Preprint. ICSE 2023.

Lightweight Approaches to DNN Regression Error Reduction: An Uncertainty Alignment Perspective

Zenan Li[1], Maorun Zhang[1], Jingwei Xu[1], Yuan Yao[1], Chun Cao[1], Taolue Chen[2], Xiaoxing Ma[1], Jian Lü[1]
[1]State Key Lab of Novel Software Technology and
Department of Computer Science and Technology, Nanjing University, China
[2]Department of Computer Science, Birkbeck, University of London, UK
{lizenan, maorunzhang}@smail.nju.edu.cn, {jingweix, y.yao, caochun}@nju.edu.cn,
t.chen@bbk.ac.uk, {xxm, lj}@nju.edu.cn

***Abstract*—Regression errors of Deep Neural Network (DNN) models refer to the case that predictions were correct by the old-version model but wrong by the new-version model. They frequently occur when upgrading DNN models in production systems, causing disproportionate user experience degradation. In this paper, we propose a lightweight regression error reduction approach with two goals: 1) requiring no model retraining and even data, and 2) not sacrificing the accuracy. The proposed approach is built upon the key insight rooted in the unmanaged model uncertainty, which is intrinsic to DNN models, but has not been thoroughly explored especially in the context of quality assurance of DNN models. Specifically, we propose a simple yet effective ensemble strategy that estimates and aligns the two models' uncertainty. We show that a Pareto improvement that reduces the regression errors without compromising the overall accuracy can be guaranteed in theory and largely achieved in practice. Comprehensive experiments with various representative models and datasets confirm that our approaches significantly outperform the state-of-the-art alternatives.**

***Index Terms*—Software regression, deep neural networks, uncertainty alignment, model ensemble**

I. Introduction

As key components of many modern software systems, deep learning models undergo continuous evolution (due to, e.g., the availability of new data or new model architectures) [1]. Similar to the case in traditional software engineering [2]–[5], the continuous evolution of machine learning models may also introduce *regression errors*, i.e., there always exist some inputs where the predictions by the old-version model were correct but the ones by the new-version model were wrong. For example, if one upgrades the ResNet-50 model [6] to the DenseNet-169 model [7], there will be up to 9.44% inputs in the ImageCLEF dataset [8] that cause regression errors.

Although a deep learning model never promises to be always correct, regression errors can be especially damaging in practice. Consider an AI-assisted medical scenario where a DNN model is incorporated to help a doctor decide whether a patient needs to be hospitalized. At the beginning, the doctor does not fully trust the model and makes her own verification. Gradually, she becomes familiar with the model's behavior, being able to identify situations where the model excels, and only needs to take more cautions in other cases. Essentially, the doctor learns a *mental model* [9, Chap. 1] for the DNN model's reliability. Now, suppose the model is upgraded to a new version with a higher overall accuracy. Regression errors may invalidate the doctor's mental model, so she has to re-adapt to the new version, incurring additional risk of wrong decisions. The negative impact of regression errors are also discussed in [10], [11], including the update to fix Tesla's braking system leaves users a bad impression that the fix slows down cars, and the update of user's photo search app fluctuates over the search results causing downvotes in the app store, etc.

Reducing regression errors has been largely ignored by the software engineering community. Although there are some recent efforts in the machine learning community [10]–[13], these approaches are often heavyweight and less friendly to software engineers because they require a large amount of *labeled* data as well as model retraining expertise. In addition, these approaches usually trade model accuracy for backward compatibility (i.e., regression error reduction), which diminishes the benefit of model upgrading.

In this work, we propose to tackle the regression error reduction problem of DNNs with two goals. First, we aim to deliver a solution under more practical constraints. That is, we require only a limited amount of *unlabeled* data or even no data. We also avoid model retraining or fine-tuning, and thus there is no need of strong training expertise or computation resources. Second, we ambitiously aim at reducing regression errors without sacrificing the accuracy of the new-version model, i.e., to achieve a Pareto improvement.

To achieve the first goal, a promising (and perhaps the only) solution is to directly combine the predictions of old-version and new-version models through ensemble techniques. However, it is non-trivial for off-the-shelf ensemble techniques to satisfy the second goal, as they require the base models to have comparable accuracies [14], which is not the case in the regression reduction problem. To overcome this issue, the key insight of our solution comes from an in-depth understanding and handling of the *uncertainty* in deep learning models. There are two levels of uncertainties associated with each prediction (output) of the model [15]. First, the model is uncertain about its prediction, and this uncertainty is usually quantified by *probability* or *confidence* (e.g., a model may say that the image is a dog with 90% confidence). Second, note that the value of this confidence itself is only a rough approximation without explicit characterization, which introduces the second level of uncertainty (i.e., the reliability of the confidence). While it is natural to combine the outputs of the two models based on their confidence values to reduce regression errors, ignoring the second-level uncertainty will easily compromise the overall prediction accuracy. To understand where the problem is, simply consider the risk of yielding to a strong opinion of a conceited newbie for a modest opinion of a humble expert. To achieve a *safe* conclusion, one must estimate the expertise (second-level uncertainty) and align the opinions before combining them.

Based on the above insight, we propose to quantify the second-level uncertainty so as to align the first-level uncertainty to reduce DNN regressions errors. Specifically, we consider two practical cases. In the first case where no data is available, we propose to quantify the uncertainty with the variance of the confidence values for the given input. Technically, we estimate the variance via input perturbation or model perturbation. In the second case, where one can collect a very small amount of *unlabeled* data, we can do better by calibrating the confidence of the old-version model against the new-version model, and safely combining the two models for a Pareto improvement. We name the above two approaches as *data-free* and *label-free* approaches, respectively.

We conduct extensive empirical evaluations with different model upgrade settings to evaluate the general efficacy of our approaches. On the one hand, our results show that existing proposals, including ensemble methods and training methods, sacrifice too much accuracy (up to 20.8%) in exchange for regression error reduction. In fact, their performance is sometimes even worse than that of a simple baseline (i.e., proportional random model selection). On the other hand, our approaches all outperform the existing methods. Our data-free approach reduces 32.7% (via input perturbation) and 29.1% (via model perturbation) regression errors with little or even no accuracy loss (at most 1.8%). Our label-free approach successfully yields Pareto improvements in all cases, i.e., achieving a regression error reduction rate of 48.1% on average without sacrificing accuracy.

The main contributions of this paper include:

- We raise and analyze the problem of DNN regression error reduction from a software engineering perspective. In particular, we highlight the important role of uncertainty in its solution.
- We propose novel, principled, and lightweight data-free and label-free approaches to reduce regression errors. Both approaches can provide Pareto improvements that reduce regression errors without sacrificing model accuracy.
- We carry out extensive experiments with various representative tasks, models, and datasets; the results confirm the efficacy of our approaches compared with the existing alternatives.

Structure. The rest of this paper is organized as follows. Section II formalizes the problem of regression error reduction and analyzes the technical challenges. Section III presents the ensemble-based, uncertainty-aware solutions in both data-free and label-free settings, and Section IV shows evaluations of our approaches. Section V discusses the related work, and Section VI concludes this work.

II. Regression Errors of DNN Models

In this section, we formalize the problem of regression error reduction, and analyze its technical challenges as well as the limitations of existing work. For easier understanding, we start with an illustrative example to show our key insight.

A. An Illustrative Example

Let us consider the following metaphoric scenario:

> *Two reviewers, a junior and a senior, are solicited to judge whether a submission to an academic conference shall be accepted or rejected. In addition to a yes/no recommendation, each reviewer is required to give a fine-grained recommendation score which quantifies the reviewer's (un)certainty about his/her recommendation (i.e., the first-level uncertainty).*
>
> *The PC Chair needs to make a final decision in case of divergent recommendations. The simplest solution is to average the two recommendation scores. However, reviewers' recommendation scores themselves are highly uncertain due to issues such as time, expertise, and enthusiasm, introducing the second-level of uncertainty. If the recommendation scores are unbalanced with uncertainty, the simple average would not work well. That is why in the real world the reviewers are asked to provide a second level of the (un)certainty (i.e., the "expertise" level in many review forms), based on which some weighted average strategy is used to make a decision.*

As an analogy between our DNN regression error problem and this example, one can view the DNN models as reviewers, and DNN confidence values as fine-grained recommendation scores. However, DNN models, unlike rational reviewers, do not provide a measure of the second-level uncertainty. If this information were missing, the Chair would have to make her own deliberation. To achieve a *safe* combination of the recommendation scores, she may align the uncertainties by, for example, identifying the humbleness and ego of the reviewers, and then scaling the scores accordingly.

B. Problem Formulation

We consider the typical K-label classification task, where a deep neural network classifier M is often devised to learn a posterior probability distribution over the label space. Namely, M outputs an estimated probability distribution conditioned on the input $\mathbf{x}$ by applying softmax on the logit $\mathbf{z}$, i.e.,

$$p_M(\mathrm{y} = k|\mathbf{x}) = \mathrm{Softmax}(\mathbf{z})_k,\ k = 1, \ldots, K.$$

We further use $p_M(\mathbf{y} \mid \mathbf{x})$ to represent a vector of the estimated probabilities for each class. The final predictive result for $\mathbf{x}$ is the class with the maximum posterior probability, i.e.,

$$f_M(\mathbf{x}) = \underset{k=1,\ldots,K}{\arg\max}\, p_M(\mathrm{y} = k|\mathbf{x}),$$

and the maximum probability is referred to as the *confidence* for this prediction [16]. We also use M_{old} (resp. M_{new}) to represent the model of the old (resp. new) version.

We now formulate the regression error reduction problem. For a given input $\mathbf{x}$ with its actual label y, we use an indicator function $\mathbb{I}(f_M(\mathbf{x}) = \mathrm{y})$ to represent whether $\mathbf{x}$ is correctly classified by M. The accuracy of M is defined as

$$\mathsf{Acc}(M) = \mathbb{E}_{(\mathbf{x},\mathrm{y})\sim\mathcal{D}}[\mathbb{I}(f_M(\mathbf{x}) = \mathrm{y})],$$

where $\mathcal{D}$ is the distribution of the population. Usually, we upgrade M_{old} to M_{new} for better accuracy, which means that, in the distinct predictions of the two models, the majority will be *positive flips* (PFs) where M_{new} corrects the mispredictions of M_{old}. However, due to the random nature of DNNs, *negative flips* (NFs), or *regression errors*, also emerge where inputs were correctly classified by M_{old} but are misclassified by M_{new}. The accuracy improvement of M_{new} comes from the difference between PFs and NFs. Formally, we define the regression error rate as the expectation of NF[1]:

$$\mathsf{Reg}(M_{old}, M_{new}) = \mathbb{E}[\mathbb{I}(f_{M_{old}}(\mathbf{x}) = \mathrm{y}, f_{M_{new}}(\mathbf{x}) \neq \mathrm{y})].$$

Although the number of NFs is usually smaller than PFs, as discussed in Section I, an NF may cost significantly more than a PF can save due to the price of human-AI re-adaptation. This disproportionate cost can neutralize or even reverse the benefit of model upgrading. Formally, we define the regression error reduction problem as follows.

Problem (Regression Error Reduction through Model Combination)**.** *Given an old-version model M_{old}, and a new-version model M_{new}, obtain a combined model $\widehat{M}$ which admits lower regression error rate $\mathsf{Reg}(M_{old}, \widehat{M})$ without loss of the overall accuracy. The corresponding optimization problem can be formulated as*

$$\min_{\widehat{M}} \mathsf{Reg}(M_{old}, \widehat{M}), \quad \text{s.t.} \quad \mathsf{Acc}(\widehat{M}) \geq \mathsf{Acc}(M_{new}). \qquad \text{(P)}$$

With regard to the accessibility of dataset, since there are increasing concerns in data privacy and proprietary limitations (e.g., EU GDPR and CCPA [17], [18]), a third-party trained model is usually provided *as is*, without access to its training data. Additionally, it is usually too expensive to collect and label enough data from the application domain during model upgrading, making effective re-training or fine-tuning on M_{new} impractical.

Instead, we aim at regression error reduction in two practical cases: (1) the data-free case where no data from original training distribution is required; and (2) the label-free case, where only a limited volume of unlabeled data sampled from the application domain $\mathcal{D}$ is needed. In such cases, combining models M_{old} and M_{new} to reduce regression errors is the only practical choice. Note that although there exist a plethora of ensemble learning techniques [19], [20], as far as we know none of them work well in these data-limited cases.

C. Technical Challenges

The key challenge is how to reduce regression errors without sacrificing model accuracy. To understand the challenge, let us first look at the existing model combination solutions.

The first mode combination strategy is to simply select the model with higher confidence [21, Sec. 13.2], i.e.,

$$f_{\widehat{M}}(\mathbf{x}) = \begin{cases} f_{M_{old}}(\mathbf{x}) & \text{if } c_{M_{old}}(\mathbf{x}) > c_{M_{new}}(\mathbf{x}), \\ f_{M_{new}}(\mathbf{x}) & \text{otherwise.} \end{cases}$$

Analogously, this strategy can be seen as deciding the recommendation of a submission by following the reviewer whoever has a stronger recommendation score. It is not hard to see that the Chair can be easily misled by a conceited reviewer.

Similarly, the Chair can make a simple average

$$p_{\widehat{M}}(\mathbf{y}|\mathbf{x}) = \frac{p_{M_{old}}(\mathbf{y}|\mathbf{x}) + p_{M_{new}}(\mathbf{y}|\mathbf{x})}{2}$$

and decide accordingly. However, this strategy only works well when the two models have aligned uncertainty in confidence [19], [22]. In practice, the old-version model tends to be significantly outperformed by the new-version model (which is the reason for model upgrade). This implicitly indicates that the old-version model has a higher uncertainty than the new-version model, and hence the simple average often damages the accuracy of the upgraded model.

Finally, taking the second-level uncertainty into consideration, the Chair can make a weighted average with a scalar parameter α, i.e.,

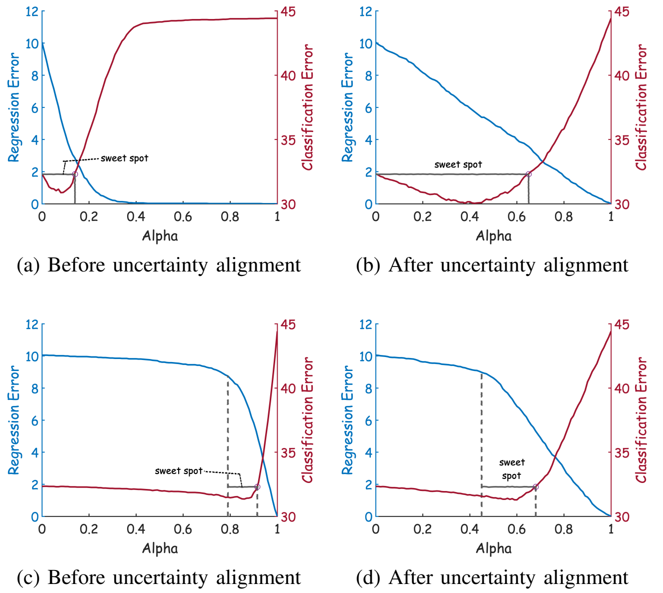
$$p_{\widehat{M}}(\mathbf{y}|\mathbf{x}) = \alpha p_{M_{old}}(\mathbf{y}|\mathbf{x}) + (1-\alpha) p_{M_{new}}(\mathbf{y}|\mathbf{x}).$$

Unfortunately, the optimal α in the context of DNN predictions is very difficult to be determined without any labeled data. To illustrate this point, we conduct a simulated experiment on two synthetic datasets and show the results in Figure 1. Specifically, we upgrade the old-version model AlexNet to the new-version model ResNet-56, and combine the two models with different α's. By the model upgrade, the classification error rate is successfully decreased from 45.6% to 32.6%, but it also introduced regression errors as high as 10.0%. From the two examples (i.e., two rows) in Figure 1, we can observe that when the model uncertainties are misaligned (the left two figures), the range of the optimal α (i.e., the sweet spot) is relatively small, and this range differs for different cases. Therefore, it is very difficult to obtain a general setting strategy of α especially when the labeled data is unavailable. In contrast, with aligned uncertainties (the right two figures), we can simply set $\alpha = 0.5$ to obtain a safe model combination.

III. Uncertainty-Aware Solutions

In this section, we present our uncertainty-aware model combination for the reduction of DNN regression errors. We first discuss the need to align the second-level uncertainty, and then give our solutions for the data-free and label-free cases.

[1] We drop the subscript $(\mathbf{x}, \mathrm{y}) \sim \mathcal{D}$ for simplicity, and the expectation is computed over $\mathcal{D}$ if it is not specified.



(a) Before uncertainty alignment

(b) After uncertainty alignment

(c) Before uncertainty alignment

(d) After uncertainty alignment

Fig. 1. The curves of classification error rate and regression error rate with different α's. The *sweet spot* indicates the optimal α's that lead to regression error reduction without sacrificing accuracy. The two rows correspond to two different examples, showing that the unaligned uncertainties shrink and shift the range of the optimal α, making it very difficult to determine a general setting of α. In contrast, with aligned uncertainties, we can simply set the optimal $\alpha = 0.5$.

A. Rationale of Uncertainty Alignment

A wise PC Chair surely needs to consider the reviewer's expertise (i.e., the second-level uncertainty) before making a final decision. Here, we examine the problem more rigorously to show the rationale behind uncertainty alignment. Consider the mean squared error (MSE) of a model M. It can be decomposed into the two levels of uncertainty, i.e.,

$$\begin{aligned}\mathsf{MSE}(M) &= \mathbb{E}[\|\mathbf{y} - p_M(\mathbf{y}|\mathbf{x})\|^2] \\ &= \underbrace{\mathbb{E}[\|\mathbf{y} - p(\mathbf{y}|\mathbf{x})\|^2]}_{\text{first level uncertainty}} + \underbrace{\mathbb{E}[\|p_M(\mathbf{y}|\mathbf{x}) - p(\mathbf{y}|\mathbf{x})\|^2]}_{\text{second level uncertainty}}, \quad (1)\end{aligned}$$

where $\mathbf{y}$ is the one-hot form of the actual label y, and $p(\mathbf{y}|\mathbf{x})$ stands for the accurate (real) confidence for the prediction. The second-level uncertainty can be further decomposed by using the bias-variance decomposition,

$$\begin{aligned}&\mathbb{E}[\|p_M(\mathbf{y}|\mathbf{x}) - p(\mathbf{y}|\mathbf{x})\|^2] \\ &= \mathbb{E}_{\mathbf{x}}[\underbrace{\|p(\mathbf{y}|\mathbf{x}) - \mathbb{E}_{\mathbf{y}|\mathbf{x}}[p_M(\mathbf{y}|\mathbf{x})]\|^2}_{\text{bias}^2} + \underbrace{\mathrm{Var}[p_M(\mathbf{y}|\mathbf{x})]}_{\text{variance}}]. \quad (2)\end{aligned}$$

The detailed computation of Eqs. (1) and (2) can be found in Appendix A. Furthermore, by taking MSE as the surrogate loss for the original accuracy loss, we can approximately reformulate Problem (P) as

$$\min_{\widehat{M}} \mathsf{Reg}(M_{old}, \widehat{M}), \quad \text{s.t.} \quad \mathsf{MSE}(\widehat{M}) \leq \mathsf{MSE}(M_{new}). \quad (3)$$

Since the first-level uncertainty is intrinsically due to the task itself, statistically it shall be similar for different models. Therefore, the accuracy difference between models is largely decided by their second-level uncertainty. Actually, we have the following proposition.

Proposition 1. *If M_{old} and M_{new} have aligned second-level uncertainties, i.e.,*

$$\mathbb{E}[\|p_{M_{old}}(\mathbf{y}|\mathbf{x}) - p(\mathbf{y}|\mathbf{x})\|^2] = \mathbb{E}[\|p_{M_{new}}(\mathbf{y}|\mathbf{x}) - p(\mathbf{y}|\mathbf{x})\|^2],$$

it holds that $\mathsf{MSE}(M_{old}) = \mathsf{MSE}(M_{new})$. In this case, the ensemble model $\widehat{M}$ derived by simple average, achieves the minimum of $\mathsf{MSE}(\widehat{M})$.

The proof of the proposition can be adapted from [23]. The above proposition essentially proves that we could simply set $\alpha = 0.5$ for a safe model combination if the uncertainties are aligned. Note that gearing toward a minimum $\mathsf{MSE}(\widehat{M})$ means less PFs are sacrificed when we reduce regression errors by incorporating M_{old} into $\widehat{M}$.

B. Variance Estimation in Data-free Cases

Here, we first show how to reduce regression errors when no additional data is available. Note that it is infeasible to directly *solve* Problem (3) without any data. In this work, we propose to align the uncertainty of the M_{old} and M_{new} for each input by estimating their prediction variance. The intuitive analogy to the PC reviewing process is as follows. If a reviewer gives divergent recommendation scores for similar papers that are deemed to be of close quality, his/her recommendation scores shall be less reliable. With this intuition, the PC Chair can estimate the expertise of each reviewer, and scale their recommendation scores accordingly to make a simple average work following Proposition 1.

More rigorously, by applying the bias-variance decomposition in Eq. (2) for the second-level uncertainty, and dropping the bias term considering the low-biased property of DNN models [24], the inequality constraint in Eq. (3) can be reduced to

$$\mathbb{E}_{\mathbf{x}}[\mathrm{Var}[p_{\widehat{M}}(\mathbf{y}|\mathbf{x})]] \leq \mathbb{E}_{\mathbf{x}}[\mathrm{Var}[p_{M_{new}}(\mathbf{y}|\mathbf{x})]]. \quad (4)$$

This formula motivates us to adapt the variance in a pointwise way since it can hold if $\widehat{M}$ achieves lower variance for each prediction. Elaborately, for a given input $\mathbf{x}$, if we know the variances of M_{old} and M_{new}, i.e.,

$$\begin{cases}\sigma^2_{M_{old}}(\mathbf{x}) = \mathrm{Var}[p_{M_{old}}(\mathbf{y}|\mathbf{x})], \\ \sigma^2_{M_{new}}(\mathbf{x}) = \mathrm{Var}[p_{M_{new}}(\mathbf{y}|\mathbf{x})],\end{cases}$$

the simple average of two models' predictions scaled in respective with α_1 and α_2, where

$$\begin{cases}\alpha_1^2(\mathbf{x}) = \left(\sigma^2_{M_{old}}(\mathbf{x}) + \sigma^2_{M_{new}}(\mathbf{x})\right)/2\sigma^2_{M_{new}}(\mathbf{x}), \\ \alpha_2^2(\mathbf{x}) = \left(\sigma^2_{M_{old}}(\mathbf{x}) + \sigma^2_{M_{new}}(\mathbf{x})\right)/2\sigma^2_{M_{old}}(\mathbf{x}),\end{cases} \quad (5)$$

can preserve the lowest variance. We conclude the result in the following proposition.

Proposition 2. *The variance inequality in Eq.* (4) *holds, if $\widehat{M}$ is the simple average of $M_{old}^{\alpha_1}$ and $M_{new}^{\alpha_2}$, where $M_{old}^{\alpha_1}$*

and $M_{new}^{\alpha_2}$ represent M_{old} and M_{new} scaled by α_1 and α_2 pointwisely, i.e.,

$$\begin{cases} p_{M_{old}^{\alpha_1}}(\mathbf{y}|\mathbf{x}) = p_{M_{old}}(\mathbf{y}|\mathbf{x})/\alpha_1(\mathbf{x}), \\ p_{M_{new}^{\alpha_2}}(\mathbf{y}|\mathbf{x}) = p_{M_{old}}(\mathbf{y}|\mathbf{x})/\alpha_2(\mathbf{x}). \end{cases} \tag{6}$$

Furthermore, if the biases of M_{old} and M_{new} nearly vanish (≈ 0), the inequality in Eq. (3) *also holds.*

The proof of this proposition is not difficult, and one can refer to [25, Chap. 4]. Actually, it can be observed that the variances of $M_{old}^{\alpha_1}$ and $M_{new}^{\alpha_2}$ are aligned to the same scale:

$$\sigma^2_{M_{old}^{\alpha_1}}(\mathbf{x}) = \sigma^2_{M_{new}^{\alpha_2}}(\mathbf{x}) = \frac{2\sigma^2_{M_{old}}(\mathbf{x}) \cdot \sigma^2_{M_{new}}(\mathbf{x})}{\sigma^2_{M_{old}}(\mathbf{x}) + \sigma^2_{M_{new}}(\mathbf{x})}.$$

It is also worth noting that the simple average of such scaling is equivalent to the inverse-variance weighting, which attains the lowest variance when the correlation between M_{old} and M_{new} is inaccessible.

Now, the rest issue is how to estimate the variance for a given input, for which we propose to apply the following two implementation techniques [26].

(a) Perturb the input via random noise [27]:

$$\hat{\sigma}^2(\mathbf{x}) = \mathrm{Var}_{\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \delta \boldsymbol{I})}[p_M(\mathbf{y}|\mathbf{x} + \boldsymbol{\varepsilon})]. \tag{7}$$

(b) Perturb the model via dropout [28]:

$$\hat{\sigma}^2(\mathbf{x}) = \mathrm{Var}_{\widetilde{M} \sim \mathrm{D}(M)}[p_{\widetilde{M}}(\mathbf{y}|\mathbf{x})]. \tag{8}$$

These two techniques are very natural, which essentially measure the prediction instability of DNN models under random noise on the input data or model architecture. Besides, the theoretical analysis of their soundness can be found in a recent study [29, Chap. 2.3].

The algorithm of reducing regression errors via variance estimation is summarized in Alg. 1. Given the input $\mathbf{x}$, we first obtain the output distributions via querying the old-version model and the new-version model, and then estimate the variance via either input perturbation or model perturbation. Next, we compute the weights of the two models based on the estimated variance, and combine the outputs of the two models to obtain the final prediction result.

C. Temperature Scaling in Label-free Cases

In cases we can collect a small set of *unlabeled* data from the application domain, we can better handle the problem. The intuitive analogy to the PC reviewing process is as follows. Still consider the conflicting reviews from a junior reviewer and a senior reviewer. The PC Chair can let the junior reviewer, who may have an unnecessary ego, learn from the senior reviewer, who is properly humble, by calibrating the former's recommendation score against the latter's on a set of submissions. In this way, the Chair avoids being misled by the junior reviewer's ego (i.e., inaccurate recommendation scores) in the balancing of the divergent recommendations.

Technically, we adopt temperature scaling to align M_{old} against M_{new}, so that we can get an optimized $\widehat{M}$ by averaging them. Temperature scaling is a simple but effective technique used for various purposes such as confidence calibration, out-of-distribution detection, and model compression [30]–[32]. It divides the logit $\mathbf{z}$ by a scalar T (called temperature) before the softmax layer, i.e.,

Algorithm 1 Data free Regression Error Reduction

```
Input: An old-version model $M_{old}$, a new-version model $M_{new}$, and a given input $\mathbf{x}$.
Output: The prediction $p_{\widehat{M}}(\mathrm{y} \mid \mathbf{x})$.
  Variance Estimation:
  Estimate variances $\hat{\sigma}^2_{M_{old}}(\mathbf{x})$ and $\hat{\sigma}^2_{M_{new}}(\mathbf{x})$ via (7) or (8).
  Compute scaling coefficients
      $\alpha_1^2(\mathbf{x}) = \left(\hat{\sigma}^2_{M_{old}}(\mathbf{x}) + \hat{\sigma}^2_{M_{new}}(\mathbf{x})\right)/2\hat{\sigma}^2_{M_{new}}(\mathbf{x}),$
      $\alpha_2^2(\mathbf{x}) = \left(\hat{\sigma}^2_{M_{old}}(\mathbf{x}) + \hat{\sigma}^2_{M_{new}}(\mathbf{x})\right)/2\hat{\sigma}^2_{M_{old}}(\mathbf{x}).$
  Prediction Combination:
  Compute the outputs $p_{M_{old}}(\mathbf{y}|\mathbf{x})$ and $p_{M_{new}}(\mathbf{y}|\mathbf{x})$, and then scale the two predictions by (6).
  Output the combined prediction
      $p_{\widehat{M}}(\mathbf{y}|\mathbf{x}) = \frac{1}{2}p_{M_{old}}(\mathbf{y}|\mathbf{x}) + \frac{1}{2}p_{M_{new}}(\mathbf{y}|\mathbf{x}).$
```

$$p_M^{(T)}(\mathbf{y}|\mathbf{x}) = \mathrm{Softmax}(\mathbf{z}_M/T).$$

The temperature T is a positive constant that controls the "sharpness" or "softness" of the class probability. Specifically, a smaller value of T ($T \to 0$) collapses the class probability to a single point mass (i.e., a one-hot vector), whereas a higher value ($T \to \infty$) encourages uniform distribution.

One specific property of the temperature scaling is that it does not change the maximum of the softmax output, and thus preserves the model's predictions. As a result, we have

$$\mathsf{Reg}(\widehat{M}, M_{old}^{(T)}) = \mathsf{Reg}(\widehat{M}, M_{old}),$$

where $M_{old}^{(T)}$ represents the model M_{old} applied by temperature scaling with given temperature T. Therefore, we can equivalently minimize the regression error $\mathsf{Reg}(M_{old}^{(T)}, \widehat{M})$ instead of $\mathsf{Reg}(M_{old}, \widehat{M})$, and obtain the following optimization problem,

$$\min_{\widehat{M}} \mathsf{Reg}(M_{old}^{(T)}, \widehat{M}), \quad \text{s.t.} \quad \mathsf{MSE}(\widehat{M}) \leq \mathsf{MSE}(M_{new}), \tag{9}$$

for which we have the following proposition.[2]

Proposition 3. *If we have $\mathsf{MSE}(M_{old}^{(T^*)}) \approx \mathsf{MSE}(M_{new})$ for the optimal temperature T^*, the model $\widehat{M}$ constructed by the simple average of $M_{old}^{(T^*)}$ and M_{new} ensures that the inequality in Eq.* (9) *holds. Moreover, $\widehat{M}$ achieves the minimum of* $\mathsf{MSE}(\widehat{M})$.

[2] This proposition is a direct result of Proposition 1. We provide several analyses in Appendix B to further illustrate our temperature scaling method.

Algorithm 2 Label-free Regression Error Reduction

```
Input: An old-version model $M_{old}$, a new-version model $M_{new}$, an unlabeled dataset $D$, and the given input $\mathbf{x}$.
Output: The prediction $p_{\widehat{M}}(\mathbf{y}|\mathbf{x})$.

Temperature scaling:
if $T^*$ not exist then
    Solve $T^* = \arg\min_T \mathsf{MSE}(M_{new}, M_{old}^{(T)})$, where MSE is empirically computed based on dataset $D$.
end if

Prediction Combination:
Compute the outputs $p_{M_{old}^{(T^*)}}(\mathbf{y}|\mathbf{x})$ and $p_{M_{new}}(\mathbf{y}|\mathbf{x})$.
Output the combined prediction
    $$p_{\widehat{M}}(\mathbf{y}|\mathbf{x}) = \frac{1}{2} p_{M_{old}^{(T^*)}}(\mathbf{y}|\mathbf{x}) + \frac{1}{2} p_{M_{new}}(\mathbf{y}|\mathbf{x}).$$
```

Essentially, the temperature T is used to align the uncertainty of M_{old} to that of M_{new},[3] which prevents the simple average from the loss of accuracy. To obtain such a T^* in label-free cases, we solve the following minimization problem instead,

$$\min_T \mathsf{MSE}(M_{old}^{(T)}, M_{new}), \tag{10}$$

where

$$\mathsf{MSE}(M_{old}^{(T)}, M_{new}) = \mathbb{E}[\|p_{M_{old}^{(T)}}(\mathbf{y}|\mathbf{x}) - p_{M_{new}}(\mathbf{y}|\mathbf{x})\|^2].$$

This optimization problem can be solved using a small set of unlabeled data. If the optimal value of this optimization problem is small enough, it can be easily derived that $\mathsf{MSE}(M_{old}^{(T^*)}) \approx \mathsf{MSE}(M_{new})$ via the triangle inequality. To achieve the global minimum of this subproblem, we apply the quasi-newton method L-BFGS with several initial points [33], [34].

The regression error reduction algorithm via temperature scaling is presented in Alg 2. Given the old-version and new-version models, we first solve Eq. (10) using the unlabled dataset. Note that the temperature scaling step only needs to be executed once, and thus the running time is utterly acceptable. We then compute the output distributions of the given input $\mathbf{x}$ from the old-version model calibrated by temperature scaling as well as the new-version model. These two distributions are averagely weighted to form the final output.

IV. Empirical evaluation

In this section, we carry out experiments to evaluate our regression error reduction approaches. We mainly focus on the following four research questions:

RQ1 Do our approaches perform in line with the theoretical expectations, i.e., do they achieve a Pareto improvement?

RQ2 Are our approaches more effective compared with the alternative approaches?

RQ3 Can our approaches be combined with existing training techniques proposed for regression error reduction?

RQ4 Is our label-free approach consistently effective when the amount of available data varies?

Implementation. We implemented our approaches via the PyTorch DL framework. The experiments were conducted on a GPU server with two Intel Xeon Gold 5118 CPU@2.30GHz, 400GB RAM, and 9 GeForce RTX 2080 Ti GPUs. The server ran Ubuntu 16.04 with GNU/Linux kernel 4.4.0. For the variance estimation approaches, we set the magnitude of the random noise to 0.01, and the dropout is executed before the last fully-connected layer of each model with rate 0.8. For the temperature scaling approach, we use three initial points ($T = 1.0, 1.5$, and 2.0) to start the L-BFGS algorithm. One can refer to our project website (https://github.com/Lizn-zn/Uncertainty-Alignment) for other implementation details.

Inference time cost. Leveraging parallel computation for both input/model perturbation and model inference, our approach barely induces additional time cost. For the tasks in our experiments, the inference time for each input varies from about 0.33ms to 33ms, which is consistent with that of only adopting the new models (0.26ms to 34ms)

A. Experimental Tasks

There are two possible reasons that may lead to the upgrade of DNN models. (1) More data is collected from the application domains. For example, the old-version model was trained on the data collected a few years ago, and it calls for an upgrade to better classify the up-to-date examples given that the new data is available now. (2) The architecture of the DNN model is updated. For example, the VGG architecture was very popular nearly ten years ago for image classification. However, new architectures such as ResNet are shown to offer higher accuracy, and hence an upgrade is needed.

To mimic a variety of practical scenarios, we design six tasks of different upgrade setups, including data update, model update, and a mixture of both. To evaluate the general efficacy of our approaches, DNN models with different complexities (ranging from ~10^3 to ~10^7 parameters), architectures (e.g., dense layer and residual blocks), and training algorithms (e.g., SGD and Adam) are employed in our tasks. Both the old- and new-version models are assumed to be provided by third parties, and are trained in the standard way.

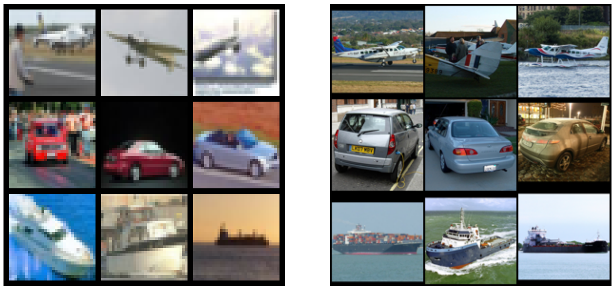
Table I summaries some critical settings in our experiments. Take Task 3 as an example. This task is a mixture of both model update and data update: the old-version model VGG-8 is trained on CIFAR10, and the new-version model ResNet-56 is trained on STL10. The update from CIFAR10 to STL10 mimics camera upgrading, as these two datasets contain images from the same ten classes but with different resolutions (see Figure 2). Since regression error reduction does not rely on the training data of the old-version model, we only specify the statistics of the data used for the new-version model in the table. For the proposed label-free approach, as a small set of unlabeled data is needed, we extract it (henceforth referred to as the validation set) from the test set. Except for Task 6,

[3] Since M_{old} achieves lower accuracy compared to M_{new}, M_{old} should have a higher uncertainty compared to M_{new}, which is also the reason that we only apply the temperature scaling on M_{old}.

TABLE I
THE UPGRADE SETUPS OF DIFFERENT TASKS.

Task	Model	Dataset	Acc. (%)	Reg. (%)	Dataset Size*
1	LeNet5 [35]	MNIST [35]↗EMNIST [36]	35.1↗84.8	3.59	(500/10/990)×10
2	AlexNet [37]↗VGG16 [38]	STL10 [39]	55.0↗73.7	7.01	(500/16/784)×10
3	VGG8 [38]↗ResNet56 [6]	CIFAR10 [40]↗STL10 [39]	44.4↗71.7	9.03	(500/16/784)×10
4	DenseNet40 [7]↗ResNet110 [6]	CIFAR100 (0.2↗0.8) [40]	44.0↗70.6	5.10	(500/20/80)×100
5	ResNet50 [6]↗DenseNet169 [7]	ImageCLEF ((c)↗(p)) [8]	69.1↗75.8	9.44	(38/6/6)×12
6	ResNet18 [6]↗WRN101 [41]	ImageNet [42]↗ObjectNet [43]	17.0↗28.7	3.37	(0/80/80)×113

* Dataset size is recorded in the form of $(a/b/c) \times d$, where $a/b/c$ is the size per class of training/validation/test set, and d is the number of classes. The ObjectNet does not contain training data (https://objectnet.dev/).



(a) CIFAR10

(b) STL10

Fig. 2. Examples of data update from CIFAR10 to STL10 in Task 3. Images resolutions increase from 32×32 to 96×96.

we set the size of the validation set to around 6-20 per class. All the approaches are evaluated on the rest of the test set for fairness.

For all the tasks, we report two evaluation metrics, i.e., the model accuracy (Acc) and the regression error rate (Reg).

B. Experimental Results

1) RQ1: Pareto Imprvoment: We first compare the proposed methods with the original new-version model (*Orig*), and show the results in Table II. For our approaches, we denote the variance estimation via perturbation by *VE-P*, the variance estimation via dropout by *VE-D*, and the temperature scaling by *TS*.

As shown in Table II, all our methods significantly reduced the regression error rate while preserving the accuracy. Elaboratively, VE-P improved the accuracy in all tasks except Task 4 (from 70.6% to 69.9%, down by 0.7%); meanwhile, it achieved significant regression error reduction compared to the respective new-version model (lowest 18.5% in Task 2 and highest 44.1% in Task 1). VE-D incurred at most 1.8% accuracy loss among six tasks, while it obtained relatively higher regression error reduction rates in Task 1 (52.1%) and Task 3 (54.0%). For TS, it did not sacrifice accuracy in any task, and attained the highest average regression error reduction rate (48.1%) compared with VE-P (32.7%) and VE-D (29.1%).

*To conclude **RQ1**, consistent with our theoretical expectations, our methods significantly reduced regression error rates while preserving the accuracy in most cases. The label-free method strictly achieved Pareto improvements in all tasks, while the data-free method practically achieved Pareto improvements with neglectable accuracy losses in some cases.*

2) RQ2: Effectiveness Comparison: We next compare the proposed methods with existing alternatives. The methods included in the comparison are as follows. It should be emphasized that all the compared ensemble methods and training methods require *labeled* data, and we additionally feed the labels for them.

Simple Methods. First of all, two simple data-free methods, viz. *simple average (Avg)* and *taking maximum confidence (Max)* as explained in Section II-C, are included. Moreover, some test adequacy metrics may be applicable as a surrogate of the original confidence value, since they can quantify the misclassification probability. To this end, we employ *DeepGini (Gini)* [44], i.e., taking the final prediction with the maximum Gini index. (We also try Surprise Adequacy [45], but are unsuccessful due to too limited data.)

Ensemble Methods. We consider the three ensemble methods introduced by Träuble et al. [11], including *MaxBelief (MB)*, *MaxBeliefMinEntropy (MBME)*, and *CostRatio (CR)* (with $c = 5$). All the three methods attempt to combine the old model and the new model based on the uncertainty built in a Bayesian method. The reserved validation data (with labels) is used as input for these ensemble methods.

Training methods. Yan et al. [46] and Xie et al. [13] propose to use *knowledge distillation (KD)* and *logit matching (LM)* to reduce regression errors. We implement these two techniques, and additionally adopt a focal-loss based *weighted-training (WT)* as competitors. WT essentially requires assigning more weights to examples that are correctly classified by the old-version model. The full training data (used for training the new-version model) with labels is provided for these methods.

The accuracy and regression error rate results of the methods

TABLE II
THE ACCURACY AND REGRESSION ERROR RATE RESULTS OF DIFFERENT METHODS (%). FOR OUR APPROACHES, THE RESULTS ARE IN BOLD IF IT REDUCES REGRESSION ERRORS AND PRESERVES AT LEAST 99% ACCURACY.

Task	**Metric**	**Simple**				**Ensemble**			**Training**			**Ours**		
		Orig	*Avg*	*Max*	*Gini*	*MB*	*MBME*	*CR*	*KD*	*LM*	*WT*	*VE-P*	*VE-D*	*TS*
1	Acc	84.8	72.0	70.9	70.4	79.9	79.8	77.5	43.8	32.7	74.4	**85.7**	83.6	**85.7**
	Reg	3.59	0.42	0.64	0.65	2.53	1.64	1.73	4.18	7.44	4.21	**2.01**	1.85	**1.72**
2	Acc	73.7	68.4	68.2	68.1	66.2	66.1	61.3	68.1	67.8	73.1	**74.1**	**73.9**	**73.8**
	Reg	7.01	1.95	2.08	2.08	5.61	5.30	2.97	8.01	7.45	7.49	**5.72**	**5.73**	**4.84**
3	Acc	71.7	55.9	55.7	55.9	68.3	70.6	67.5	44.9	54.7	73.3	**72.5**	69.1	**71.7**
	Reg	9.03	1.69	1.68	1.65	7.15	5.25	4.21	5.96	4.55	7.36	**6.61**	4.16	**5.20**
4	Acc	70.6	65.2	64.7	64.5	51.3	52.2	49.7	54.1	51.7	70.9	**69.9**	**70.1**	**70.6**
	Reg	5.10	1.39	1.52	1.55	4.12	3.46	2.27	4.41	4.91	4.59	**3.11**	**4.37**	**2.01**
5	Acc	75.8	71.8	72.6	70.8	70.5	76.1	74.4	73.7	76.6	75.8	**76.3**	75.0	**75.9**
	Reg	9.44	3.05	3.61	4.16	10.5	6.94	4.02	3.75	8.19	9.58	**6.11**	7.49	**3.19**
6	Acc	28.7	26.9	26.4	27.9	-	-	-	-	-	-	**29.4**	**28.6**	**28.7**
	Reg	3.37	1.73	1.98	2.11	-	-	-	-	-	-	**2.26**	**2.73**	**2.13**

are shown in Table II.[4] We first observe that none of the compared methods yield consistent Pareto improvements, i.e., they reduce regression errors with a significant loss in accuracy in most cases. For the two data-free methods Avg and Max, we can see that they could reduce regression errors but with significant accuracy drops in all cases. The reasons have been discussed in Section II-C. For the Gini, its result is consistent with Avg and Max, since the Gini index is still computed based on the confidence.

Alternative ensemble methods also significantly decrease the accuracy. For example, MB, MBME, and CR relatively decrease the accuracy of Orig (the new-version model) by 11.01%, 8.68%, and 12.55%, respectively. The only exception is MBME in Task 5 where it reduced regression error rate and improved accuracy at the same time. However, MBME is still worse than VE-P in both accuracy and regression error rate. In addition, they performed even worse than the simple Avg and Max in Tasks 2 and 4.

The training methods do not achieve a Pareto improvement in most cases either. The only exceptions are from LM on Task 5 and WT on Tasks 3 and 4. However, in these three cases, the reduction of regression error rate was minor (by 1.25%, 0.51%, and 1.67%, respectively). Contrastingly, our TS method achieved an average of 3.44% reduction. Additionally, these methods may incur significant accuracy drops. For example, KD and LM incurred significant accuracy drops in Tasks 1-4, and KD even did not achieve lower regression error rates than TS with such large accuracy drops. LM only obtained lower regression error rates than TS in Task 3, but with a 17% accuracy drop. In Task 5, the models were obtained via fine-tuning pre-trained models on ImageNet, and hence the accuracy was preserved for the three methods. Nonetheless, in this task, our TS still attained lower regression error rates compared to all these three methods. WT ensured a stable accuracy of the new model in Tasks 2–5, but it did not reduce more regression errors than our methods, and even introduced some new errors in Tasks 1, 2, and 5. We also find that WT achieves nearly zero regression error, but fails to generalize the backward compatibility due to overfitting.

Surprisingly, even with labeled data, ensemble methods and training methods are outperformed by our data-free and label-free methods in many practical settings. In the following, we analyze the failure of these two kinds of methods, respectively. First, the alternative ensemble methods attempts to *reconstruct* an accurate uncertainty estimation. However, the two levels of uncertainty are not clearly distinguished, and the first-level uncertainty, i.e., the confidence provided by the model, is not fully exploited, which leads to their low data efficiency. Second, the training methods (KD and LM) have been proved to be essentially a simple average of old- and new-version models [47], and thus also suffer from the issue of unaligned uncertainties.

Next, to compare the relative effectiveness of the methods, we consider the *exchange rate* $\frac{\Delta PF}{\Delta NF}$ which is the ratio of the number of sacrificed positive flips to the number of reduced negative flips. The smaller the exchange rate, the better. For Pareto improvement, this ratio must be no greater than 1. As different methods strike different trade-offs between regression error reduction and accuracy loss, we also introduce a conceptual baseline for fair and intuitive comparison. Suppose we randomly select M_{old} (with probability p) or M_{new} (with probability $(1-p)$) as $\widehat{M}$ to predict over the current input.

[4] The results of the ensemble methods and training methods for Task 6 are not shown. The ensemble methods need a large set of data to output an accurate Bayesian estimate of the uncertainty, but the validation set is much smaller than the entire ImageNet dataset, rendering a meaningless uncertainty estimate. For training methods, ObjectNet only contains a small test set as mentioned in the task setup.

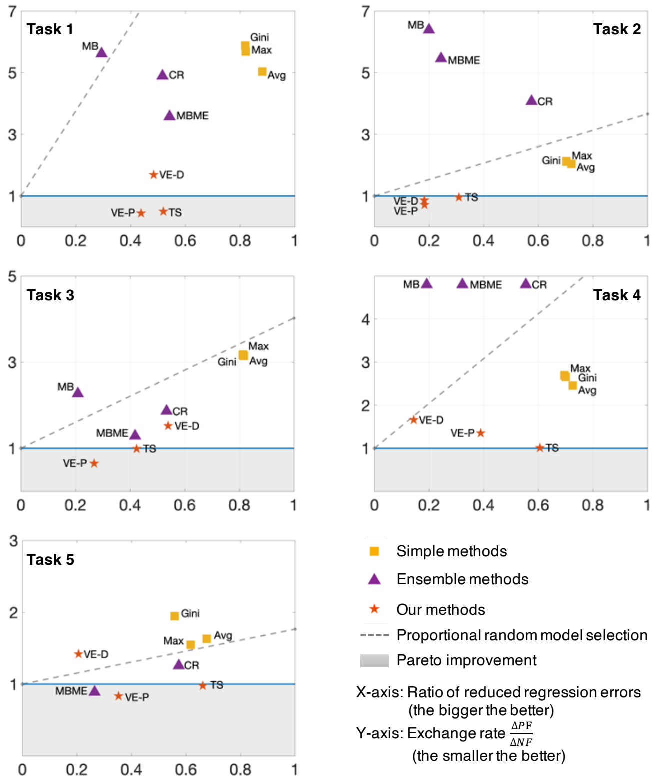


Fig. 3. Exchange rates of different methods. Particularly, for MB and MBME, they are Pareto-dominated by TS in all tasks. CR is also Pareto-dominated by TS in Task 1, 4, and 5, and incurs large accuracy sacrifice in Task 2 and 3.

TABLE III
THE ACCURACY OF DIFFERENT METHODS (%).

Case	ResNet-34 ↗ ResNet-56	ResNet-56 ↗ ResNet-34
Avg	80.3	80.3
Max	80.0	79.9
KD	78.5	78.0
LM	79.2	80.0
TS	80.2	80.9
KD+TS	79.0	80.0
LM+TS	79.3	80.4

*Results of Avg and Max should be symmetric in the two cases, but the randomness leads to a very slight difference.

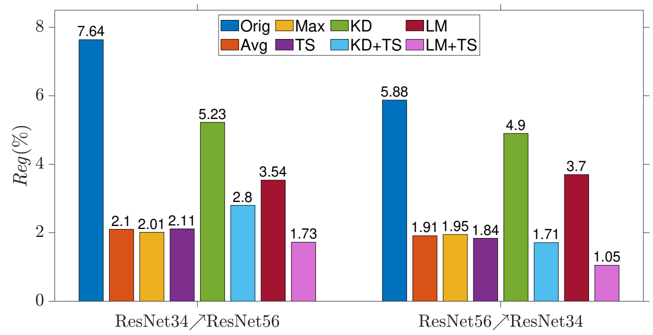


Fig. 4. Regression error rates of different methods. Avg and Max are effective when two models have aligned uncertainties. TS achieves similar results with them, and one can further reduce regression errors by combining TS with KD.

Then by setting p from 0 to 1 we can have a flexible method reducing 0% to 100% of regression errors. We call this baseline method *proportional random model selection*.

We show the results of exchange rates in Figure 3, where exchange rate (y-axis) is plotted against the percentage of the reduced regression errors (the x-axis). The result of the proportional random model selection is plotted as the dash line. Notice that the training methods are omitted in Figure 3 since they are unable to achieve Pareto improvement in most cases. From Figure 3, we can see that our methods are optimized for exchange rate and give priority to preserving accuracy. When the damage of regression errors is so severe which warrants some accuracy sacrifice, one can easily combine our methods with the Avg, Max methods for more regression error reduction.

*To conclude **RQ2**, our methods are more effective than the existing competitors, even in the case where these competitors use extra labeled training data.*

3) RQ3: Combination with Existing Techniques: We study a special case in which the old-version model and the new-version model have close performance. This situation is what the training methods, KD and LM, are originally designed for.

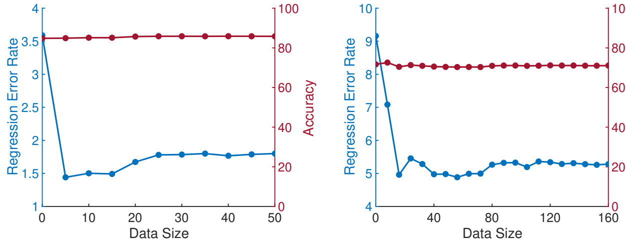
Theoretically, the second-level uncertainties of two models achieving close accuracy are naturally aligned, and thus our TS will obtain temperature T^* close to 1, which means that TS will degenerate to the simple average. To confirm this result empirically, we conduct two experiments on the CIFAR100 dataset. For details, we trained a ResNet-34 model and a ResNet-50 model, which achieved accuracy 76.8% and 78.4%, respectively. In the first experiment, we used the former as the old model and the latter as the new model, and we exchanged their roles in the second experiment.

The accuracy results of Orig, Avg, Max, KD, LM, and TS are shown in Table III, where we also include results of KD+TS and LM+TS, i.e., applying temperature scaling for models trained by KD and LM.

The results of regression error reduction are shown in Figure 4. We can first observe that TS can achieve close results with Avg and Max, just as predicted by theory. For KD and LM, although we successfully reproduce their results, they cannot achieve lower regression error rates compared with others. Second, we can observe that, combining TS and the training methods can further reduce regression errors, especially for LM+TS. This combination achieves the lowest regression error rates in both two experiments.

*To conclude **RQ3**, combining the proposed approach TS with existing training methods KD and LM can further reduce the regression errors, even in cases when these methods have been effective in reducing regression errors.*

4) RQ4: Efficiency of TS: We finally investigate the data efficiency of our label-free method TS, i.e., how many unlabeled examples are needed to effectively determine the temperature T. We conduct experiments in Tasks 1 and 3 for brevity.



(a) Task 1

(b) Task 3

Fig. 5. Regression error rate and accuracy of different data size. The data size of zero represents the results of Orig. It can be observed that TS still works well with extremely limited data.

Specifically, in Task 1, we apply the TS method on the data whose size ranges from 5 to 50 with interval 5, and in Task 3, the data size is in the range from 8 to 160 with interval 8.

Intuitively, only a scalar T is involved in the TS method, and thus it should not need many examples. The results in Figure 5 also confirm this point. In Task 1, the regression error rate has achieved the minimum even when only five unlabeled examples are provided. The regression error rate and accuracy, together with temperature T, converge to stable results at the data size of 30. In Task 3, the regression error reduction is also effective with only eight unlabeled examples, and TS successfully converges by using about 120 examples.

*To conclude **RQ4**, the proposed approach TS is still effective even with a very limited set of unlabeled data, and it easily converges as more data are collected.*

V. Related work

We briefly discuss related work from the software engineering and machine learning communities.

Software regression testing and debugging. Software *regression* refers to the situation where a software update breaks a feature that worked before the update. Regressions in traditional software are viewed as newly introduced bugs that need to be exposed, localized, and fixed. So the research efforts have been focused on improving the efficiency and efficacy of regression testing [2]–[4], [48] and debugging [5], [49]–[51]. On the contrary, an individual regression case of an upgraded DNN model is *not* a bug in the traditional sense because of the statistical nature of DNN [52]–[54]. It is also generally infeasible to *explain* or *localize* the regression with the constitutes of a DNN model [55]–[57]. That's why we propose non-intrusive model combination approaches to reduce DNN regressions.

Regression errors of DNN-based systems. In the software engineering community, little effort has been put on the problem of DNN regression error reduction [58]. One may wonder if error-inducing input detection techniques for DNNs [44], [45], [59] could be used to reduce regression errors. Unfortunately, they are not suitable for this purpose because their quantifications of misclassification probability are not systematically calibrated [31], [60], which makes their uncertainty far from aligned. Our experiments with DeepGini [44] confirmed this. Moreover, these quantifications are all too sensitive to data drift [61], [62], which also makes them ineffective in practical settings.

In the machine learning community, related techniques, unlike ours, require labeled data to work. They fall into three categories. The first is retraining-based approaches [13], [46], [63]. They are mainly designed for the regression cases caused by catastrophic forgetting [64], and thus employ knowledge distillation or weighted training to overcome this forgetting. However, as shown by a recent theoretical analysis [47], these retraining strategies are essentially simple average ensemble, and thus limited to the case that the old- and new-version models have similar performance.

The second category is ensemble-based approaches [11]. A typical example is applying the classic Bayesian ensemble strategy to reduce regression errors. Nevertheless, this strategy wastes the confidence information provided by the model and incurs a high cost of data labeling. For other possible choices, the boosting strategy has been shown to be ineffective for DNN models [47], and the bagging strategy requires that base models should have aligned second-level uncertainty [19], which does not hold in general model upgrade cases.

The third category contains research on the measurement of backward compatibility. The compatibility problem of an updated AI software is regarded as its inconsistency with the user's prior experience, and various metrics are proposed to quantify this inconsistency [10], [65]. As highlighted by a recent empirical analysis [12], the challenge of backward compatibility comes from the data shifts when model deployed in application. The experimental results also indicate that DNN models' confidence is insufficient for regression error detection, which confirms the necessity of the consideration of the second-level uncertainty.

Decision under uncertainty. Uncertainty takes an increasingly important role in modern computing systems, and it increasingly requires explicit management rather than being abstracted away [66]. In the software engineering community, researchers of self-adaptive software systems have a particular interest in uncertainty management [67], [68]. However, they usually stay at the first level of uncertainty.

In the machine learning community, uncertainty is better understood and managed. The confidence of DNN models (i.e., the first-level uncertainty in this paper) is used in a wide range of scenarios such as sample selection [52], [69], [70], out-of-distribution, and adversarial detection [32], [71]. Nevertheless, the second-level uncertainty (i.e., "uncertainty" of the confidence) is still often ignored [72].

Our work is also closely related to the mixture of experts (MoE) [73]–[76], and three straightforward combination solutions discussed in Section II-C actually correspond to the stochastic selection, winner-takes-all, and weights introduced in MoE [20]. In contrast to common ensemble methods assuming that models have aligned uncertainty, MoE methods also consider the second-level uncertainty (i.e., the biased experts). However, existing MoE methods deal with this issue by training a gate network to combine the experts, which is infeasible in our data-free and label-free setting [77].

VI. Conclusion

The current paper is devoted to the reduction of regression errors emerged during DNN model upgrading, which, we believe, is an important software engineering problem for machine learning applications. On the one hand, the problem resembles the classic software regression but requires a different mindset to solve. On the other hand, solutions from the machine learning community ignore the practical engineering difficulties (e.g., data availability). Leveraging uncertainty-aware ensemble, we propose data-free and label-free solutions that are not only practically feasible and effective, but also theoretically appealing in guaranteeing Pareto improvement. More generally, we would advocate embracing the uncertainty that is intrinsic in deep learning models which we believe will play a central role in developing new-generation software artifacts.

Appendix

A. The Derivation of Equations (1) and (2)

First, the decomposition of MSE (Eq. (1)) is derived by

$$\begin{aligned}\mathsf{MSE}(M) &= \mathbb{E}[\|\mathbf{y} - p(\mathbf{y}|\mathbf{x}) + p(\mathbf{y}|\mathbf{x}) - p_M(\mathbf{y}|\mathbf{x})\|^2] \\ &= \mathbb{E}[\|\mathbf{y} - p(\mathbf{y}|\mathbf{x})\|^2] + \mathbb{E}[p(\mathbf{y}|\mathbf{x}) - p_M(\mathbf{y}|\mathbf{x})\|^2] \\ &\quad + 2\mathbb{E}[(\mathbf{y} - p(\mathbf{y}|\mathbf{x}))(p(\mathbf{y}|\mathbf{x}) - p_M(\mathbf{y}|\mathbf{x}))].\end{aligned}$$

Noting that the last term in the equation can be removed, because the error $(\mathbf{y} - p(\mathbf{y}|\mathbf{x}))$ is independent to both $p(\mathbf{y}|\mathbf{x})$ and $p_M(\mathbf{y}|\mathbf{x})$, and we have $\mathbb{E}_{\mathbf{y}|\mathbf{x}}[\mathbf{y}] = p(\mathbf{y}|\mathbf{x})$ for any given $\mathbf{x}$ by the definition of $p(\mathbf{y}|\mathbf{x})$.

For the second-level uncertainty, we have

$$\begin{aligned}&\mathbb{E}[\|p_M(\mathbf{y}|\mathbf{x}) - p(\mathbf{y}|\mathbf{x})\|^2] \\ &\quad = \mathbb{E}_{\mathbf{x}} \sum_{i=1}^{K} \mathbb{E}_{\mathbf{y}|\mathbf{x}}[(p_M(\mathbf{y}_i|\mathbf{x}) - p(\mathbf{y}_i|\mathbf{x}))^2] \\ &\quad = \mathbb{E}_{\mathbf{x}} \sum_{i=1}^{K} (p(\mathbf{y}_i|\mathbf{x}) - \mathbb{E}_{\mathbf{y}|\mathbf{x}}[p_M(\mathbf{y}_i|\mathbf{x})])^2 + \mathrm{Var}[p_M(\mathbf{y}|\mathbf{x})],\end{aligned}$$

where the variance in Eq. (2) is defined as $\mathrm{Var}[p_M(\mathbf{y}|\mathbf{x})] = \sum_{i=1}^{K} \mathrm{Var}[p_M(\mathbf{y}_i|\mathbf{x})]$. The second equation is obtained by the classic bias-variance decomposition, i.e., using the variance equation $\mathrm{Var}[X] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$ and substituting X by $(p_M(\mathbf{y}_i|\mathbf{x}) - p(\mathbf{y}_i|\mathbf{x}))$.

B. The Proof of Proposition 3

We first give the following proposition, showing that our temperature scaling method will shift $\widehat{M}$ into a safer region.

Proposition 4. *For the model $\widehat{M}$ constructed by the simple average of $M_{old}^{(T^*)}$ and M_{new}, the upper bound of $\mathsf{MSE}(\widehat{M})$ decreases with $\mathsf{MSE}(M_{old}^{(T)}, M_{new})$ decreased.*

Proof. For the model $\widehat{M}$ constructed by the simple average of $M_{old}^{(T^*)}$ and M_{new}, $\mathsf{MSE}(\widehat{M})$ can be computed as

$$\begin{aligned}\mathsf{MSE}(\widehat{M}) &= \mathbb{E}[\|\mathbf{y} - \frac{1}{2} p_{M_{old}^{(T)}}(\mathbf{y}|\mathbf{x}) - \frac{1}{2} p_{M_{new}}(\mathbf{y}|\mathbf{x})\|^2] \\ &= \frac{1}{4}(\mathsf{MSE}(M_{old}^{(T)}) + \mathsf{MSE}(M_{new})) \\ &\quad + \frac{1}{2}\,\mathbb{E}[(\mathbf{y} - p_{M_{old}^{(T)}}(\mathbf{y}|\mathbf{x}))(\mathbf{y} - p_{M_{new}}(\mathbf{y}|\mathbf{x}))],\end{aligned}$$

and $\mathsf{MSE}(M_{old}^{(T)}, M_{new})$ can be computed as

$$\begin{aligned}\mathsf{MSE}(M_{old}^{(T)}, M_{new}) &= \mathbb{E}[\|p_{M_{old}^{(T)}}(\mathbf{y}|\mathbf{x}) - p_{M_{new}}(\mathbf{y}|\mathbf{x})\|^2] \\ &= \mathsf{MSE}(M_{old}^{(T)}) + \mathsf{MSE}(M_{new}) \\ &\quad - 2\,\mathbb{E}[(\mathbf{y} - p_{M_{old}^{(T)}}(\mathbf{y}|\mathbf{x}))(\mathbf{y} - p_{M_{new}}(\mathbf{y}|\mathbf{x}))].\end{aligned}$$

Putting these two equations together, we can obtain that

$$\begin{aligned}\mathsf{MSE}(\widehat{M}) &= \frac{1}{2}(\mathsf{MSE}(M_{old}^{(T)}) + \mathsf{MSE}(M_{new})) \\ &\quad - \frac{1}{4}\mathsf{MSE}(M_{old}^{(T)}, M_{new}).\end{aligned}$$

Using the triangle inequality, we can compute the upper bounds of $\mathsf{MSE}(M_{old}^{(T)})$:

$$\mathsf{MSE}(M_{old}^{(T)}) \le \left(\sqrt{\mathsf{MSE}(M_{new})} + \sqrt{\mathsf{MSE}(M_{old}^{(T)}, M_{new})}\right)^2$$

Therefore, the upper bound of $\mathsf{MSE}(\widehat{M})$ is

$$\begin{aligned}\mathsf{MSE}(\widehat{M}) &\le \mathsf{MSE}(M_{new}) + \frac{1}{4}\mathsf{MSE}(M_{old}^{(T)}, M_{new}) \\ &\quad + \sqrt{\mathsf{MSE}(M_{new}) \cdot \mathsf{MSE}(M_{old}^{(T)}, M_{new})}.\end{aligned}$$

□

In a nutshell, by minimizing $\mathsf{MSE}(M_{old}^{(T)}, M_{new})$, the temperature scaling method strictly controls the upper bound of $\mathsf{MSE}(\widehat{M})$, and thus effectively guarantees the performance of the model $\widehat{M}$.

Acknowledgment

We thank the anonymous reviewers for their insightful comments and suggestions. This work is supported by the National Natural Science Foundation of China (Grants #62025202, #62172199). T. Chen is also partially supported by Birkbeck BEI School Project (EFFECT) and an overseas grant of the State Key Laboratory of Novel Software Technology under Grant #KFKT2022A03. Jingwei Xu (jingweix@nju.edu.cn) and Xiaoxing Ma (xxm@nju.edu.cn) are the corresponding authors.

References

[1] I. Ozkaya, "What is really different in engineering ai-enabled systems?" *IEEE Software*, vol. 37, no. 4, pp. 3–6, 2020.

[2] S. Yoo and M. Harman, "Regression testing minimization, selection and prioritization: a survey," *Software testing, verification and reliability*, vol. 22, no. 2, pp. 67–120, 2012.

[3] R. H. Rosero, O. S. Gómez, and G. Rodríguez, "15 years of software regression testing techniques—a survey," *International Journal of Software Engineering and Knowledge Engineering*, vol. 26, no. 05, pp. 675–689, 2016.

[4] L. Chen, F. Hassan, X. Wang, and L. Zhang, "Taming behavioral backward incompatibilities via cross-project testing and analysis," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, ser. ICSE '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 112–124. [Online]. Available: https://doi.org/10.1145/3377811.3380436

[5] H. Wang, Y. Lin, Z. Yang, J. Sun, Y. Liu, J. Dong, Q. Zheng, and T. Liu, "Explaining regressions via alignment slicing and mending," *IEEE Transactions on Software Engineering*, vol. 47, no. 11, pp. 2421–2437, 2021.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[7] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[8] B. Caputo, H. Müller, J. Martinez-Gomez, M. Villegas, B. Acar, N. Patricia, N. Marvasti, S. Üsküdarlı, R. Paredes, M. Cazorla *et al.*, "Imageclef 2014: Overview and analysis of the results," in *International Conference of the Cross-Language Evaluation Forum for European Languages*, Springer. Springer, 2014, pp. 192–211.

[9] D. Norman, *The design of everyday things: Revised and expanded edition*. Basic books, 2013.

[10] G. Bansal, B. Nushi, E. Kamar, D. S. Weld, W. S. Lasecki, and E. Horvitz, "Updates in human-ai teams: Understanding and addressing the performance/compatibility tradeoff," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 2429–2437.

[11] F. Träuble, J. Von Kügelgen, M. Kleindessner, F. Locatello, B. Schölkopf, and P. V. Gehler, "Backward-compatible prediction updates: A probabilistic approach," in *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[12] M. Srivastava, B. Nushi, E. Kamar, S. Shah, and E. Horvitz, "An empirical analysis of backward compatibility in machine learning systems," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 3272–3280.

[13] Y. Xie, Y.-a. Lai, Y. Xiong, Y. Zhang, and S. Soatto, "Regression bugs are in your model! measuring, reducing and analyzing regressions in nlp model updates," *arXiv preprint arXiv:2105.03048*, 2021.

[14] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[15] Y. Gal *et al.*, "Uncertainty in deep learning," 2016.

[16] T. Silva Filho, H. Song, M. Perello-Nieto, R. Santos-Rodriguez, M. Kull, and P. Flach, "Classifier calibration: How to assess and improve predicted class probabilities: a survey," *arXiv e-prints*, pp. arXiv–2112, 2021.

[17] Council of European Union, "Council regulation (EU) no 269/2014," 2014.

[18] C. S. Legislature, "Privacy: personal information: businesses," 2018.

[19] Z.-H. Zhou, *Ensemble methods: foundations and algorithms*. CRC press, 2012.

[20] L. I. Kuncheva, *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons, 2014.

[21] T. Roughgarden, "Algorithmic game theory," *Communications of the ACM*, vol. 53, no. 7, pp. 78–86, 2010.

[22] T. Hastie, R. Tibshirani, and J. Friedman, "The elements of statistical learnin," *Cited on*, p. 33, 2009.

[23] S. Hashem, "Optimal linear combinations of neural networks," *Neural networks*, vol. 10, no. 4, pp. 599–614, 1997.

[24] Z. Yang, Y. Yu, C. You, J. Steinhardt, and Y. Ma, "Rethinking bias-variance trade-off for generalization of neural networks," in *International Conference on Machine Learning*. PMLR, 2020, pp. 10 767–10 777.

[25] J. Hartung, G. Knapp, B. K. Sinha, and B. K. Sinha, *Statistical meta-analysis with applications*. Wiley Online Library, 2008, vol. 6.

[26] A. Loquercio, M. Segu, and D. Scaramuzza, "A general framework for uncertainty estimation in deep learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3153–3160, 2020.

[27] B. J. Frey and G. E. Hinton, "Variational learning in nonlinear gaussian belief networks," *Neural Computation*, vol. 11, no. 1, pp. 193–213, 1999.

[28] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.

[29] A. Camuto, "Understanding gaussian noise injections in neural networks," Ph.D. dissertation, University of Oxford, 2021.

[30] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[31] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1321–1330.

[32] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," *arXiv preprint arXiv:1706.02690*, 2017.

[33] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical programming*, vol. 45, no. 1, pp. 503–528, 1989.

[34] ccs.neu.edu, "Numerical optimization: Understanding l-bfgs," 2017.

[35] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[36] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "Emnist: Extending mnist to handwritten letters," in *2017 international joint conference on neural networks (IJCNN)*. IEEE, 2017, pp. 2921–2926.

[37] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[38] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[39] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 215–223.

[40] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[41] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[42] J. Deng, W. Dong, R. Socher, L. jia Li, K. Li, and L. Fei-fei, "Imagenet: A large-scale hierarchical image database," in *In CVPR*. CVPR, 2009.

[43] A. Barbu, D. Mayo, J. Alverio, W. Luo, C. Wang, D. Gutfreund, J. Tenenbaum, and B. Katz, "Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models," *Advances in neural information processing systems*, vol. 32, 2019.

[44] Y. Feng, Q. Shi, X. Gao, J. Wan, C. Fang, and Z. Chen, "Deepgini: prioritizing massive tests to enhance the robustness of deep neural networks," in *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2020, pp. 177–188.

[45] J. Kim, R. Feldt, and S. Yoo, "Guiding deep learning system testing using surprise adequacy," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019, pp. 1039–1049.

[46] S. Yan, Y. Xiong, K. Kundu, S. Yang, S. Deng, M. Wang, W. Xia, and S. Soatto, "Positive-congruent training: Towards regression-free model updates," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 14 299–14 308.

[47] Z. Allen-Zhu and Y. Li, "Towards understanding ensemble, knowledge distillation and self-distillation in deep learning," *arXiv preprint arXiv:2012.09816*, 2020.

[48] A. Shi, M. Hadzi-Tanovic, L. Zhang, D. Marinov, and O. Legunsen, "Reflection-aware static regression test selection," *Proc. ACM Program. Lang.*, vol. 3, no. OOPSLA, oct 2019. [Online]. Available: https://doi.org/10.1145/3360613

[49] Q. Yi, Z. Yang, J. Liu, C. Zhao, and C. Wang, "A synergistic analysis method for explaining failed regression tests," in *Proceedings of the 37th International Conference on Software Engineering - Volume 1*, ser. ICSE '15. IEEE Press, 2015, pp. 257–267.

[50] K. Yu, M. Lin, J. Chen, and X. Zhang, "Practical isolation of failure-inducing changes for debugging regression faults," in *Proceedings of the 27th IEEE/ACM International Conference on Automated Software*

Engineering, ser. ASE 2012. New York, NY, USA: Association for Computing Machinery, 2012, pp. 20–29. [Online]. Available: https://doi.org/10.1145/2351676.2351681

[51] F. Pastore, L. Mariani, and A. Goffi, "Radar: A tool for debugging regression problems in c/c++ software," in *Proceedings of the 2013 International Conference on Software Engineering*, ser. ICSE '13. IEEE Press, 2013, pp. 1335–1338.

[52] Z. Li, X. Ma, C. Xu, C. Cao, J. Xu, and J. Lü, "Boosting operational dnn testing efficiency through conditioning," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2019, pp. 499–509.

[53] J. M. Zhang, M. Harman, L. Ma, and Y. Liu, "Machine learning testing: Survey, landscapes and horizons," *IEEE Transactions on Software Engineering*, 2020.

[54] X. Sun, T. Zhou, R. Wang, Y. Duan, L. Bo, and J. Chang, "Experience report: investigating bug fixes in machine learning frameworks/libraries," *Frontiers of Computer Science*, vol. 15, no. 6, pp. 1–16, 2021.

[55] Z. Li, X. Ma, C. Xu, and C. Cao, "Structural coverage criteria for neural networks could be misleading," in *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. IEEE, 2019, pp. 89–92.

[56] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Artificial intelligence safety and security*. Chapman and Hall/CRC, 2018, pp. 99–112.

[57] V. Buhrmester, D. Münch, and M. Arens, "Analysis of explainers of black box deep neural networks for computer vision: A survey," *Machine Learning and Knowledge Extraction*, vol. 3, no. 4, pp. 966–989, 2021.

[58] G. Giray, "A software engineering perspective on engineering machine learning systems: State of the art and challenges," *Journal of Systems and Software*, vol. 180, p. 111031, 2021.

[59] H. Wang, J. Xu, C. Xu, X. Ma, and J. Lu, "Dissector: Input validation for deep learning applications by crossing-layer dissection," in *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. IEEE, 2020, pp. 727–738.

[60] P. A. Flach, "Classifier calibration," in *Encyclopedia of machine learning and data mining*. Springer US, 2016.

[61] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek, "Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift," *Advances in neural information processing systems*, vol. 32, 2019.

[62] Z. Li, X. Ma, C. Xu, J. Xu, C. Cao, and J. Lü, "Operational calibration: Debugging confidence errors for dnns in the field," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, pp. 901–913.

[63] Y. Shen, Y. Xiong, W. Xia, and S. Soatto, "Towards backward-compatible representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6368–6377.

[64] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, "An empirical investigation of catastrophic forgetting in gradient-based neural networks," *arXiv preprint arXiv:1312.6211*, 2013.

[65] T. Sakai, "A generalized backward compatibility metric," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 1525–1535.

[66] J. Wing, "Embracing uncertainty," in *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, ser. SIGCSE '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 7. [Online]. Available: https://doi.org/10.1145/3017680.3025045

[67] S. M. Hezavehi, D. Weyns, P. Avgeriou, R. Calinescu, R. Mirandola, and D. Perez-Palacin, "Uncertainty in self-adaptive systems: A research community perspective," *ACM Trans. Auton. Adapt. Syst.*, vol. 15, no. 4, dec 2021. [Online]. Available: https://doi.org/10.1145/3487921

[68] N. Esfahani, E. Kouroshfar, and S. Malek, "Taming uncertainty in self-adaptive software," in *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*, ser. ESEC/FSE '11. New York, NY, USA: Association for Computing Machinery, 2011, pp. 234–244. [Online]. Available: https://doi.org/10.1145/2025113.2025147

[69] M. Li and I. K. Sethi, "Confidence-based active learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 8, pp. 1251–1261, 2006.

[70] M. Dredze and K. Crammer, "Active learning with confidence," in *Proceedings of ACL-08: HLT, Short Papers*, 2008, pp. 233–236.

[71] Y.-C. Hsu, Y. Shen, H. Jin, and Z. Kira, "Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 951–10 960.

[72] E. Hüllermeier and W. Waegeman, "Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods," *Machine Learning*, vol. 110, no. 3, pp. 457–506, 2021.

[73] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural computation*, vol. 3, no. 1, pp. 79–87, 1991.

[74] M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the em algorithm," *Neural computation*, vol. 6, no. 2, pp. 181–214, 1994.

[75] M. Woźniak, M. Grana, and E. Corchado, "A survey of multiple classifier systems as hybrid systems," *Information Fusion*, vol. 16, pp. 3–17, 2014.

[76] G. J. McLachlan, S. X. Lee, and S. I. Rathnayake, "Finite mixture models," *Annual review of statistics and its application*, vol. 6, pp. 355–378, 2019.

[77] S. Masoudnia and R. Ebrahimpour, "Mixture of experts: a literature survey," *Artificial Intelligence Review*, vol. 42, no. 2, pp. 275–293, 2014.