP-ML(Q) are described and QA methodology is proposed to determine whether these tasks were performed according to current standards from industry best practice and academic literature, which have proven to be general and stable, and are therefore suitable to mitigate the task specific risks. The selection reflects tasks and methods that are considered the most important. It is noted that the processes and QA methods in this document are not designed for safety-critical systems. Safety-critical systems might require different or additional processes and quality measures.
Modified on	28/02/2023 15:35

Name	Towards CRISP-ML(Q)~ A Machine Learning Process Model with Quality Assurance Methodology
Number of Coding References	11
Number of Codes Coding	3
Coverage	25.29%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	07/02/2022 11:05
Coded Text	Machine learning is an established and frequently used technique in industry and academia, but a standard process model to improve success and efficiency of machine learning applications is still missing. Project organizations and machine learning practitioners face manifold challenges and risks when developing machine learning applications and have a need for guidance to meet business expectations. This paper therefore proposes a process model for the development of machine learning applications, covering six phases from defining the scope to maintaining the deployed machine learning application. Business and data understanding are executed simultaneously in the first phase, as both have considerable impact on the feasibility of the project. The next phases are comprised of data preparation, modeling, evaluation, and deployment. Special focus is applied to the last phase, as a model running in changing real-time environments requires close monitoring and maintenance to reduce the risk of performance degradation over time. With each task of the process, this work proposes quality assurance methodology that is suitable to address challenges in machine learning development that are identified in the form of risks. The methodology is drawn from practical experience and scientific literature, and has proven to be general and stable. The process model expands on CRISP-DM, a data mining process model that enjoys strong industry support, but fails to address machine learning specific tasks. The presented work proposes an industry- and applicationneutral process model tailored for machine learning applications with a focus on technical tasks for quality assurance.
Modified on	28/02/2023 15:34

Name	Towards CRISP-ML(Q)~ A Machine Learning Process Model with Quality Assurance Methodology
Number of Coding References	11
Number of Codes Coding	3
Coverage	25.29%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	04/02/2022 13:45
Coded Text	Many industries, such as manufacturing [1,2], personal transportation [3], and health- Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https:// creativecommons.org/licenses/by/ 4.0/). care [4,5], are currently undergoing a process of digital transformation, challenging established processes with machine learning driven approaches. The expanding demand is highlighted by the Gartner report [6], claiming that organizations expect to double the number of Machine Learning (ML) projects within a year. However, 75 to 85 percent of practical ML projects currently do not match their spon- sors' expectations, according to surveys of leading technology companies [7]. Reference [8] name data and software quality among others as the key challenges in the machine learning life cycle. Another reason is the lack of guidance through standards and development process models specific to ML applications. Industrial organizations, in particular, rely Mach. Learn. Knowl. Extr. 2021, 3, 392–413. https://doi.org/10.3390/make3020020 https://www.mdpi.com/journal/make Mach. Learn. Knowl. Extr. 2021, 3 393 heavily on standards to guarantee a consistent quality of their products or services. A Japanese industry consortium (QA4AI) was founded to address those needs [9].
Modified on	28/02/2023 15:35

Name	Towards CRISP-ML(Q)~ A Machine Learning Process Model with Quality Assurance Methodology Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Towards MLOps in Mobile Development with a Plug-in Architecture for Data Analytics
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Towards MLOps in Mobile Development with a Plug-in Architecture for Data Analytics Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Towards MLOps~ A Case Study of ML Pipeline Platform
Number of Coding References	5
Number of Codes Coding	2
Coverage	5.59%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 10:15
Coded Text	
Modified on	03/02/2022 10:32

Name	Towards MLOps~ A Case Study of ML Pipeline Platform
Number of Coding References	5
Number of Codes Coding	2
Coverage	5.59%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 10:15
Coded Text	ML platforms such as TFX [4] and ModelOps [20] address the above concerns. They provide end-to-end lifecycle management for ML applications and systems by offering a set of key components that are separately responsible for tasks such as data preprocessing, model training, model evaluation, and model serving. Based on the pluggable and customizable components, these platforms intend to provide a generic solution for multiple development scenarios that need to accomplish different ML tasks. Through configuring and linking these components, the ML workflows can be orchestrated into ML pipelines that are execution-supported on these platforms. While there is a trend of training ML models on the cloud with abundant GPU-backed VMs and containers, these platforms are able to carry out ML pipelines in hybrid environments.
Modified on	03/02/2022 10:31

Name	Towards MLOps~ A Case Study of ML Pipeline Platform
Number of Coding References	5
Number of Codes Coding	2
Coverage	5.59%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 10:15
Coded Text	Some platforms, such as TensorFlow Extended (TFX), ModelOps, and Kubeflow [4, 6, 20], have provided the solutions for the end-to-end lifecycle management of ML applications by orchestrating its phases into multistep ML pipelines. Several key components and utilities are offered to manage data storage, access control, pipeline execution, job scheduling, and performance monitoring. ModelOps and Kubeflow also provide pipeline configuration or Software Development Kit (SDK) for developers to describe the ML workflows.
Modified on	03/02/2022 10:30

Name	Towards MLOps~ A Case Study of ML Pipeline Platform
Number of Coding References	5
Number of Codes Coding	2
Coverage	5.59%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 10:17
Coded Text	Building the workflow from data preprocessing to application runtime monitoring can be time-consuming and error-prone, automating this process can help users focus on the ML application development. Besides, when new training data arrives or model performance degradation is detected, the computational models need to be retrained and reserved, which requires effective feedback loops from the monitoring system to previous phases.
Modified on	03/02/2022 10:31

Name	Towards MLOps~ A Case Study of ML Pipeline Platform
Number of Coding References	5
Number of Codes Coding	2
Coverage	5.59%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 10:17
Coded Text	The development and deployment of machine learning (ML) applications differ significantly from traditional applications in many ways, which have led to an increasing need for efficient and reliable production of ML applications and supported infrastructures. Though platforms such as TensorFlow Extended (TFX), ModelOps, and Kubeflow have provided end-toend lifecycle management for ML applications by orchestrating its phases into multistep ML pipelines, their performance is still uncertain. To address this, we built a functional ML platform with DevOps capability from existing continuous integration (CI) or continuous delivery (CD) tools and Kubeflow, constructed and ran ML pipelines to train models with different layers and hyperparameters while time and computing resources consumed were recorded. On this basis, we analyzed the time and resource consumption of each step in the ML pipeline, explored the consumption concerning the ML platform and computational models, and proposed potential performance bottlenecks such as GPU utilization. Our work provides a valuable reference for ML pipeline platform construction in practice.
Modified on	03/02/2022 10:28

Name	Towards MLOps~ A Framework and Maturity Model
Number of Coding References	11
Number of Codes Coding	7
Coverage	9.76%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 10:15
Coded Text	
Modified on	03/02/2022 10:26

Name	Towards MLOps~ A Framework and Maturity Model
Number of Coding References	11
Number of Codes Coding	7
Coverage	9.76%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 10:15
Coded Text	Based on the SLR and the GLR, we present a maturity model in which we outline four stages in which companies evolve when adopting MLOps practices. The four stages are a) Automated Data Collection b) Automated Model Deployment c) Semi-automated Model Monitoring and d) Fully-automated Model Monitoring. These stages capture key transition points in the adoption of MLOps in practice. Below, we detail each MLOps stage and preconditions for a company to reach this stage.
Modified on	03/02/2022 10:25

Name	Towards MLOps~ A Framework and Maturity Model
Number of Coding References	11
Number of Codes Coding	7
Coverage	9.76%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 10:15
Coded Text	MLOPS FRAMEWORK AND MATURITY MODEL Based on the SLR and the GLR, we derive an MLOps framework that identifies the activities involved in MLOps adoption. Figure 2 depicts the MLOps framework. The entire framework is divided into three pipelines: a) Data Pipeline b) Modeling Pipeline and c) Release Pipeline.
Modified on	03/02/2022 10:25

Name	Towards MLOps~ A Framework and Maturity Model
Number of Coding References	11
Number of Codes Coding	7
Coverage	9.76%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 10:15
Coded Text	The adoption of continuous software engineering practices such as DevOps (Development and Operations) in business operations has contributed to significantly shorter software development and deployment cycles. Recently, the term MLOps (Machine Learning Operations) has gained increasing interest as a practice that brings together data scientists and operations teams. However, the adoption of MLOps in practice is still in its infancy and there are few common guidelines on how to effectively integrate it into existing software development practices. In this paper, we conduct a systematic literature review and a grey literature review to derive a framework that identifies the activities involved in the adoption of MLOps and the stages in which companies evolve as they become more mature and advanced. We validate this framework in three case companies and show how they have managed to adopt and integrate MLOps in their large-scale software development companies.
Modified on	03/02/2022 10:15

Name	Towards MLOps~ A Framework and Maturity Model
Number of Coding References	11
Number of Codes Coding	7
Coverage	9.76%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 10:15
Coded Text	The use of MLOps enables automation, versioning, reproducibility, etc., with successful collaboration of required skills such as data engineer, data scientist, ML engineer/developer [40] [29]. For example, data scientists must specialize in SE skills such as modularization, testing, versioning, etc. [36]. Supporting processes formalized in policies serve as the basis for governance [31] and can be automated to ensure solution reliability and compliance [31]. MLOps also support explainability (GDPR regulation [25]) and audit trails [40]
Modified on	03/02/2022 10:25

Name	Towards MLOps~ A Framework and Maturity Model
Number of Coding References	11
Number of Codes Coding	7
Coverage	9.76%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	31/01/2022 14:42
Coded Text	To release ML models, package [41], validate [41] and deploy models [40] to production [41]. When deploying a model to production, it has to be integrated with other models as well as existing applications [30] [41]. When the model is in production, it serves requests. Despite the fact that training is often a batch process, the inferences can be REST endpoint/custom code, streaming engine, micro-batch, etc. [35]. When performance drops, monitor the model [41] and enable the data feedback loop [41] to retrain the models . In a fully mature MLOps context, perform continuous integration and delivery by enabling the CI/CD pipeline and continuous retraining through CT pipeline [41] [31].
Modified on	03/02/2022 10:24

Name	Towards MLOps~ A Framework and Maturity Model
Number of Coding References	11
Number of Codes Coding	7
Coverage	9.76%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	03/02/2022 10:17
Coded Text	Challenges associated with MLOps In our own previous research [16] [17], we have identified a number of challenges when it comes to the business case, data, modeling and deployment of ML or Deep Learning (DL) models. These include high AI costs and expectations, fewer data scientists, need for large datasets, privacy concerns and noisy data, lack of domain experts, labeling issues, increasing feature complexity, improper feature selection, introduction of bias when experimenting with models, highly complex DL models, need for deep DL knowledge, difficulty in determining final model, model execution environment, more hyperparameter settings, and verification and validation. It also includes less DL deployment, integration issues, internal deployment, need for an understandable model, training-serving skew, enduser communication, model drifts, and maintaining robustness. Some of the challenges in MLOps practice [5] include tracking and comparing experiments, lack of version control, difficulty in deploying models, insufficient purchasing budgets and a challenging regulatory environment.
Modified on	03/02/2022 10:17

Name	Towards MLOps~ A Framework and Maturity Model
Number of Coding References	11
Number of Codes Coding	7
Coverage	9.76%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	06/01/2022 15:15
Coded Text	To release ML models, package [41], validate [41] and deploy models [40] to production [41]. When deploying a model to production, it has to be integrated with other models as well as existing applications [30] [41]. When the model is in production, it serves requests. Despite the fact that training is often a batch process, the inferences can be REST endpoint/custom code, streaming engine, micro-batch, etc. [35]. When performance drops, monitor the model [41] and enable the data feedback loop [41] to retrain the models . In a fully mature MLOps context, perform continuous integration and delivery by enabling the CI/CD pipeline and continuous retraining through CT pipeline [41] [31].
Modified on	03/02/2022 10:24

Name	Towards MLOps~ A Framework and Maturity Model
Number of Coding References	11
Number of Codes Coding	7
Coverage	9.76%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	03/02/2022 10:24
Coded Text	In ML model development, provisions should be made to run experiments in parallel, optimize the chosen model with hyperparameters, and finally evaluate the model to ensure that it fits the business case. After versioning, the code is stored in the code repository [42] [23]. The model repository [39] keeps track of the models that will be used in production, and the metadata repository contains all the information about the models (e.g., hyperparameter information).
Modified on	03/02/2022 10:24

Name	Towards MLOps~ A Framework and Maturity Model
Number of Coding References	11
Number of Codes Coding	7
Coverage	9.76%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	04/02/2022 21:53
Coded Text	C. Semi-automated Model Monitoring: At this stage, companies have a manual model monitoring in place. With MLOps, they can attain a transition from manual monitoring to semi-automated model monitoring. Preconditions: To reach this transition, there should be provisions for triggering [43] when performance degrades and availability of tools for diagnostics, performance monitoring and addressing model drift [43] [27] [36]. It also requires 338 Authorized licensed use limited to: UNIVERSITY OF OSLO. Downloaded on January 31,2022 at 12:51:22 UTC from IEEE Xplore. Restrictions apply. automation scripts to manage and monitor models based on drift [38] and ability to perform continuous model tracking [31]. For easy monitoring of models, MLOps professionals has to be provided with visual tools [34], and dedicated and centralized dashboards [38] [27] [28]. It also requires data orchestration pipelines and rule-based data governance to ensure data changes [31], feedback loop and continuous model retraining [43]. There should be also a mechanism to automatically train model in production using fresh data based on live pipeline triggers and feedback loops [38] D. Fully-automated Model Monitoring: The companies have deployment and monitoring of models in place where performance degradation is acknowledged by alert. By utilizing MLOps, they undergo transition towards fully automated monitoring of models. Preconditions: For this transition, company requires CI/CD integration with automation and orchestration [43] and CT pipeline to retrain models when performance degrades [31]. For this transition, there is a need to ensure certification of models [32] [23], governance and security controls [43] [34] [36], model explainability [43] [36], auditing of model usage [34] [43], reproducible workflow and models [36]. There should be mechanisms to perform end-to-end QA test and performance checks [43]. There should be assurance that data security and privacy requirements are built into data pipelines [31] as well as retrain production models on newer data using the data, algorithms and code used to create the original [34].
Modified on	24/02/2022 10:37

Name	Towards MLOps~ A Framework and Maturity Model
Number of Coding References	11
Number of Codes Coding	7
Coverage	9.76%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	03/02/2022 10:26
Coded Text	
Modified on	03/02/2022 10:26

Name	Training Data Debugging for the Fairness of Machine Learning Software
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Training Data Debugging for the Fairness of Machine Learning Software Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	Tunability~ Importance of Hyperparameters of Machine Learning Algorithms
Number of Coding References	9
Number of Codes Coding	2
Coverage	8.51%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	03/02/2022 10:04
Coded Text	
Modified on	11/02/2022 14:55

Name	Tunability~ Importance of Hyperparameters of Machine Learning Algorithms
Number of Coding References	9
Number of Codes Coding	2
Coverage	8.51%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	03/02/2022 10:04

Coded Text	Modern supervised machine learning algorithms involve hyperparameters that have to be set before running them. Options for setting hyperparameters are default values from the software package, manual configuration by the user or configuring them for optimal predictive performance by a tuning procedure. The goal of this paper is two-fold. Firstly, we formalize the problem of tuning from a statistical point of view, define data-based defaults and suggest general measures quantifying the tunability of hyperparameters of algorithms. Secondly, we conduct a large-scale benchmarking study based on 38 datasets from the OpenML platform and six common machine learning algorithms. We apply our measures to assess the tunability of their parameters. Our results yield default values for hyperparameters and enable users to decide whether it is worth conducting a possibly time consuming tuning strategy, to focus on the most important hyperparameters and to choose adequate hyperparameter spaces for tuning.
Modified on	11/02/2022 14:54

Name	Tunability~ Importance of Hyperparameters of Machine Learning Algorithms
Number of Coding References	9
Number of Codes Coding	2
Coverage	8.51%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	03/02/2022 10:04
Coded Text	Our paper provides concise and intuitive definitions for optimal defaults of ML algorithms and the impact of tuning them either jointly, tuning individual parameters or combinations, all based on the general concept of surrogate empirical performance models. Tunability values as defined in our framework are easily and directly interpretable as how much performance can be gained by tuning this hyperparameter?. This allows direct comparability of the tunability values across different algorithms. In an extensive OpenML benchmark, we computed optimal defaults for elastic net, deci- sion tree, k-nearest neighbors, SVM, random forest and xgboost and quantified their tunability and the tunability of their individual parameters. This—to the best of our knowledge— has never been provided before in such a principled manner. Our results are often in line with common knowledge from literature and our method itself now allows an analogous analysis for other or more complex methods. Our framework is based on the concept of default hyperparameter values, which can be seen both as an advantage (default values are a valuable output of the approach) and as an inconvenience (the determination of the default values is an additional analysis step and needed as a reference point for most of our measures). We now compare our method with van Rijn and Hutter (2017). In contrast to us, they apply the functional ANOVA framework from Hutter et al. (2014) on a surrogate random forest to assess the importance of hyperparameters regarding empirical performance of a support vector machine, random forest and adaboost, which results in numerical importance
Modified on	11/02/2022 14:56

Name	Tunability~ Importance of Hyperparameters of Machine Learning Algorithms
Number of Coding References	9
Number of Codes Coding	2
Coverage	8.51%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	03/02/2022 10:04
Coded Text	scores for individual hyperparameters. Their numerical scores are - in our opinion - less directly interpretable, but they do not rely on defaults as a reference point, which one might see as an advantage. They also propose a method for calculating hyperparameter priors, combine it with the tuning procedure hyperband, and assess the performance of this new tuning procedure. In contrast, we define and calculate ranges for all hyperparameters. Setting ranges for the tuning space can be seen as a special case of a prior distribution - the uniform distribution on the specified hyperparameter space. Regarding the experimental setup, we compute more hyperparameter runs (around 2.5 million vs. 250000), but consider only the 38 binary classification datasets of OpenML100 while van Rijn and Hutter (2017) use all the 100 datasets which also contain multiclass datasets. We evaluate the performance of different surrogate models by 10 times repeated 10-fold cross-validation to choose an appropriate model and to assure that it performs reasonably well.
Modified on	11/02/2022 14:56

Name	Tunability~ Importance of Hyperparameters of Machine Learning Algorithms
Number of Coding References	9
Number of Codes Coding	2
Coverage	8.51%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	03/02/2022 10:04
Coded Text	The best hyperparameter value for one parameter i on dataset j, when all other parameters are set to defaults from θ□ := (θ□ 1, ..., θ□ θ(j)□ k), is denoted by i := arg min θ∈Θ,θI=θ□ R(j)(θ). I ∀ I□=i A natural measure for tunability of the i-th parameter on dataset j is then the difference in risk between the above and our default reference configuration: d(j) Furthermore, we define d(j),rel i i := R(j)(θ□) −R(j)(θ(j)□ d(j) = i), for j = 1, ..., m, i = 1, ..., k. (6) d(j) as the fraction of performance gain, when we only i tune parameter i compared to tuning the complete algorithm, on dataset j. Again, one can calculate the mean, the median or quantiles of these two differences over the n datasets, to get a notion of the overall tunability di of this parameter.
Modified on	11/02/2022 14:55

Name	Tunability~ Importance of Hyperparameters of Machine Learning Algorithms
Number of Coding References	9
Number of Codes Coding	2
Coverage	8.51%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	03/02/2022 10:04
Coded Text	Tunability of Hyperparameter Combinations and Joint Gains As an example, Table 4 displays the average tunability di1,i2 of all 2-way hyperparameter combinations for rpart. Obviously, the increased flexibility in tuning a 2-way combination enables larger improvements when compared with the tunability of one of the respective individual parameters. In Table 5 the joint gain of tuning two hyperparameters gi1,i2 instead of only the best as defined in Section 3.5 can be seen. The parameters minsplit and minbucket have the biggest joint effect, which is not very surprising, as they are closely related: minsplit is the minimum number of observations that must exist in a node in order for a split to be attempted and minbucket the minimum number of observations in any terminal leaf node. If a higher value of minsplit than the default performs better on a dataset it is possibly not enough to set it higher without also increasing minbucket, so the strong relationship is quite clear. Again, further figures for other algorithms are available through the shiny app. Another remarkable example is the combination of sample.fraction and min.node.size in ranger: the joint gain is very low and tuning sample.fraction only seems to be enough, which is concordant to the results of Scornet (2018). Moreover, in xgboost the joint gain of nrounds and eta is relatively low, which is not surprising, as these parameters are highly connected with each other (when setting nrounds higher, eta should be set lower and vice versa). 5.5. Hyperparameter Space for Tuning The hyperparameter space for tuning, as defined in Equation (10) in Section 3.6 and based on the 0.05 and 0.95 quantiles, is displayed in Table 3. All optimal defaults are contained in this hyperparameter space while some of the package defaults are not.
Modified on	11/02/2022 14:56

Name	Tunability~ Importance of Hyperparameters of Machine Learning Algorithms
Number of Coding References	9
Number of Codes Coding	2
Coverage	8.51%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	03/02/2022 10:18
Coded Text	Alternatively, one can use hyperparameter tuning strategies, which are data-dependent, second-level optimization procedures (Guyon et al., 2010), which try to minimize the expected generalization error of the inducing algorithm over a hyperparameter search space of considered candidate configurations, usually by evaluating predictions on an independent test set, or by running a resampling scheme such as cross-validation (Bischl et al., 2012). For a recent overview of tuning strategies, see, e.g., Luo (2016). These search strategies range from simple grid or random search (Bergstra and Bengio, 2012) to more complex, iterative procedures such as Bayesian optimization (Hutter et al., 2011; Snoek et al., 2012; Bischl et al., 2017b) or iterated F-racing (Birattari et al., 2010; Lang et al., 2017). In addition to selecting an efficient tuning strategy, the set of tunable hyperparameters and their corresponding ranges, scales and potential prior distributions for subsequent sampling have to be determined by the user. Some hyperparameters might be safely set to default values, if they work well across many different scenarios. Wrong decisions in these areas can inhibit either the quality of the resulting model or at the very least the efficiency and fast convergence of the tuning procedure. This creates a burden for: 1. ML users—Which hyperparameters should be tuned and in which ranges? 2. Designers of ML algorithms—How do I define robust defaults? We argue that many users, especially if they do not have years of practical experience in the field, here often rely on heuristics or spurious knowledge. It should also be noted that designers of fully automated tuning frameworks face at least very similar problems. It is not clear how these questions should be addressed in a data-dependent, automated, optimal and objective manner. In other words, the scientific community not only misses answers to these questions for many algorithms but also a systematic framework, methods and criteria, which are required to answer these questions.
Modified on	11/02/2022 14:54

Name	Tunability~ Importance of Hyperparameters of Machine Learning Algorithms
Number of Coding References	9
Number of Codes Coding	2
Coverage	8.51%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	03/02/2022 10:18
Coded Text	In order to select an appropriate hyperparameter configuration for a specific dataset at hand, users of ML algorithms can resort to default values of hyperparameters that are specified in implementing software packages or manually configure them, for example, based on recommendations from the literature, experience or trial-and-error.
Modified on	11/02/2022 14:54

Name	Tunability~ Importance of Hyperparameters of Machine Learning Algorithms
Number of Coding References	9
Number of Codes Coding	2
Coverage	8.51%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	03/02/2022 10:18
Coded Text	Our study has some limitations that could be addressed in the future: a) We only con- sidered binary classification, where we tried to include a wider variety of datasets from different domains. In principle this is not a restriction as our methods can easily be applied to multiclass classification, regression, survival analysis or even algorithms not from machine learning whose empirical performance is reliably measurable on a problem instance. b) Uniform random sampling of hyperparameters might not scale enough for very high dimensional spaces, and a smarter sequential technique might be in order here, see Bossek et al. (2015) for an potential approach of sampling across problem instances to learn optimal mappings from problem characteristics to algorithm configurations. c) We currently are learning static defaults, which cannot depend on dataset characteristics (like number of features, or further statistical measures). Doing so might improve performance results of optimal defaults considerably, but would require a more complicated approach. A recent paper regarding this topic was published by van Rijn et al. (2018). d) Our approach still needs initial ranges to be set, in order to run our sampling procedure. Only based on these wider ranges we can then compute more precise, closer ranges.
Modified on	11/02/2022 14:56

Name	Ultron-AutoML~ an open-source, distributed, scalable framework for efficient hyper-parameter optimization
Number of Coding References	12
Number of Codes Coding	8
Coverage	6.74%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 10:07
Coded Text	The framework supports the creation of datapipelines to stream batches of shuffled and augmented data from a distributed file system. This comes in handy for t raining Deep Learning models based on self-supervised, semi-supervised or representation learning algorithms over large training datasets. We demonstrate the framework's reliability and efficiency by running a BERT pre-training job over a large training corpus using pre-emptible GPU compute targets. Despite the inherent unreliability of the underlying compute nodes, the framework is able to complete such long running jobs at 30% of the cost
Modified on	03/02/2022 10:07

Name	Ultron-AutoML~ an open-source, distributed, scalable framework for efficient hyper-parameter optimization
Number of Coding References	12
Number of Codes Coding	8
Coverage	6.74%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering
Created on	03/02/2022 10:06
Coded Text	The framework supports the creation of datapipelines to stream batches of shuffled and augmented data from a distributed file system. This comes in handy for t raining Deep Learning models based on self-supervised, semi-supervised or representation learning algorithms over large training datasets. We demonstrate the framework's reliability and efficiency by running a BERT pre-training job over a large training corpus using pre-emptible GPU compute targets. Despite the inherent unreliability of the underlying compute nodes, the framework is able to complete such long running jobs at 30% of the cost
Modified on	03/02/2022 10:06

Name	Ultron-AutoML~ an open-source, distributed, scalable framework for efficient hyper-parameter optimization
Number of Coding References	12
Number of Codes Coding	8
Coverage	6.74%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	03/02/2022 10:04
Coded Text	Hyperparameter Optimization (HPO), also referred to as AutoML in the literature, can be cast as the optimization of an unknown, possibly stochastic, objective function mapping the hyper -parameter search space to a real valued scalar, the ML model's accuracy or any other performance metric on the validation dataset. The search-space can extend beyond algorithm or architecture specific elements to encompass the space of data pre-processing and data-augmentation techniques, feature selections, as well as choice of algorithms. This is sometimes referred to as the CASH (Combined Algorithm Search and Hyper-parameter tuning) problem for which algorithms have been proposed [28], [48]. Neural Architecture Search (NAS) is a special type of HPO where the focus is on algorithm driven design of neural network architecture components or cells [26]. Models trained with architectures composed of these algorithmically designed neural network cells have been shown to outperform their hand-crafted counterparts in image recognition, object detection [57], and semantic segmentation [21], underscoring the practical importance of this field. Random Search [18] and Grid Search are effective HPO strategies when the computational budget is limited or the hyper-parameter search space is high dimensional. Both are easy to implement and completely parallelizable. Random Search is also widely regarded as a good baseline for benchmarking new hyper-parameter optimization algorithms [33]. Bayesian Optimization (BO) is a dominant paradigm for HPO [20], [27], [45]. Here, the objective function is modeled as a Gaussian Process [50], with the Kernel design reflecting assumptions about the objective function's smoothness properties. Under this assumption, the posterior distribution of the validation score for a candidate architecture is a Gaussian
Modified on	03/02/2022 10:09

Name	Ultron-AutoML~ an open-source, distributed, scalable framework for efficient hyper-parameter optimization
Number of Coding References	12
Number of Codes Coding	8
Coverage	6.74%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	03/02/2022 10:04
Coded Text	We present Ultron-AutoML, an open-source, dis- tributed framework for efficient h yper-parameter optimization (HPO) of ML models. Considering that hyper-parameter optimization is compute intensive and time-consuming, the framework has been designed for reliability – the ability to successfully complete an HPO Job in a multi-tenant, failure prone environment, as well as efficiency – c ompleting t he j ob w ith minimum compute cost and wall-clock time. From a user's perspective, the framework emphasizes ease of use and customizability. The user can declaratively specify and execute an HPO Job, while ancillary tasks – containerizing and running the user's scripts, model checkpointing, monitoring progress, parallelization – are handled by the framework.
Modified on	03/02/2022 10:04
Name	Ultron-AutoML~ an open-source, distributed, scalable framework for efficient hyper-parameter optimization
Number of Coding References	12
Number of Codes Coding	8
Coverage	6.74%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	03/02/2022 10:18
Coded Text	Hyperparameter Optimization (HPO), also referred to as AutoML in the literature, can be cast as the optimization of an unknown, possibly stochastic, objective function mapping the hyper -parameter search space to a real valued scalar, the ML model's accuracy or any other performance metric on the validation dataset. The search-space can extend beyond algorithm or architecture specific elements to encompass the space of data pre-processing and data-augmentation techniques, feature selections, as well as choice of algorithms. This is sometimes referred to as the CASH (Combined Algorithm Search and Hyper-parameter tuning) problem for which algorithms have been proposed [28], [48]. Neural Architecture Search (NAS) is a special type of HPO where the focus is on algorithm driven design of neural network architecture components or cells [26]. Models trained with architectures composed of these algorithmically designed neural network cells have been shown to outperform their hand-crafted counterparts in image recognition, object detection [57], and semantic segmentation [21], underscoring the practical importance of this field. Random Search [18] and Grid Search are effective HPO strategies when the computational budget is limited or the hyper-parameter search space is high dimensional. Both are easy to implement and completely parallelizable. Random Search is also widely regarded as a good baseline for benchmarking new hyper-parameter optimization algorithms [33]. Bayesian Optimization (BO) is a dominant paradigm for HPO [20], [27], [45]. Here, the objective function is modeled as a Gaussian Process [50], with the Kernel design reflecting assumptions about the objective function's smoothness properties. Under this assumption, the posterior distribution of the validation score for a candidate architecture is a Gaussian
Modified on	03/02/2022 10:09

Name	Ultron-AutoML~ an open-source, distributed, scalable framework for efficient hyper-parameter optimization
Number of Coding References	12
Number of Codes Coding	8
Coverage	6.74%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	03/02/2022 10:04
Coded Text	Queues are mirrored for high availability. Certain com- ponents require a key value store for which we use Redis [14]. It is made highly available using Redis sentinels. For durability of Redis and RabbitMQ, we rely on persistent volumes mounted on their pods. A high-level view of the process flow of the HPO Job is as follows: 1) The user fires an HTTP POST request containing the HPO Job payload shown in Fig. 2. 2) The Ingress Controller forwards the incoming request to the Entry Point Service, which validates the request and forwards it to the On Demand Containerizer. 3) The On Demand Containerizer creates master and worker images based on the user-supplied code-base. The master and worker images implement the interfaces in Fig. 5, Fig. 4 respectively. Finally, it forwards the request to the Job Agent. 4) The Job Agent acquires resources from the Resource Limiter, launches the master and creates two queues: Work queue, Results queue. 5) The master, on initialization, starts a pool of parallel worker pods. The master and workers co-ordinate via the Work queue, Results queue to run the HPO algorithm to completion. 6) Finally, the Completion Manager deletes the master, associated workers, Results queue and Work queue, marking the end of the HPO Job.
Modified on	03/02/2022 10:11
Name	Ultron-AutoML~ an open-source, distributed, scalable framework for efficient hyper-parameter optimization
Number of Coding References	12
Number of Codes Coding	8
Coverage	6.74%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	03/02/2022 10:04
Coded Text	We present Ultron-AutoML, an open-source, dis- tributed framework for efficient h yper-parameter optimization (HPO) of ML models. Considering that hyper-parameter optimization is compute intensive and time-consuming, the framework has been designed for reliability – the ability to successfully complete an HPO Job in a multi-tenant, failure prone environment, as well as efficiency – c ompleting t he j ob w ith minimum compute cost and wall-clock time. From a user's perspective, the framework emphasizes ease of use and customizability. The user can declaratively specify and execute an HPO Job, while ancillary tasks – containerizing and running the user's scripts, model checkpointing, monitoring progress, parallelization – are handled by the framework.
Modified on	03/02/2022 10:04

Name	Ultron-AutoML~ an open-source, distributed, scalable framework for efficient hyper-parameter optimization
Number of Coding References	12
Number of Codes Coding	8
Coverage	6.74%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	03/02/2022 10:07
Coded Text	
Modified on	03/02/2022 10:12

Name	Ultron-AutoML~ an open-source, distributed, scalable framework for efficient hyper-parameter optimization
Number of Coding References	12
Number of Codes Coding	8
Coverage	6.74%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	03/02/2022 10:07

| **Coded Text** | The framework supports the creation of datapipelines to stream batches of shuffled and augmented data from a distributed file system. This comes in handy for t raining Deep Learning models based on self-supervised, semi-supervised or representation learning algorithms over large training datasets. We demonstrate the framework's reliability and efficiency by running a BERT pre-training job over a large training corpus using pre-emptible GPU compute targets. Despite the inherent unreliability of the underlying compute nodes, the framework is able to complete such long running jobs at 30% of the cost |
| **Modified on** | 03/02/2022 10:07 |

Name	Ultron-AutoML~ an open-source, distributed, scalable framework for efficient hyper-parameter optimization
Number of Coding References	12
Number of Codes Coding	8
Coverage	6.74%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	03/02/2022 10:07
Coded Text	The framework supports the creation of datapipelines to stream batches of shuffled and augmented data from a distributed file system. This comes in handy for t raining Deep Learning models based on self-supervised, semi-supervised or representation learning algorithms over large training datasets. We demonstrate the framework's reliability and efficiency by running a BERT pre-training job over a large training corpus using pre-emptible GPU compute targets. Despite the inherent unreliability of the underlying compute nodes, the framework is able to complete such long running jobs at 30% of the cost
Modified on	03/02/2022 10:07

Name	Ultron-AutoML~ an open-source, distributed, scalable framework for efficient hyper-parameter optimization
Number of Coding References	12
Number of Codes Coding	8
Coverage	6.74%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	03/02/2022 10:19
Coded Text	Queues are mirrored for high availability. Certain com- ponents require a key value store for which we use Redis [14]. It is made highly available using Redis sentinels. For durability of Redis and RabbitMQ, we rely on persistent volumes mounted on their pods. A high-level view of the process flow of the HPO Job is as follows: 1) The user fires an HTTP POST request containing the HPO Job payload shown in Fig. 2. 2) The Ingress Controller forwards the incoming request to the Entry Point Service, which validates the request and forwards it to the On Demand Containerizer. 3) The On Demand Containerizer creates master and worker images based on the user-supplied code-base. The master and worker images implement the interfaces in Fig. 5, Fig. 4 respectively. Finally, it forwards the request to the Job Agent. 4) The Job Agent acquires resources from the Resource Limiter, launches the master and creates two queues: Work queue, Results queue. 5) The master, on initialization, starts a pool of parallel worker pods. The master and workers co-ordinate via the Work queue, Results queue to run the HPO algorithm to completion. 6) Finally, the Completion Manager deletes the master, associated workers, Results queue and Work queue, marking the end of the HPO Job.
Modified on	03/02/2022 10:11

Name	Ultron-AutoML~ an open-source, distributed, scalable framework for efficient hyper-parameter optimization
Number of Coding References	12
Number of Codes Coding	8
Coverage	6.74%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	03/02/2022 10:19
Coded Text	We present Ultron-AutoML, an open-source, dis- tributed framework for efficient h yper-parameter optimization (HPO) of ML models. Considering that hyper-parameter optimization is compute intensive and time-consuming, the framework has been designed for reliability – the ability to successfully complete an HPO Job in a multi-tenant, failure prone environment, as well as efficiency – c ompleting t he j ob w ith minimum compute cost and wall-clock time. From a user's perspective, the framework emphasizes ease of use and customizability. The user can declaratively specify and execute an HPO Job, while ancillary tasks – containerizing and running the user's scripts, model checkpointing, monitoring progress, parallelization – are handled by the framework.
Modified on	03/02/2022 10:04

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	31/01/2022 14:42
Coded Text	
Modified on	31/01/2022 16:19

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	31/01/2022 14:42
Coded Text	5.1 Researchers
Modified on	31/01/2022 16:22

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	31/01/2022 14:42
Coded Text	5.2 Tool Builders
Modified on	31/01/2022 16:22

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems
Created on	31/01/2022 14:42
Coded Text	5.3 Library Vendors
Modified on	31/01/2022 16:22

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Software Mantainability Aspects
Created on	20/06/2022 11:56
Coded Text	~ Observation 3: Developers update ML libraries more frequently than the traditional libraries. Moreover, 41.52% ofML library updates trigger cascading library updates and 60.41% ofthem update a composite library. We also observed that ML library downgrades are more frequent (22.04% of version updates) compared to traditional libraries (10.90% of version updates).
Modified on	21/06/2022 13:47

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Software Mantainability Aspects
Created on	20/06/2022 11:56
Coded Text	~ ~ 2 3 4 5 6 Scikit-Learn ⇒ Pandas Scikit-Learn ⇒ NumPy Matplotlib, Scikit-Learn ⇒ Scipy Scikit-Learn ⇒ Scipy Matplotlib, TensorFlow⇒ NumPy Theano, PyYAML⇒Keras 7 Tensorflow⇒ Matplotlib 8 9 Keras ⇒ TensorFlow Observation 2: 40.10% ofthe projects in our corpus use a combination ofML libraries. 34.91% of these projects share ML libraries between data and model oriented stages while 18.89% of the
Modified on	21/06/2022 13:48

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Software Mantainability Aspects
Created on	20/06/2022 11:56
Coded Text	~ ~ Observation 1: There is an increasing number ofnew projects using ML library(s) each year. ML library usage increases from 1.75% in 2013 to 49.63% in 2018. Furthermore, the amount ofML library API usage increases: In 50% of the projects in our corpus, developers add 0.237 ML APIs per Python file each month. We also observed an increasing number of forks ofML Library
Modified on	21/06/2022 13:48

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Software Mantainability Aspects
Created on	20/06/2022 11:56
Coded Text	5.1 Researchers
Modified on	21/06/2022 13:58

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Software Mantainability Aspects
Created on	20/06/2022 11:56
Coded Text	5.2 Tool Builders
Modified on	21/06/2022 13:59

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Software Mantainability Aspects
Created on	20/06/2022 11:56
Coded Text	5.3 Library Vendors
Modified on	21/06/2022 13:59

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Software Mantainability Aspects
Created on	20/06/2022 11:56
Coded Text	5.5 Software Developers and Educators
Modified on	21/06/2022 13:59

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Software Mantainability Aspects
Created on	20/06/2022 11:56
Coded Text	RQ1: What is the trend in ML library usage? We found that between 2013 and 2018 the ratio of new Software-2.0 projects to other new projects in GitHub each year has increased from 1.75% to 49.63%. In the same period, the ratio ofnew ML library forks to all new forks added in GitHub each year has increased from 0.41% to 3.21%. RQ2: What combinations of libraries do developers use? Among many of our findings, we observed that 40.10% of the Software-2.0 projects use at least two ML libraries to implement various stages of the ML development workflow. RQ3: How do developers update ML library dependencies? Among others, we found that developers update versions of ML libraries more often than the other libraries used in the projects. ML library downgrades are more frequent compared to the other libraries in the projects, and 41.52% ofML library updates trigger cascading library updates. RQ4: What challenges arise when updating ML libraries? We surveyed developers about the challenges of evolving Software-2.0 and found seven common challenges, including binary incompatibility of trained ML models, re-selecting decision thresholds, benchmarking ML models, and supporting multiple versions. RQ5: What help do developers seek for updating ML libraries? Our survey showed that 26.66% of the ML library updates have taken more than a week to complete the update process. Further, developers mostly use the tools that are designed for Software-1.0 when updating ML library versions while preferring six different kinds of new tool support to evolve Software-2.0. RQ6: What challenges arise when retrofitting ML libraries to Software-1.0? We surveyed developers about why they retrofit ML libraries and what challenges they encountered. We found five reasons and eight challenges. Among others, we identified reasons such as augmenting functionalities with ML and replacing existing non-ML techniques with ML techniques. We identified challenges such as gathering data, managing data-pipeline, and adding ML to edge devices.
Modified on	21/06/2022 13:39

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Software Mantainability Aspects
Created on	20/06/2022 11:56
Coded Text	Trend of Creating ML Projects. We group the projects in our corpus of 18,122 Python Projects, by the year in which they were created. For the creation date we consider the first commit date in Git log instead of project creation date given by GitHub search API. (to handle cases where projects moved from a different code management system like GitLab/BitBucket to GitHub). Figure 2 shows library usage distribution in each year. It further shows the ratio of new projects using ML libraries (Software-2.0) to the total number of projects created in each year. While in 2013 only 1.75% new projects used ML library(s), this ratio increased to 49.63% by 2018. To conduct a more accurate and fine-grained trend analysis, we split the data further into monthly intervals and apply a statistical trend test on it. Particularly we used Mann–Kendal trend
Modified on	21/06/2022 13:44

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	31/01/2022 14:42
Coded Text	
Modified on	31/01/2022 16:21

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	31/01/2022 14:42
Coded Text	(i) Data-dependence: ML algorithms infer rules that govern the behavior ofsystems from training data and produce MLmodels that comprise the learned rules. The accuracy of the decisions made by ML models depends on the training data [20]. (ii) Dependence upon Pre-trained model: Software-2.0 run trained ML models to make the decisions in production environments (e.g., identify an email as spam or not spam) [1, 108]. (iii) Rapid evolution: researchers discover new ML algorithms and continue to improve existing ML algorithms. Therefore, ML libraries that implement these algorithms change rapidly releasing new versions very frequently. (iv) Requirement for optimized hardware: ML libraries usually require optimised hardware such as GPU and TPU to efficiently train MLmodels. Due to these unique characteristics, the use ofML libraries deserves special considerations from researchers.
Modified on	31/01/2022 15:53

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	31/01/2022 14:42
Coded Text	~ Percentage 38.33% 31.67% Automated test coverage generation 13.33% Static Analysis tools 11.67% 6.67% 3.33% 48.33% Table 12. Development Effort Effort Less than a day A day Percentage 45.00% 5.00% Few days but less than week 13.33% A week Few weeks but less than month 3.33% Months I don’t know Survey group = Group A. Number of responses = 60. Observation 4: ML library update process consists ofunique challenges such as “retraining,” “benchmarking newMLmodels,” and “updating decision thresholds” due to data/model dependency. Our quantitative analysis shows ML libraries are highly vulnerable to the challenges “supporting multiple library versions” and “dependency hell.”
Modified on	31/01/2022 16:17

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	31/01/2022 14:42
Coded Text	Among others, we found an increasing trend of using ML libraries: The ratio of new Python projects that use ML libraries increased from 2% in 2013 to 50% in 2018. We identify several usage patterns including the following: (i) 36% of the projects use multiple ML libraries to implement various stages of the ML workflows, (ii) developers update ML libraries more often than the traditional libraries, (iii) strict upgrades are the most popular for ML libraries among other update kinds, (iv) ML library updates often result in cascading library updates, and (v) ML libraries are often downgraded (22.04% of cases). We also observed unique challenges when evolving and maintaining Software-2.0 such as (i) binary incompatibility of trained ML models and (ii) benchmarking ML models. Finally, we present actionable implications ofour findings for researchers, tool builders, developers, educators, library vendors, and hardware vendors.
Modified on	31/01/2022 15:47

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	31/01/2022 14:42
Coded Text	Table 9. Top Challenges of ML Library Version Updates
Modified on	31/01/2022 16:16

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	31/01/2022 14:42
Coded Text	The ML libraries continuously and rapidly release updates that fix bugs, enhance features and introduce new functionalities to compete with other ML libraries. For example, Table 1 highlights that TensorFlow released almost 24 versions each year. The respondent S19 said, “Itiseasierto stay up to date with a more stable package like Matplotlib than it is with an evolving package like Scikit-Learn.” Another respondent S18 said, “ML libraries are dependent upon input data and pretrained models, which makes it difficult to update them.” Therefore, understanding the challenges for updating ML libraries that are dependent upon input data and pre-trainedmodels will highlight blind spots in research and tool support for Software-2.0. To gain a deeper insight into the associated challenges, we employ a quantitative and quali- tative method (developer survey). We use 81 survey responses from groups A and B (Table 3), to identify the impediments to update ML dependencies and understand the reasons behind it. We then analyzed the source code to highlight how widespread are the problems identified from the developer surveys. Table 9 summarizes the challenges that are identified from the developer surveys. In
Modified on	31/01/2022 16:15

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	31/01/2022 14:42
Coded Text	there is no systematic study about the use ofML libraries in Software-2.0. How do developers maintain and evolve ML libraries in Software-2.0? What challenges do developers face when using ML libraries? What are the tools the developers are currently using to evolve Software-2.0? What are the new opportunities to better assist Software-2.0 developers? We have very little quantitative and qualitative knowledge to answer these questions.
Modified on	31/01/2022 15:57

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications
Created on	31/01/2022 14:42
Coded Text	Understanding Software-2.0 55:3 (1) Researchers are unaware of the research gaps (i.e., the actual unsolved problems faced by the Software-2.0 developers) that arise with the special attributes ofML libraries, and thus miss opportunities to improve the current state of the art for Software-2.0. (2) Library vendors are unaware of the challenges that their clients face and whether the clients use the library according to their design goals. (3) Hardware vendors are left in the dark without knowing what to optimise on accelerators. (4) Tool builders do not know how to tailor their tools to the actual needs and practices of the developers when maintaining and evolving ML libraries. (5) Developers are not aware of the good and bad practices related to the use ofML libraries.
Modified on	31/01/2022 15:58

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	" Observation 6: We found five reasons and eight challenges for retrofitting ML libraries. Among others, we identified reasons such as add brand new functionality (39.29% of respondents), replace existing non-ML techniques with ML (21.43% of respondents). Among others, we identified challenges such as data gathering/labeling (39.29% of respondents), retrofitting ML to edge devices (17.86% of respondents), and managing data-pipeline (10.71% of respondents).
Modified on	21/06/2022 13:58

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	~ Percentage 38.33% 31.67% Automated test coverage generation 13.33% Static Analysis tools 11.67% 6.67% 3.33% 48.33% Table 12. Development Effort Effort Less than a day A day Percentage 45.00% 5.00% Few days but less than week 13.33% A week Few weeks but less than month 3.33% Months I don’t know Survey group = Group A. Number of responses = 60. Observation 4: ML library update process consists ofunique challenges such as “retraining,” “benchmarking newMLmodels,” and “updating decision thresholds” due to data/model dependency. Our quantitative analysis shows ML libraries are highly vulnerable to the challenges “supporting multiple library versions” and “dependency hell.”
Modified on	21/06/2022 13:50

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	Among others, we found an increasing trend of using ML libraries: The ratio of new Python projects that use ML libraries increased from 2% in 2013 to 50% in 2018. We identify several usage patterns including the following: (i) 36% of the projects use multiple ML libraries to implement various stages of the ML workflows, (ii) developers update ML libraries more often than the traditional libraries, (iii) strict upgrades are the most popular for ML libraries among other update kinds, (iv) ML library updates often result in cascading library updates, and (v) ML libraries are often downgraded (22.04% of cases). We also observed unique challenges when evolving and maintaining Software-2.0 such as (i) binary incompatibility of trained ML models and (ii) benchmarking ML models. Finally, we present actionable implications ofour findings for researchers, tool builders, developers, educators, library vendors, and hardware vendors.
Modified on	21/06/2022 13:36

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	Containers: Developers use containers (Docker) to implement clean independent environments for builds. Respondent S59 said, “We use Docker which Ifeel was very helpful in library update process. Dockerfile clearly tells the instructions to follow and we can easily experiment with changes.” Our survey shows that 6.67% of the projects use a container tool. Applications like Docker were originally designed to support Software-1.0. New open source tools like MLflow10 and Pachyderm11 are designed to specifically assist developers containerize Software-2.0,
Modified on	21/06/2022 13:56

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	Continuous Integration: Continuous integration (CI) automates the compilation, building, and testing ofsoftware. Hilton et al. [60] attribute the proliferation ofCI for identifying bugs, testing across multiple platforms, or enforcing a workflow. Respondent S55 said, “Mostofour dependency version issues after a library update have been caught using CIwith AppVeyor and Travis.”
Modified on	21/06/2022 13:56

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	Despite having lots of similarities with Software-1.0, Software-2.0 inherits additional chal- lenges from ML libraries as follows: (i) Data-dependence: ML algorithms infer rules that govern the behavior ofsystems from training data and produce MLmodels that comprise the learned rules. The accuracy of the decisions made by ML models depends on the training data [20]. (ii) Dependence upon Pre-trained model: Software-2.0 run trained ML models to make the decisions in production environments (e.g., identify an email as spam or not spam) [1, 108]. (iii) Rapid evolution: researchers discover new ML algorithms and continue to improve existing ML algorithms. Therefore, ML libraries that implement these algorithms change rapidly releasing new versions very frequently. (iv) Requirement for optimized hardware: ML libraries usually require optimised hardware such as GPU and TPU to efficiently train MLmodels. Due to these unique characteristics, the use ofML libraries deserves special considerations from researchers.
Modified on	21/06/2022 13:37

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	Incompatibility of serializedMLmodels: Themodel training stage ofML software devel- opment lifecycle is resource-intensive [8]. Survey respondent S15 said, “Library updates are almost always painful. For ML libraries like Scikit-Learn and TensorFlow, issues usually arrive from lack ofbinary compatibility ofserializedmodels, requiring retraining.” A case study done at Twitter [88] describes that the models are trained, serialized and then written to memory,
Modified on	21/06/2022 13:49

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	Library Version Conflicts. Survey respondent S12 said, “it’s difficult to maintain the de- pendencies, because ofthe conflict between two library versions.” Problems arise when using shared libraries on which several other libraries have dependencies but they depend on different and incompatible versions of the shared libraries [35]. For example, a project uses NumPy with TensorFlow and NumPy is a dependency library
Modified on	21/06/2022 13:50

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	RQ6: What challenges arise when retrofitting ML libraries? There is an abundance of Software-1.0 systems that can be improved by transforming to Software-2.0. For example, detecting anomalies in old banking systems, making mobile applications more personalised, suggesting items in old retail systems, and so on. Some of these software systems have been developed before ML gained widespread adoption. While the new software applications already leverage the advantages of ML, we would like older software applications to take advantage of these opportunities. What motivates developers to retrofit ML into their existing systems? What are the challenges of retrofitting ML to an old software system? Answering these questions will motivate Software-1.0 developers to integrate ML techniques into existing software. It will also highlight blind spots in current research and tool development while opening new opportunities for researchers and tool builders.
Modified on	21/06/2022 13:57

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	Selecting decision thresholds: Some ML models (like a SVM classifier for spam emails) work with decision thresholds. Developers conduct several experiments to select decision thresholds to get good tradeoff on certain metrics, such as precision and recall. Sculley et al. [121] observed that changing the training data may force users to re-select a new threshold value from possible threshold values.
Modified on	21/06/2022 13:50

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	Source Code Update: Respondents revealed two API update tools that they use to perform ML library API updates: (i) Dependabord [36] reports the available new version updates and creates pull requests with API update changes (ii) TensorFlow upgrade tool adapts the TensorFlow clients to the API update. For example respondent S44 said, “This update consisted mainly ofrenaming functions and prefixing them with tensorflow.compat.v1 and were done mostly automatically with the tf_upgrade_v2 tool shipped with TensorFlow. This tool had a bunch ofbugs at the time though so it did not work perfectly.”
Modified on	21/06/2022 13:56

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	Static analysis tools: To identify undesirable behavior of a program that might lead to vulnerabilities, developers use static analysis tools after a ML library update. Respondent S60 said, “ifthere is more static analysis in the form ofPython linters andMypy, the source code is more robust against error. So mypy usage just builds trust.” Tools like Jedi, Mypy [100]are static type checkers for Python that aim to combine the benefits ofdynamic typing and static typing. MyPy performs type checking on program that have type annotations (introduced in Python-v3.5 in 2015).
Modified on	21/06/2022 13:56

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	Supporting Multiple Library Versions. Respondent S19 said, “We are currently supporting 7 versions ofscikit-learn; 2 major versions, 0.20 and 0.21 and 5 minor versions.” About 15% of the respondents reveal that they support multiple versions of the ML libraries in their project. This is challenging, because developers have to implement workarounds for breaking changes and deprecation.
Modified on	21/06/2022 13:50

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	Table 9. Top Challenges of ML Library Version Updates
Modified on	21/06/2022 13:52

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	The ML libraries continuously and rapidly release updates that fix bugs, enhance features and introduce new functionalities to compete with other ML libraries. For example, Table 1 highlights that TensorFlow released almost 24 versions each year. The respondent S19 said, “Itiseasierto stay up to date with a more stable package like Matplotlib than it is with an evolving package like Scikit-Learn.” Another respondent S18 said, “ML libraries are dependent upon input data and pretrained models, which makes it difficult to update them.” Therefore, understanding the challenges for updating ML libraries that are dependent upon input data and pre-trainedmodels will highlight blind spots in research and tool support for Software-2.0. To gain a deeper insight into the associated challenges, we employ a quantitative and quali- tative method (developer survey). We use 81 survey responses from groups A and B (Table 3), to identify the impediments to update ML dependencies and understand the reasons behind it. We then analyzed the source code to highlight how widespread are the problems identified from the developer surveys. Table 9 summarizes the challenges that are identified from the developer surveys. In some survey responses, developers expressed multiple challenges (thus the percentages do not add to 100%). In the following subsections, we provide a detailed explanation for the challenges along with real examples from source code and quotes from the developers.
Modified on	21/06/2022 13:49

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	Tools SuggestedbyDevelopers. Table 13 summaries the new tool ideas revealed by survey respondents. In the following section, we provide a detailed explanation for the tool suggestion along with quotes from the developers. (1) Testing tools Model Coverage: Survey respondent S88 said, “Normal software update requires only to test source code, butML projects need a test for the dataset. A little bit ofchange in data will result in a crash. Therefore, knowing the testedML model and code coverage from the testing dataset would be helpful.” Depending on how representative the testing data is of the real-world data, the tested coverage of the model could be incomplete and the model may not be a true representation of real world application. Therefore, the concept of code coverage (i.e., a metric that describes the degree to which the source code of a program is executed when a particular test suite is executed) of Software-1.0 should be extended to model coverage in Software-2.0. Researchers [109, 132] suggest test dataset that gets full or high neurons activation ofa deep learning model can be considered as a good dataset for testing. Building end user tools that implement these techniques and integrating them to CI of Software-2.0 will help developers when deciding to ship the ML model to the production.
Modified on	21/06/2022 13:56

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	Tools Used by Developers. The survey results reveal that 48.33% of the developers do all the required changes manually while 43.33% of the developers use tool support to update ML libraries. We summarise the observed tools in Table 11. The rest of this subsection explains how ML developers use these tools in the ML library update process. (1) Version Control System (VCS): Survey respondents reveal that ML models are checked for fairness, performance, or accuracy after the update, prior to releasing it into production. Respondent S47 said, “ifwe are nothappywith theupdate, gitversion control helps to roll things back ifneeded.” As highlighted in Section 4.4.1, developers of Software-2.0 version data and ML models along with the source code. Moreover, a recent online article [53] discusses the limitation of the current version control system (e.g., Git) to assist in ML-specific problems like (i) storing and versioning large files and (ii) versioning data, models, and source code such that multiple experiments can co-exist. Tools like DVC7 and LFS8 try to solve this
Modified on	21/06/2022 13:56

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Software Mantainability Aspects
Created on	03/02/2022 15:26
Coded Text	we do not know what challenges developers encounter when they use ML libraries. With this knowledge gap, researchers miss opportunities to contribute to new research directions, tool builders do not invest resources where automation is most needed, library designers cannot make informed decisions when releasing ML library versions, and developers fail to use common practices when using ML libraries.
Modified on	21/06/2022 13:35

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Research gap in Maintaining ML systems
Created on	31/01/2022 14:42
Coded Text	RQ1: What is the trend in ML library usage? We found that between 2013 and 2018 the ratio of new Software-2.0 projects to other new projects in GitHub each year has increased from 1.75% to 49.63%. In the same period, the ratio ofnew ML library forks to all new forks added in GitHub each year has increased from 0.41% to 3.21%. RQ2: What combinations of libraries do developers use? Among many of our findings, we observed that 40.10% of the Software-2.0 projects use at least two ML libraries to implement various stages of the ML development workflow. RQ3: How do developers update ML library dependencies? Among others, we found that developers update versions of ML libraries more often than the other libraries used in the projects. ML library downgrades are more frequent compared to the other libraries in the projects, and 41.52% ofML library updates trigger cascading library updates. RQ4: What challenges arise when updating ML libraries? We surveyed developers about the challenges of evolving Software-2.0 and found seven common challenges, including binary incompatibility of trained ML models, re-selecting decision thresholds, benchmarking ML models, and supporting multiple versions. RQ5: What help do developers seek for updating ML libraries? Our survey showed that 26.66% of the ML library updates have taken more than a week to complete the update process. Further, developers mostly use the tools that are designed for Software-1.0 when updating ML library versions while preferring six different kinds of new tool support to evolve Software-2.0. RQ6: What challenges arise when retrofitting ML libraries to Software-1.0? We surveyed developers about why they retrofit ML libraries and what challenges they encountered. We found five reasons and eight challenges. Among others, we identified reasons such as augmenting functionalities with ML and replacing existing non-ML techniques with ML techniques. We identified challenges such as gathering data, managing data-pipeline, and adding ML to edge devices.
Modified on	31/01/2022 16:01

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56
Coded Text	(4) We make the collected data publicly available for further research and reuse [91].
Modified on	31/01/2022 16:03

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Snowballing
Parent Name	
Created on	31/01/2022 15:56
Coded Text	In contrast, there are very few studies about ML libraries used in Software-2.0. Islam et al. [69, 70] study StackOverflow posts and GitHub bug reports to identify ML-related bugs. Humbatova et al. [62] study GitHub issues and StackOverflow posts to create a taxonomy ofbugs for Software-2.0 while Han et al. [56] study the same subject systems to discover the trends of discussion topics in Software-2.0. However, none of these answer fundamental questions about usage, evolution, challenges, practices, and tool support for using ML libraries in Software-2.0
Modified on	31/01/2022 15:56

Name	Understanding Software-2.0~ A Study of Machine Learning Library Usage and Evolution
Number of Coding References	43
Number of Codes Coding	7
Coverage	9.26%
Folder Location	Codes\\Snowballing
Parent Name	Codes\\Snowballing\\Snowballing
Created on	21/06/2022 14:00
Coded Text	IMPLICATIONS Using our findings, we offer an empirically justified set of practical implications for researchers, tool builders, library vendors, hardware vendors, and software developers. 5.1 Researchers R1. Call for more research on Software-2.0 Libraries (RQ1):Our results reveal that the ratio of new Python projects that used a ML library increased from 2% in 2013 to 50% in 2018. This increasing trend highlights the prevalence of Software-2.0 in open source Python software systems. However, current research has focused on usage of libraries in Software-1.0: ACM Transactions on Software Engineering and Methodology, Vol. 30, No. 4, Article 55. Pub. date: July 2021. ! Understanding Software-2.0 55:29 evolution and maintenance [30, 38, 40, 75, 137], development practices [60], and analysing library usage specific to languages such as C# [106], Java [135]. Our dataset [91] contains a set of 3,340 Python projects that use a single or a combination of ML libraries and the 1,211 GitHub commits where these projects updated or adopted ML library(s). We hope that the research community would use our rich information as a starting point to investigate software evolution and development practices specific to Software-2.0. R2. NewTechniques to analyze and supportmulti-library projects (RQ2):We observed that 40.10% of the projects use multiple ML libraries at once. Moreover, we found that developers prefer to use multiple libraries to implement specific stages of the ML development workflow (e.g., Scikit-Learn for Data collection,and TensorFlow for Model training). Therefore, these multi-library environments pose several unique challenges for developers: the evolution of multi-library systems, partial and multiple library migrations, communication incompatibilities between libraries, inconsistent support for hardware accelerators, and so on. These challenges offer a plethora of problems that the research community can address. We provide examples of such multi-library environments in our dataset that contains 1,338 multi-library projects. R3. New techniques for solving ML model incompatibility (RQ4): When updating ML library version, 32 developers (of81) revealed that they retrained their ML models after performing a ML library update due to the binary incompatibility of serialized ML models that was trained on previous library versions. While Software-1.0 works based on source code of the program, Software-2.0 works based on source code of the program and trained ML models. This highlights new research opportunity for researchers to extend the previous research on source code incompatibilities [30, 38, 40, 120, 139] to (i) understand the ML model incompatibility or (ii) explore the interplay between source code and ML model incompatibilities. R4. Catalog breaking changes for Software-2.0 (RQ4 and RQ5): We found that that 52% of the respondents (of 81) adapt the source code when updating a ML library. Our results highlight Python-specific (e.g., keyword removing, keyword reordering, convert keyword parameter to positional parameter, etc.) or ML library specific breaking changes (e.g., change of computing architecture). Software-1.0 researchers [30, 38, 40] have focused on studying Java applications and created a catalog of the most common breaking changes. Further, Cossette et al. [30] categorize this catalog as fully, partially, or non automatable and Dietrich et al. [38] categorize the breaking changes as binary (and/or) source incompatible. Our dataset [91] contains the commits where a ML library was updated, which can be used to extend the catalog and explore breaking changes for Software-2.0. R5. Extend version control systems for Software-2.0 (RQ5): Our results shows that 32% ofrespondents use VCS (e.g., GitHub) in the library update process. Software-1.0 researchers have previously studied how the VCS affects the granularity of software changes [23, 77]. However, current VCS systems have limitations when versioning large data files or ML models. ML models are versioned in binary formats, so they are stored as large binary objects (which can waste a lot of disk space) and users cannot understand model changes. Researchers need to extend the current VCS tools to address unique challenges of Software-2.0, study how the proposed extensions impact the evolution of Software-2.0, and study the effectiveness of recently introduced ML specific VCS like DVC [128]. 5.2 Tool Builders

T1. Update ML libraries (RQ3): We observed several differences on how Software-2.0 developers use ML libraries: (i) they upgrade/downgrade ML libraries more often than ACM Transactions on Software Engineering and Methodology, Vol. 30, No. 4, Article 55. Pub. date: July 2021.
55:30 M. Dilhara et al.
traditional libraries, (ii) strict upgrades are the most popular among other update kinds (see Table 2), (iii) ML library upgrades/downgrades often result in cascading library updates (see Table 7), and (iv) developers often downgrade ML libraries (22.04% of updates). The current research [37, 85] and tooling [28, 47, 86] for library updates in statically typed languages work from one specific version to another. This highlights blind spots in the current tooling that does not account for the strict and non-strict nature of library updates in Python. Moreover, current techniques that support only upgrades should also support downgrades. To help advance the current tooling, we release 1,055 GitHub commits with ML library (cascading) updates, categorized by update kind. Tool builders [42, 46, 90, 97] can mine our dataset to learn from our mined exemplars.
T2. Tools for evolving trained ML models (RQ5): From our survey with 81 developers who performed a ML library update we found that developers need several tools specific for Software-2.0. (i) Tools for reporting model coverage (24.59% of respondents) compute coverage ofML model based on the testing data. (ii) Tools for benchmarking ofML models (18.03% of respondents) compute and compare metrics such as precision, recall, performance, energy consumption, model bias, with ML models trained on other library versions. (iii) Impact analysis tools (8.20% of respondents) examine the impact due to code and ML model changes.
T3. Tools for type-related changes (RQ6):We found that when developers retrofit ML libraries they start using optimized datatypes (e.g., numpy.array, pandas.DataFrame). Our survey also reveals that developers experiment with multiple libraries to find a good tradeoff between accuracy and performance. This highlights the need for tools that perform type-related changes (e.g., type migration or library migration). The previous tools [81, 133, 137] that perform typerelated changes heavily rely on the type-binding information provided by the compiler. Since in dynamically typed languages (like Python) type binding information is available only at runtime, the type-related change tools need to be extended with type inference techniques.
T4. Static analysis tools (RQ5): Our respondents said that static type checkers like MyPy and linters (that catch stylistic errors or suspicious constructs) build developer trust. However, MyPy requires type annotations to type check the program. To reduce the development effort ofmanually adding type annotations, Hellendoorn et al. [57] proposed a type annotation inference tool for JavaScript based on deep-learning. Extending this technique for Python and integrating it with MyPy will be useful for Software-2.0 developers. Moreover, tool builders can extend tools that identify and eliminate bugs or code smells (e.g., Error prone [2]or JDeodorant [136]) to identify and repair ML-specific bug patterns proposed by Islam et al. [69, 70].
5.3 Library Vendors L1. Manage increasing number of forks (RQ1): Our results highlight that there is an increasing number offorks ofML Library(s) each year: 0.41% of all newly created forks in 2013 in GitHub were for our studied ML Libraries, and this percentage grew to 3.21% in 2018. We also found that TensorFlow is the most forked Python repository on Github with over 78K forks. Similarly, other ML libraries like Keras, Caffe and Scikit-Learn have been forked around 18K times. Zhouetal. [146] observed that forking results in (i) community fragmentation and competition, (ii) lack of synchronization across the forks and the upstream, and (iii) disengagement of valuable contributors from the main project. This highlights a need for library vendors and tool builders to invest resources for mitigating the challenges faced by the ever growing number of forks.
L2. Improve API documentation (RQ6): Developers who lack expertise in mathematics or statistics find it hard to understand the documentation of ML libraries. This is confirmed by 21.43% of
ACM Transactions on Software Engineering and Methodology, Vol. 30, No. 4, Article 55. Pub. date: July 2021.
Understanding Software-2.0 55:31
respondents from our survey. To fill this gap, library developers should use novel techniques [3, 98] that simplify and summarize documentation ofML libraries.
L3. Add API support for more languages (RQ6): Our survey results show that the unavailability of ML library APIs across all major languages is a significant challenge. While TensorFlow supports a wide variety of languages (like C++, Python, Java, C, R, Go, and Swift), other ML libraries only support a subset ofPython, Java, and C++. The unavailability ofML library APIs across all major languages is an impediment for projects that are trying to retrofit ML. While providing support across all languages is not feasible, library vendors could invest resources to develop bindings for their libraries for multiple languages.
5.4 Hardware Vendors
H1. Optimise ML library combinations (RQ2): We observe that 40.10% of the projects use multi ML libraries at once. This highlights a blind spot for hardware manufacturers who are optimising their hardware for one specific library [7, 10, 63, 66](e.g., Intel is optimising its CPUs for TensorFlow and Caffe, Apple is optimising TensorFlow, Keras, and Caffe). Our dataset contains popular projects that use multi-libraries. We reveal the most frequently used combinations and also identify the features that developers use in various stages oftheMLworkflow (Table 6). When designing hardware accelerators for ML tasks, hardware vendors can use our dataset to gain insights into representative patterns ofcomputations in multi-library environments. This helps them prioritize what to optimize on their hardware.
H2. Optimise edge devices (RQ6): We observe that 18% of respondents retrofit ML in applications for edge devices such as mobile and smartwatches. This

	is challenging, because edge devices have limited processing power and energy for running ML algorithms. Our respondents call for hardware optimizations that allow processing on edge devices while maintaining the accuracy of the output. 5.5 Software Developers and Educators S1. Rich educational resources (RQ2 and RQ3): Developers learn and educators teach new programming constructs through examples. Robillard et al. [117] studied the challenges of learning APIs and concluded that one of the important factors is the lack of usage examples. Using our dataset of 809,534 total number ofML constructs that we mined in our corpus, developers and educators can learn from real-world examples (e.g., selecting proper values for hyperparameters in ML library APIs is not easy [17]. Developers can use our dataset to learn the values from other codes). Moreover, we also release our dataset with 1,055 commits that contain ML library API updates. S2. Awareness about ML-specific development tools (RQ5): We observed that developers use VCS and CI systems that are specifically designed for Software-1.0 development processes. Even though there exists tools like DVC [128], Pachyderm [64], and MLFlow [34] that are specifically designed for versioning and integrating Software-2.0, these tools are still not popular among developers. We encourage developers to use these novel tools for Software-2.0.
Modified on	21/06/2022 14:00
Name	WALTS~ Walmart AutoML Libraries, Tools and Services
Number of Coding References	5
Number of Codes Coding	2
Coverage	13.43%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	A pertinent question to ask is why do we need to build an AutoML service within Walmart when there are similar commercial services provided by Google [3], H2O [4], Dataiku [5] and many more. We internally evaluated some of these services and found some gaps as mentioned below, which we address with our own AutoML capability. Note that details on WALTS will be covered subsequently in Section III. • Customizability: Currently, all off-the-shelf AutoML services provide a fixed set of metrics (e.g., Google Cloud AutoML supports only accuracy, precision, recall, Area Under Curve (AUC) and F1 score) with no provision to extend this set by the users. Some of these, at times, can even be misleading, e.g., accuracy when the classes are imbalanced. After talking to internal clients' requirements, we may easily introduce new metrics, such as, balanced accuracy which is better suited for imbalanced classes. • Transparency: Typically, the AutoML frameworks do not report which other models these have tried out before choosing the winners. For WALTS, the details of all the explored models are readily available in the documentation page. Now consider this scenario: what if there are some hard latency requirements? In such cases,
Modified on	16/02/2023 12:50

Name	WALTS~ Walmart AutoML Libraries, Tools and Services
Number of Coding References	5
Number of Codes Coding	2
Coverage	13.43%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	Architectural Overview From an architectural perspective, our solution is developed keeping three primary aspects in mind: • Reliability: In most scenarios, the training jobs will be long running jobs and there will be multiple jobs running concurrently in the system; therefore, the system should be able to handle the high load and should be capable of handling various failure scenarios. • Efficiency: The system should be efficient both in terms of the time taken to complete the training job and the number of resources (compute cost) used during the training. Thus, the various training algorithms (corresponding to various experiments) should be executed in parallel and the system should be capable of distributing the jobs. Based upon resource availability, the users should have the control to run the jobs with better compute power (CPUs and/or GPUs) to decrease the overall training completion time. • Scalability: The system should be able to handle the increased job load at Walmart scale. Within Walmart, we have an in-house ML platform called element [28], that has been designed keeping the above mentioned aspects in mind. It has already been used to productionize solutions for various business use-cases in Walmart such as, aisle assortment, wait-time at stores and personalization. Fig. 1 shows architectural overview of WALTS framework that has been built on top of the element platform. As shown in this figure, the training jobs are triggered via the user interface of WALTS Services that takes the usual parameters – a ML task to accomplish, a dataset and a metric, as inputs. Additionally, we supply a predetermined configuration (which may be customized by the expert users) that includes the maximum number of epochs to train for, the compute power (i.e., the number of CPUs, GPUs), output file location, etc. To achieve reliability, efficiency and scalability, we are using Kubernetes [29] and Airflow [30] configured with Kubernetes executor. Every experiment in a training job runs as a containerised application on Kubernetes pod as an Airflow worker. The training requests with parameters (e.g., task, models to explore) are converted into an Airflow pipeline (also referred to as Directed Acyclic Graph or DAG) where the nodes correspond to the experiments. Each of these nodes are configured with a custom Git operator which sets up the container by checking out the predefined training code with AutoML algorithms from the Git and installs the libraries required to execute the code. During execution, it runs a number of algorithms in parallel depending on the availability of resources. The execution of each experiment happens inside a pod that follows a predefined configuration. These Kubernetes pods get created on the fly and this gives us the capability to handle burstable workloads. As the system is multi-tenant and there may be jobs running concurrently, there can be cases where a single job/tenant may try to over-consume the resources. To avoid such scenarios, the system needs to put an upper threshold on the consumption of resources by a tenant. The resource limit per tenant is controlled using the namespace level ResourceQuota in Kubernetes. In case of resource unavailability for a training job for a tenant, the jobs will be queued and will start execution as soon as the resources get added for the tenant or already running experiments complete their execution and release the resources. The Kubernetes infrastructure is shared by all the tenants, and we rely on the Kubernetes cluster auto-scaler for scalability. Sometimes the system experiences a greater load, in such cases, Kubernetes can automatically increase the cluster nodes and deploy pods as necessary to handle the increased demand. When the load decreases, Kubernetes can then reduce the nodes and pods dynamically. Once an experiment is done, all the performance logs are saved in Azure SQL database which can be easily retrieved through queries submitted by the users – this is typically done by them to compare the metrics of various experiments; other artifacts such as models are stored to GCS buckets in Google Cloud Platform. The storages are managed by MLflow [27] that also acts as a bridge for the user to retrieve the desired logs and artifacts.
Modified on	16/02/2023 12:51

Name	WALTS~ Walmart AutoML Libraries, Tools and Services
Number of Coding References	5
Number of Codes Coding	2
Coverage	13.43%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	Automated Machine Learning (AutoML) is an up- coming field in machine learning (ML) that searches the candidate model space for a given task, dataset and an evaluation metric and returns the best performing model on the supplied dataset as per the given metric. AutoML not only reduces the man-power and expertise needed to develop ML models but also decreases the time-to-market for ML models substantially. In Walmart, we have designed an enterprise-scale AutoML framework called WALTS to meet the rising demand of employing ML in the retail business, and thus help democratize ML within our organization. In this work, we delve into the design of WALTS from both algorithmic and architectural perspectives. Specifically, we elaborate on how we explore models from a pool of candidates along with describing our choice of technology stack to make the whole process scalable and robust. We illustrate the process with the help of a business use-case, and finally underline how WALTS has impacted our business so far.
Modified on	16/02/2023 12:49

Name	WALTS~ Walmart AutoML Libraries, Tools and Services
Number of Coding References	5
Number of Codes Coding	2
Coverage	13.43%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications
Created on	03/02/2022 14:38
Coded Text	the client may be happy with the second best model (in terms of accuracy, say) if it fits her latency needs — this kind of support can only be given by WALTS. • Unique Enhancements: We plan to extend WALTS with unique enhancements which other AutoML frameworks do not provide, such as, noise-aware training [6], intelligent data imputation [7] (interestingly, we found that other AutoML tools usually prefer dropping rows with missing data instead of imputing these), etc. • Low Precision: WALTS supports half-precision floating point (FP16) datatype for deep learning models which can be exploited while running on specialized hardware such as, Nvidia GPUs [8]. To the best of our knowledge, all AutoML frameworks provide only full-precision (FP32) models, which need to be separately converted into low precision models. • Data Sensitivity: Since WALTS is an in-house tool, Walmart clients feel more safe with their data – running WALTS at on-prem devices excludes the risk of exposing sensitive data to external cloud service providers (CSPs). • Reduced time-to-explore: While commercial AutoML's complex search algorithms may be beneficial to find adhoc architectures (by trying out a combinatorial number of different neural network layers), WALTS's simple policy of exploring only a pre-defined set of models may be less time consuming. Moreover, the models explored by WALTS being already popular, the clients may be more open to adopting these. • Cost Efficient: The costs are less with WALTS; moreover, we do not have the concept of minimum charges which is imposed by external vendors. The paper is organized as follows. Section II covers some of the related work in AutoML literature. Section III gives an overview of WALTS both from algorithmic and architectural perspectives. We further provide details on the user interfaces for WALTS in this section. We illustrate the benefit of WALTS with the help of a business use-case in Section IV. The experimental results are given in Section V. Section VI concludes the paper.
Modified on	16/02/2023 12:50

Name	WALTS~ Walmart AutoML Libraries, Tools and Services
Number of Coding References	5
Number of Codes Coding	2
Coverage	13.43%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications
Created on	07/02/2022 14:10
Coded Text	Auto-WEKA [9] that tried to simultaneously search for the optimal learning algorithm along with best performing hyperparameter values is considered as the first AutoML tool. Its scope, however, was limited to only classification task, and the models and the datasets explored were rather small by today's comparison. ATM [10] is a more recent distributed and scalable AutoML framework. However, ATM also focuses on only classification task – the authors test their framework on datasets that have a maximum of 4 classes only; moreover, it is unable to handle missing data. Katib [11] is another recent platform that targets distributed AutoML on Kubernetes. A shortcoming of Katib is that it requires hints from the users to explore the model search space, which can be a major roadblock for non-experts. It may be important to note that Ultron [12] is another AutoML framework designed by a different Walmart team. Its objective is limited to hyperparameter optimization only and does not include model search for various tasks as we do. In addition, there are commercial tools such as, Google Cloud AutoML [3], H2O AutoML [4] and Dataiku AutoML [5]. The major differences with all these tools which we have explored have already been covered in the previous section
Modified on	16/02/2023 12:51

Name	WALTS~ Walmart AutoML Libraries, Tools and Services Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	What Are We Really Testing in Mutation Testing for Machine Learning~ A Critical Reflection
Number of Coding References	6
Number of Codes Coding	1
Coverage	4.18%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Clearly settle on what the system under test (SUT) is when extending software testing techniques to ML systems. Currently, mutation testing for DL is a common umbrella for all techniques that target DL models, the training procedure, and the surrounding software artifacts. However, as we argued in Section II, ML model development (both within and outside of DL) involves several development stages with associated artifacts, each requiring dedicated, different testing strategies. Furthermore, where in traditional ML, one test set suffices for a specific ML task (regardless of what ML model is chosen for this task), the mutation testing paradigm implies that the adequacy of a given test suite may differ, depending on the program it is associated with. Thus, depending on the program, a given test suite may need different amplifications.
Modified on	31/01/2022 15:36

Name	What Are We Really Testing in Mutation Testing for Machine Learning~ A Critical Reflection
Number of Coding References	6
Number of Codes Coding	1
Coverage	4.18%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Finally, we wish to point out that the evolution of ML programs differs from ‘traditional’ software. In traditional software, revisions are performed iteratively, applying small improvements to previous production code. This is uncommon in ML, where revisions are typically considered at the task level (e.g. ‘find an improved object recognition model’), rather than at the level of production code (e.g. ‘see whether changing neuron connections in a previously-trained deep neural network can improve performance’)
Modified on	31/01/2022 15:37

Name	What Are We Really Testing in Mutation Testing for Machine Learning~ A Critical Reflection
Number of Coding References	6
Number of Codes Coding	1
Coverage	4.18%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Investigate what 'a small mutant' is. This aspect is rather complex in ML: repeating the exact same training process on the exact same data is not mutation, but may yield major model differences, due to stochasticity in training. Similarly, a small back-propagation step may affect many weights in a DL model. At the same time, given the high dimensionality of data and high amounts of parameters in DL models, other mutations (e.g. noise additions to pixels) may be unnoticeable as first-order mutants, and may need to be applied in many places at once to have any effect.
Modified on	31/01/2022 15:36

Name	What Are We Really Testing in Mutation Testing for Machine Learning~ A Critical Reflection
Number of Coding References	6
Number of Codes Coding	1
Coverage	4.18%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	two mutation testing proposals were proposed in parallel: MuNN by Shen et al. [16], and DeepMutation by Ma et al. [17], both seeking to propose mutation operators tailored to deep models. Both methods focus on convolutional neural network (CNN) models for 'classical' image recognition tasks (MNIST digit recognition in MuNN and DeepMutation, plus CIFAR-10 object recognition in DeepMutation).
Modified on	31/01/2022 15:24

Name	What Are We Really Testing in Mutation Testing for Machine Learning~ A Critical Reflection
Number of Coding References	6
Number of Codes Coding	1
Coverage	4.18%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	When defining new mutation operators for ML, be explicit about what production and test code are.
Modified on	31/01/2022 15:35

Name	What Are We Really Testing in Mutation Testing for Machine Learning~ A Critical Reflection
Number of Coding References	6
Number of Codes Coding	1
Coverage	4.18%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering
Created on	31/01/2022 15:24
Coded Text	Work with experts to better determine what ‘realistic mu- tants’, ‘realistic faults’ and ‘realistic tests’ are. Ma et al. [17] rightfully argued that not all faults in DL systems are created by humans, but little insight exists yet on how true, realistic faults can be mimicked.
Modified on	31/01/2022 15:36

Name	Why is Developing Machine Learning Applications Challenging~ A Study on Stack Overflow Posts
Number of Coding References	5
Number of Codes Coding	3
Coverage	1.07%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	31/01/2022 14:44
Coded Text	(3) the data preprocessing and model deployment phases are where most of the challenges lay; and (4) addressing most of these challenges require more ML implementation knowledge than ML conceptual knowledge.
Modified on	31/01/2022 14:44

Name	Why is Developing Machine Learning Applications Challenging~ A Study on Stack Overflow Posts
Number of Coding References	5
Number of Codes Coding	3
Coverage	1.07%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	31/01/2022 14:44
Coded Text	Data Pre-processing and Manipulation (DP) We assume the developer is preparing his data for a ML model(s). Questions about data loading, data accessing, data cleaning, data splitting, data format changing, data labelling, data imbalance issues, data normalization, etc.
Modified on	23/02/2022 20:02

Name	Why is Developing Machine Learning Applications Challenging~ A Study on Stack Overflow Posts
Number of Coding References	5
Number of Codes Coding	3
Coverage	1.07%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering
Created on	31/01/2022 14:44
Coded Text	The most challenging ML topics show difficulty with data and feature preprocessing, environment setup, and model deployment.
Modified on	23/02/2022 20:03

Name	Why is Developing Machine Learning Applications Challenging~ A Study on Stack Overflow Posts
Number of Coding References	5
Number of Codes Coding	3
Coverage	1.07%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	06/01/2022 15:15
Coded Text	(3) the data preprocessing and model deployment phases are where most of the challenges lay; and (4) addressing most of these challenges require more ML implementation knowledge than ML conceptual knowledge.
Modified on	31/01/2022 14:43

Name	Why is Developing Machine Learning Applications Challenging~ A Study on Stack Overflow Posts
Number of Coding References	5
Number of Codes Coding	3
Coverage	1.07%
Folder Location	Codes\\Maintainable ML
Parent Name	Codes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering
Created on	03/02/2022 10:24
Coded Text	Model Fitting (MF) We assume the developer has a specific model in mind (e.g., SVM), so questions related to a specific model implementation, training, convergence determination, etc. Model Tuning (MT) We assume the developer has trained a specific model and is aiming to fine tune it through hyper-parameter tuning, learning rate, regularization, etc. Model Evaluation and Result Interpretation (ME) Model Deployment and Environment Setup (MD) Others We assume the developer completed the training and tuning of a single or multiple ML models. Questions related to evaluation or measuring the performance of a model. Questions related to results interpretation Questions related to environment setup, memory or storage issues, deployment performance tuning, etc
Modified on	23/02/2022 20:03

Name	Workflow Improvement for KubeFlow DL Performance over Cloud-Native SmartX AI Cluster
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	

Name	ZeRO~ Memory optimizations Toward Training Trillion Parameter Models
Number of Coding References	10
Number of Codes Coding	2
Coverage	12.64%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\Challenges in scaling a ML system or application
Created on	07/02/2022 19:54
Coded Text	Deep Learning (DL) models are becoming larger, and the increase in model size offers significant accuracy gain. In the area of Natural Language Processing (NLP), Transformers [1] have paved way for large models like Bert-large (0.3B) [2], GPT-2 (1.5B) [3], Megatron-LM (8.3B) [4], T5 (11B) [5]. To enable the continuation of model size growth from 10s of billions to trillions of parameters, we experience the challenges of training them - they clearly do not fit within the memory of a single device, e.g., GPU or TPU, and simply adding more devices will not help scale the training. Basic data parallelism (DP) does not reduce memory per de- vice, and runs out of memory for models with more than 1.4B parameters on GPUs with 32GB memory when trained using *Equal Contribution SC20, November 9-19, 2020, Is Everywhere We Are 978-1-7281-9998-6/20/$31.00 c~2020 IEEE common settings like mixed precision and ADAM optimizer [6]. Other existing solutions such as Pipeline Parallelism (PP), Model Parallelism (MP), CPU-Offloading, etc, make tradeoffs between functionality, usability, as well as memory and compute/communication efficiency, all of which are crucial to training with speed and scale. Among different existing solution for training large models, MP is perhaps the most promising one. The largest models in the current literature, the 11B T5 model [5], and MegatronLM 8.3B [4], were both powered by model parallelism, implemented in Mesh-Tensorflow [7] and Megatron-LM[4], respectively. However, MP cannot scale much further beyond these models sizes. MP splits the model vertically, partitioning the computation and parameters in each layer across multiple devices, requiring significant communication between each layer. As a result, they work well within a single node where the inter-GPU communication bandwidth is high, but the efficiency degrades quickly beyond a single node [4]. We tested a 40B parameter model using Megatron-LM across two DGX-2 nodes and observe about 5Tflops per V100 GPU (less than 5% of hardware peak) compared to near 30Tflops for models that can be trained within a node.
Modified on	28/02/2023 13:22

Name	ZeRO~ Memory optimizations Toward Training Trillion Parameter Models
Number of Coding References	10
Number of Codes Coding	2
Coverage	12.64%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	II. RELATED WORK A. Data, Model and Pipeline Parallelism Parallelization is a key strategy on training large models at scale. For a model that fits in the device memory for training, data parallelism (DP) is used to scale training to multiple devices. In DP, model parameters are replicated on each device. At each step, a mini-batch is divided evenly across all the data parallel processes, such that each process executes the forward and backward propagation on a different subset of data samples, and uses averaged gradients across processes to update the model locally. When a model does not fit in the device memory, model parallelism (MP) [7], [4], [11] and pipeline parallelism (PP) [12], [13] split the model among processes, in vertical and horizontal way respectively. Sec. I discussed how ZeRO relates to DP and MP. We now discuss PP and how it relates to reducing memory consumption. PP splits a model horizontally across layers running each partition on a different device and use micro-batching to hide the pipeline bubble [12], [13]. Model functionalities such as tied-weights and batch-normalization are difficult to implement due to horizontal splitting and micro-batching, respectively. Popular PP implementation such as G-pipe [12] partitions both model parameters and total activations but requires a batch size proportional to number of pipeline partitions to hide the pipeline bubble. The large batch size can affect the convergence rate, while also requiring significant memory to store activations. A different implementation of PP in PipeDream [14] keeps multiple copies of stale parameters
Modified on	28/02/2023 13:22

Name	ZeRO~ Memory optimizations Toward Training Trillion Parameter Models
Number of Coding References	10
Number of Codes Coding	2
Coverage	12.64%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Large deep learning models offer significant accuracy gains, but training billions to trillions of parameters is challenging. Existing solutions such as data and model parallelisms exhibit fundamental limitations to fit these models into limited device memory, while obtaining computation, communication and development efficiency. We develop a novel solution, Zero Redundancy Optimizer (ZeRO), to optimize memory, vastly improving training speed while increasing the model size that can be efficiently trained. ZeRO eliminates memory redundancies in data- and model-parallel training while retaining low communication volume and high computational granularity, allowing us to scale the model size proportional to the number of devices with sustained high efficiency. Our analysis on memory requirements and communication volume demonstrates: ZeRO has the potential to scale beyond 1 Trillion parameters using today's hardware. We implement and evaluate ZeRO: it trains large models of over 100B parameter with super-linear speedup on 400 GPUs, achieving throughput of 15 Petaflops. This represents an 8x increase in model size and 10x increase in achievable performance over state-of-the-art. In terms of usability, ZeRO can train large models of up to 13B parameters (e.g., larger than Megatron GPT 8.3B and T5 11B) without requiring model parallelism which is harder for scientists to apply. Last but not the least, researchers have used the system breakthroughs of ZeRO to create Turing-NLG, the world's largest language model at the time (17B parameters) with record breaking accuracy.
Modified on	28/02/2023 13:21

Name	ZeRO~ Memory optimizations Toward Training Trillion Parameter Models
Number of Coding References	10
Number of Codes Coding	2
Coverage	12.64%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	So, how can we overcome the limitations of existing so- lutions and train large models more efficiently? To answer this question, we first analyze the full spectrum of memory consumption of the existing systems on model training and classify it into two parts: 1) For large models, the majority of the memory is occupied by model states which include the optimizer states (such as momentum and variances in Adam [6]), gradients, and parameters. 2) The remaining memory is consumed by activation, temporary buffers and unusable fragmented memory, which we refer to collectively as residual states. We develop ZeRO— Zero Redundancy Optimizer — to optimize memory efficiency on both while obtaining high compute and communication efficiency. As these two parts face different challenges, we develop and discuss their solutions correspondingly. Optimizing Model State Memory Model states often consume the largest amount of memory during training, but existing approaches such as DP and MP do not offer a satisfying solution. DP has good compute/communication efficiency but poor memory efficiency while MP can have poor compute/communication efficiency. More specifically, DP replicates the entire model states across all data parallel process resulting in redundant memory consumption; while MP partition these states to obtain high memory efficiency,
Modified on	28/02/2023 13:22

Name	ZeRO~ Memory optimizations Toward Training Trillion Parameter Models
Number of Coding References	10
Number of Codes Coding	2
Coverage	12.64%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	Super-Linear Scalability ZeRO-100B demonstrates super-linear scalability for very large model sizes. Figure 3 shows scalability results for a 60B parameter model going from 64 to 400 GPUs and we expect this trend to continue further for more GPUs. Pos+g reduces per GPU memory consumption of ZeRO-100B with increase in DP degree, allowing ZeRO-100B to fit larger batch sizes per GPU, which in turn improves throughput as a result of increasing arithmetic intensity. Please note that the throughput scalability shown here do not represent strong or weak scaling in the traditional sense, however, it is a proxy for the end-to-end training time, which is more meaningful in the context of DL training. In DL training, the amount of computation per iteration changes with different batch sizes, while the total end-to-end computation for a fixed total training samples is the same. Therefore, despite the change in amount of computation per iteration from an increased batch size, the end-to-end training time is inversely proportional to the absolute throughput number shown in Figure 3. It is also worth pointing out that a significant batch size increase without increasing the total training samples can lead to poor convergence. However, this is not the case for our results. The largest batch size used in this experiment is 1.6K, which is well within range of batch sizes used to train LM models (512-2K) [3], [31]. In fact, a growing body of literature show that convergence rate for LM is resilient to batch sizes that are much larger than the ones used in this experiments [32], [25]. D. Democratizing Large Model Training Using MP and PP is challenging for many data scientists, which is a well-known hurdle to train large models. ZeRO does not require any changes to the model itself and it can be used as simple as baseline DP while delivering significantly boosted model size and speed. Fig. 8 shows that ZeRO-100B can train models with up to 13B parameters without MP on 128 GPUs, achieving throughput over 40 TFlops per GPU on average. In comparison, without ZeRO, the largest trainable model with DP alone has 1.4B parameters with throughput less than 20 TFlops per GPU. Furthermore, in the absence of the communication overhead from MP, these models can be trained with lower-end compute nodes without very fast intranode interconnect such as NVLINK or NVSwitch, which is required to achieve good efficiency with MP. E. Memory and Performance Analysis We look into the benefits and impact of different optimizations on maximum model size, memory consumption and performance. These optimizations are referred to as Config 1 to 5 (C1-C5) in Table. III. a) Maximum Model Size: Figure 5 shows the largest trainable model by enabling different ZeRO optimizations for a fixed batch size and MP of 16. The model size increase from 40B to 60B when trained with C1 vs C2 due to a 16x (MP degree) reduction in activation memory from using Pa, while the jump to 140B using C4 is from enabling Pos+g which halves the memory requirement by the model states compared to Pos in C2. The increase to 150B using C5 is solely due to further reduction in activation memory from offloading the partitioned activation checkpoints to the CPU memory. b) Max Cached Memory: Figure 6 shows the maximum memory cached by PyTorch during each training iteration for a 40B and a 100B parameter model. The decrease of the cached memory size is as expected from C1 to C2. The difference in memory consumption between C2 and C3 depends on the size of the model states in comparison to the activation memory, and can increase when activation memory is larger, or decrease when the model states are larger. It is note worthy that the cached memory does not decrease from C4 to C5 for 40B but it does for 100B. This is simply because the activation memory for 100B is much larger for the decrease to be noticeable. This makes Pa+cpu a valuable tool to fit a larger batch size when we get to very large models. In Figure 7, Pa+cpu is needed for 170B model to execute without running out of memory. c) Max Achievable Performance: Figure 7 shows the

	best achievable performance for different set of optimizations. Notice that performance improvement corresponds to decrease in memory consumption between the optimizations. As mentioned earlier, lower memory consumption allows for larger batch size which improves performance. The only caveat is the performance drop between C4 and C5 for 60B parameter model. Despite lower memory consumption, C5 incurs activation movement to and from the CPU, this will result in worse performance in most cases, except for a few where the model is so large that the model simply cannot run without C5 or the batch size that can run without C5 is very small (such as model with 170B parameters in Figure 7). During training, Pa+cpu is turned on only when it is beneficial. F. Turing-NLG, SOTA language model with 17B parameters As of April 22, 2020, Turing-NLG [10] was the largest model in the world with over 17B parameters. It achieved the new SOTA for language models with Webtext-103 perplexity of 10.21. Turing-NLG was trained end-to-end using ZeRO100B and Fig. 9 shows the validation perplexity over 300K iterations compared to previous SOTA, Megatron-LM 8.3B parameter model. ZeRO-100B achieves a sustained throughput of 41.4 TFlops/GPU for this model.
Modified on	28/02/2023 13:23
Name	ZeRO~ Memory optimizations Toward Training Trillion Parameter Models
Number of Coding References	10
Number of Codes Coding	2
Coverage	12.64%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	the scalability across GPUs, especially when crossing node boundaries. On the contrary, DP has both higher computational granularity and lower communication volume, allowing for much higher efficiency. b) DP is memory inefficient as model states are stored redundantly across all data-parallel processes. On the contrary, MP partitions the model states to obtain memory efficiency. c) Both DP and MP keep all the model states needed over the entire training process, but not everything is required all the time. For example, parameters corresponding to each layer is only needed during the forward propagation and backward propagation of the layer. Based on these insights, ZeRO-DP retains the training efficiency of DP while achieving the memory efficiency of MP. ZeRO-DP partitions the model states instead of replicating them (Section V) and uses a dynamic communication schedule that exploits the intrinsically temporal nature of the model states while minimizing the communication volume (Section VII). By doing so, ZeRO-DP reduces per-device memory footprint of a model linearly with the increased DP degree while maintaining the communication volume close to that of the default DP, retaining the efficiency. B. Insights and Overview: ZeRO-R 1) Reducing Activation Memory: Two key insights are: a) MP partitions the model states but often requires repli- cation of the activation memory. For example, if we split the parameters of a linear layer vertically and compute them in parallel across two GPUs, each GPU requires the entire activation to compute its partition b) For models such as GPT-2 or larger, the arithmetic intensity (ratio of the amount of computation per iteration to amount of activation checkpoints per iteration) is very large (≥ 10K) and increases linearly with hidden dimension making it possible to hide the data-movement cost for the activation checkpoints, even when the bandwidth is low. ZeRO removes the memory redundancies in MP by parti- tioning the activations checkpoints across GPUs, and uses allgather to reconstruct them on demand. The activation memory footprint is reduced proportional to the MP degree. For very large models, ZeRO can even choose to move the activation partitions to the CPU memory, while still achieving good efficiency due to large arithmetic intensity in these models. 2) Managing Temporary buffers: ZeRO-R uses constant size buffers to avoid temporary buffers from blowing up as the model size increases, while making them large enough to remain efficient. 3) Managing fragmented Memory: Memory fragmentation is a result of interleaving between short lived and long lived memory objects. During the forward propagation activation checkpoints are long lived but the activations that recomputed are short lived. Similarly, the backward computation, the activation gradients are short lived while the parameter gradients are long lived. Based on this insight, ZeRO performs on-thefly memory defragmentation by moving activation checkpoints and gradients to pre-allocated contiguous memory buffers.
Modified on	28/02/2023 13:23

Name	ZeRO~ Memory optimizations Toward Training Trillion Parameter Models
Number of Coding References	10
Number of Codes Coding	2
Coverage	12.64%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	to hide the pipeline bubble without increasing the batch size significantly, making it less memory efficient. Additionally, the implementation is not equivalent to the standard DL training and has implications on training convergence. In contrast, ZeRO obtains the same or better memory efficiency than PP without incurring functionality, performance and convergence related restrictions of PP.
Modified on	28/02/2023 13:22

Name	ZeRO~ Memory optimizations Toward Training Trillion Parameter Models
Number of Coding References	10
Number of Codes Coding	2
Coverage	12.64%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	XI. CONCLUDING REMARKS [10] Microsoft. Turing-nlg: A 17-billion-parameter language model by microsoft. From a HPC and system perspective, we believe that ZeRO represents a revolutionary transformation in the large model training landscape. While our implementation, ZeRO-100B, enables 8x increase in model sizes, over 10x in throughput improvement, achieves super-linear speedups on modern GPU clusters, and trains the largest model in the world, it is still just a tip of the iceberg. ZeRO in its entirety has the potential to increase the model size by yet another order of magnitude, enabling the training of trillion parameter models of the future. Perhaps, what we feel most optimistic about ZeRO is that it imposes no hurdles on the data scientists. Unlike existing approaches such as MP and PP, no model refactoring is necessary, and it is as easy to use as standard DP, making ZeRO a prime candidate for future investigations on large model training. Through open sourcing and community feedback, we plan to make ZeRO fully accessible to the DL community to catalyze the evolution and democratization of large model training at scale.
Modified on	28/02/2023 13:23

Name	ZeRO~ Memory optimizations Toward Training Trillion Parameter Models
Number of Coding References	10
Number of Codes Coding	2
Coverage	12.64%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	ZeRO:INSIGHTS AND OVERVIEW ZeRO has two sets of optimizations: i) ZeRO-DP aimed at reducing the memory footprint of the model states, and ii) ZeRO-R targeted towards reducing the residual memory consumption. We present an overview of the optimizations and the insights behind, which allows ZeRO to reduce memory footprint while remaining efficient. Please note efficiency is a key here: without this constraint, trivial solutions like moving all the parameter states to the CPU memory, or increasing the MP degree arbitrarily can reduce memory footprint. A. Insights and Overview: ZeRO-DP ZeRO powered DP is based on three key insights: a) DP has better scaling efficiency than MP because MP reduces the granularity of the computation while also increasing the communication overhead. Beyond a certain point, lower computational granularity reduces the efficiency per GPU, while the increased communication overhead, hiders
Modified on	28/02/2023 13:23

Name	ZeRO~ Memory optimizations Toward Training Trillion Parameter Models
Number of Coding References	10
Number of Codes Coding	2
Coverage	12.64%
Folder Location	Codes\\Scalable ML
Parent Name	Codes\\Scalable ML\\what are the available solution to scale ML systems
Created on	07/02/2022 14:03
Coded Text	ZeRO-DP, ZeRO-powered data parallelism, that achieves the computation/communication efficiency of DP while achieving memory efficiency of MP. ZeRO-DP removes the memory state redundancies across dataparallel processes by partitioning the model states instead of replicating them, and it retains the compute/communication efficiency by retaining the computational granularity and communication volume of DP using a dynamic communication schedule during training. ZeRO-DP has three main optimization stages (as depicted in Figure 1), which correspond to the partitioning of optimizer states, gradients, and parameters (Sec. V). When enabled cumulatively: 1) Optimizer State Partitioning (Pos): 4x memory reduction, same communication volume as DP; 2) Add Gradient Partitioning (Pos+g): 8x memory reduction, same communication volume as DP; 3) Add Parameter Partitioning (Pos+g+p): Memory reduc- tion is linear with DP degree Nd. For example, splitting across 64 GPUs (Nd = 64) will yield a 64x memory reduction. There is a modest 50% increase in communication volume. ZeRO-DP eliminates memory redundancies and makes the full aggregate memory capacity of a cluster available. With all three stages enabled, ZeRO can train a trillion-parameter model on just 1024 NVIDIA GPUs. A trillion-parameter model with an optimizer like Adam [6] in 16-bit precision requires approximately 16 terabytes (TB) of memory to hold the optimizer states, gradients, and parameters. 16TB divided by 1024 is 16GB, which is well within a reasonable bound for a GPU (e.g., with 32GB of on-device memory). Optimizing Residual State Memory After ZeRO-DP boosts memory efficiency for model states, the rest of the
Modified on	28/02/2023 13:22

Name	ZeRO~ Memory optimizations Toward Training Trillion Parameter Models Imported Notes
Number of Coding References	0
Number of Codes Coding	0
Coverage	0.00%
Folder Location	
Parent Name	
Created on	
Coded Text	
Modified on	