AutoML Decathlon: Diverse Tasks, Modern Methods, and Efficiency at Scale

Nicholas Roberts,[*1] **Samuel Guo,**[*2] **Cong Xu,**[*3] **Ameet Talwalkar,**[2,3]
Lead organizers

David Lander,[4] **Lvfang Tao,**[5] **Linhang Cai,**[5] **Shuaicheng Niu,**[5] **Jianyu Heng,**[5] **Hongyang Qin,**[5] **Minwen Deng,**[5] **Johannes Hog,**[6] **Alexander Pfefferle,**[6] **Sushil Ammanaghatta Shivakumar,**[6] **Arjun Krishnakumar,**[6] **Yubo Wang,**[6] **Rhea Sukthanker,**[6] **Frank Hutter,**[6] **Euxhen Hasanaj,**[2] **Tien-Dung Le,**[7]
Top-5 teams

Mikhail Khodak,[2] **Yuriy Nevmyvaka,**[8] **Kashif Rasul,**[8] **Frederic Sala,**[1] **Anderson Schneider,**[8] **Junhong Shen,**[2] **Evan Sparks**[3]
Organizing team

Editors: Marco Ciccone, Gustavo Stolovitzky, Jacob Albrecht

Abstract

The vision of Automated Machine Learning (AutoML) is to produce high performing ML pipelines that require very little human involvement or domain expertise to use. Competitions and benchmarks have been critical tools for accelerating progress in AutoML. However, much of the prior work on AutoML competitions has focused on well-studied domains in machine learning such as vision and language—these are domains which have benefited from several years of ML pipeline design by domain experts, which brings the usage of AutoML into question in the first place. Recently, AutoML for diverse tasks has emerged as an important research area that aims to bring AutoML to the domains where it can have the most impact: the long tail of ML tasks *beyond vision and language*. We present a retrospective report of the AutoML Decathlon—an AutoML for diverse tasks competition hosted at NeurIPS 2022. The AutoML Decathlon presented participants with a set of 10 machine learning tasks that are diverse along several axes: domain, input dimension, output dimension, output type, objective function, and scale. Participants were tasked with developing AutoML methods that performed well on a *separate* set of 10 hidden diverse test tasks within a certain time budget, so as to discourage overfitting to the initial set of tasks and to encourage efficiency. In this report, we outline the details of the competition, discuss the top-5 submissions, analyze the results, and compare top submissions to additional state-of-the-art baselines designed specifically for diverse tasks. We conclude that the combination of existing efficient AutoML techniques with modern advancements in ML such as large-scale transfer learning, modern architectures, and differentiable Neural Architecture Search (NAS) is a promising direction for AutoML for diverse tasks.

Keywords: AutoML, Diverse Tasks, Neural Architecture Search, Deep Learning

[1]University of Wisconsin-Madison, [2]Carnegie Mellon University, [3]Hewlett Packard Enterprise, [4]TrueFit.AI, [5]Tencent AI Lab, [6]University of Freiburg, [7]KBC Belgium, [8]Morgan Stanley
[*] Equal contribution

1. Introduction

Automated Machine Learning (AutoML) is an important field offering substantial possibilities in expanding the use of machine learning to the broader community. It reduces the need for human involvement in the process of building machine learning pipelines by automating design choices that would have otherwise required experience. The AutoML vision is to produce performant systems with virtually no need for human input or domain expertise. If fully achieved, this goal will have significant implications for both industrial applications (Jumper et al., 2021; Degrave et al., 2022), academic research (Real et al., 2020; Fawzi et al., 2022), and more.

Benchmarks are a crucial tool for measuring progress towards the AutoML vision. While successful as a yardstick for AutoML techniques in certain scenarios, much of the prior efforts in benchmarking have had a myopic focus on domains that have already received significant attention from the ML community. These include vision and language. AutoML techniques can lead to improvements on problems in these well-studied domains, but its larger promise lies in the long tail of less-studied ML tasks from diverse domains. For these under-studied tasks, dramatically less human effort has gone into designing tailor-made ML pipelines.

AutoML for diverse tasks is an emerging area aiming to take advantage of these opportunities. This promising line of work includes a number of recently-proposed methods (Shen et al., 2023, 2022; Roberts et al., 2021) and benchmarks (Tu et al., 2022; Roberts et al., 2022). To further accelerate progress in this area, we ran the AutoML Decathlon, an AutoML competition centered on diverse tasks. Participants were given a set of 10 machine learning tasks which were diverse along various axes: domain, input type/shape, output type/shape, learning objective, evaluation metric, and scale. To discourage overfitting to a specific domain or task type, a separate, unobserved set of 10 additional diverse tasks were used to evaluate the submitted methods. Ultimately, we received 24 submissions, with nine of them outperforming our baselines at the conclusion of the competition.

This report consists of a full breakdown of the design of the competition, including task details, evaluation, infrastructure, hackathons, an analysis of the results, a set of lessons learned, and implications for the future of AutoML. In particular, we highlight the following key takeaways from the AutoML Decathlon:

- top methods combined standard AutoML approaches with a wide variety of neural network architectures, large scale transfer learning, and simpler ML models,
- existing methods targeting AutoML for diverse tasks perform strongly,
- methods that performed strongly across task types outperformed specialized methods.

We conclude that a combination of modern ML methods—transfer learning, modern architectures, and differentiable NAS—with advanced Hyperparameter Optimization (HPO) and ensembling is a promising direction for the field of AutoML for diverse tasks.

2. Competition setup

The competition consisted of two phases: the development phase and the test phase. During the development phase, participants developed their methods for 10 public tasks with set

time and submission limits. The test phase consisted of offline evaluations of final submissions on 10 private test tasks. Unlike previous AutoML competitions (Guyon et al., 2019; Liu et al., 2020), participants in the AutoML Decathlon were encouraged to consider a wide range of approaches, from traditional hyperparameter optimization to modern techniques like NAS and large-scale transfer learning. The competition was challenging, but also offered a valuable opportunity for participants to gain experience in working with complex tasks and finding innovative solutions to difficult problems.

We hosted the competition on CodaLab (Pavao et al., 2022).[1] To ensure a smooth and timely execution of the competition, we made several modifications to the CodaLab Docker image, including virtual partitioning of a multi-GPU dual-socket server, prevention of test dataset label leaks, increase in shared memory size, fixation of CPU/GPU affinity, backup of the entire working directory for each submission for post-analysis, and various bug fixes.

Moreover, to accommodate the various deep learning frameworks used by participants, we provided a customized application Docker image.[2] This image included popular deep learning frameworks, such as TensorFlow (Abadi et al., 2015), PyTorch (Paszke et al., 2019), XGBoost (Chen and Guestrin, 2016), and scikit-learn (Pedregosa et al., 2011). This ensured that participants could use their preferred framework without any compatibility issues.

2.1. Tasks

The competition is based on a set of 20 tasks that have been carefully selected to cover a diverse range of practical applications across scientific, technological, and industrial domains. In the development phase, we released 10 tasks, with task metadata, as shown in Table 1(*a*). These tasks differ in their domains, which include spherically projected images, financial timeseries, audio, and the natural sciences. They also cover different problem types such as regression, single-label, and multi-label classification. These tasks contain several thousand to hundreds of thousands of observations, with sizes ranging from 23 MB to 36 GB, making them complex and challenging to work with.

The aim of selecting 10 tasks in the evaluation phase, shown in Table 1(*b*), was to have a diverse set of data that is representative of the development phase, while avoiding a situation where the leaderboard is completely different from the development phase. For example, to complement the Navier Stokes task (Li et al., 2021), we chose to use the shallow water PDE (Takamoto et al., 2022) as a test task, and to complement the Crypto development task, we included a high-frequency stock prediction test task (Ntakaris et al.). Moreover, these tasks use different evaluation metrics for each of the tasks, although the evaluation metrics that were given as part of the development tasks were representative of those of the test tasks. While competitors had access to the development tasks throughout the competition, the test tasks remained private for the duration of the competition and was used solely for evaluation purposes. This ensured that participants were not able to access the test data beforehand, making the competition fair for all. Finally, to enable the use of modern AutoML methods, we allocated a relatively large compute budget for each task in comparison to other AutoML competitions—small tasks are allocated up to 5 hours and large tasks are allowed up to 20 hours on a single V100 GPU.

1. https://codalab.lisn.upsaclay.fr/competitions/6325
2. https://hub.docker.com/r/automldec/decathlon

Table 1: Development and test task metadata. All listed metadata except for the domain was available to competitors during the competition.

(a) Development tasks and metadata.

Task	N	Type	Domain	In Dim.	Out Dim.	Size	Metric
Navier-Stokes (Li et al., 2021)	1200	Continuous	PDE	(20, 1, 64, 64)	(64, 64)	396 MB	L2 rel. error
Spherical (Cohen et al., 2018)	60000	Single-label	Image	(1, 3, 60, 60)	(100)	619 MB	0-1 Error
NinaPro (Atzori et al., 2012)	3956	Single-label	CompBio	(1, 16, 52, 1)	(18)	26 MB	0-1 Error
FSD50K (Fonseca et al., 2017)	51000	Multilabel	Audio	(1, 96, 101, 1)	(200)	36 GB	1 - MAP
Cosmic (Zhang and Bloom, 2020)	5250	Multilabel	Image	(1, 1, 128, 128)	(128, 128)	7.7 GB	1 - AUROC
ECG (Clifford et al., 2017)	330000	Single-label	CompBio	(1, 1, 1000, 1)	(4)	2.5 GB	1 - F1 Score
DeepSEA (et al., 2004)	250000	Multilabel	CompBio	(1, 4, 1000, 1)	(36)	896 MB	1 - AUROC
Nottingham (Bai et al., 2018)	1200	Multilabel	Audio	(1792, 88, 1, 1)	(88)	23 MB	NLL
Crypto*	1559	Continuous	Time-series	(3000, 13, 1, 1)	(600)	524 MB	L2 rel. error
EMBER (Anderson and Roth, 2018)	900000	Single-label	Tabular	(1, 2381, 1, 1)	(2)	7.2 GB	0-1 Error

*The Crypto dataset was generously provided by Morgan Stanley.

(b) Test tasks and metadata.

Task	N	Type	Domain	In Dim.	Out Dim.	Size	Metric
SphericalTinyImageNet (Le and Yang, 2015)	100000	Single-label	Image	(1, 3, 30, 30)	(200)	1.6 GB	0-1 Error
Satellite (Petitjean et al., 2012)	900000	Single-label	Time-series	(46, 1, 1, 1)	(24)	359 MB	1 - F1 Score
JSBChorales (Bai et al., 2018)	18104	Multilabel	Audio	(40, 88, 1, 1)	(88)	100 KB	NLL
HumanGait (Vajdi et al., 2019)	47276	Single-label	Time-series	(128, 3, 1, 1)	(88)	155 MB	1 - MAP
SpokenMNIST (Jackson et al., 2018)	2700	Single-label	Audio	(1, 4, 64, 64)	(10)	47 MB	0-1 Error
Stock (Ntakaris et al.)	7234	Continuous	Time-series	(1000, 40, 1, 1)	(200)	2.4 GB	L2 rel. error
MYO (Côté-Allard et al., 2019)	224406	Single-label	Time-series	(12, 8, 7, 1)	(7)	722 MB	0-1 Error
ShallowWater (Takamoto et al., 2022)	3600	Continuous	PDE	(10, 1, 64, 64)	(64, 64)	625 MB	L2 rel. error
Year (Bertin-Mahieux et al., 2011)	463715	Continuous	Audio	(1, 90, 1, 1)	(1)	181 MB	L2 rel. error
HumanProteinAtlas (Sullivan et al., 2018)	3107	Multilabel	CompBio	(1, 1, 128, 128)	(128, 128)	777 MB	1 - AUROC

2.2. Evaluation and scoring metrics

Each individual task had a metric, standardized such that a lower score indicated better performance. These are all common metrics—such as 0-1 error, L2 relative error, the negative log likelihood (NLL)—or simple transformations thereof, e.g. 1−F1. To rank each method's performance on all tasks, we developed a score based on performance profiles (Dolan and Moré, 2001), called the Area Under Performance profile curve (AUP). First, the performance profile curve parameterized by τ is defined in the formula below:

$$\rho_s(\tau) = \frac{1}{|P|}\left|p \in P : \frac{\text{LPM}_{p,s}}{\min_{s\in S}\text{LPM}_{p,s}} \leq \tau\right|,$$

where the performance metric of method s on task p as $\text{LPM}_{p,s}$. The AUP score of task s is computed by integrating this expression with respect to $\log(\tau)$. The upper bound, u, of the integral must be the smallest value such that $\rho_s(10^u) = 1\ \forall s$, or equivalently, this is the upper bound of the standard performance profile curve. Then the AUP score is given as

$$\text{AUP}_s = \int_0^u \rho_s(\tau)d\sigma\ ,\ \sigma = \log_{10}(\tau).$$

This metric measures how often a method is within a multiplicative factor of the best score on the leaderboard for each task. The formula uses individual task scores to create a performance profile curve, plotting the proportion of task instances in which a particular task method is within a factor τ of the optimal method's score against $\log_{10}(\tau)$, then computing the area under the curve.

This AUP metric offers certain advantages over other commonly-used aggregate metrics such as average rank. AUP is more holistic because it factors in the degree of difference in performance between methods, rather than just the relative order of performance. For example, if there exists a task on which many methods achieve strong but relatively similar scores, the 10th-ranked method may not be far off in performance to the 1st-ranked method. In this case, the AUP metric would correctly account for the fact that the former method is still very good, whereas it would be penalized heavily by the average rank metric. Throughout the competition, we maintained a leaderboard of competitors' AUP on the development tasks, and the final ranking was determined by the AUP on the test tasks.

2.3. Compute resources

Hewlett Packard Enterprise (HPE) allocated three Apollo 6500 GPU servers, each equipped with eight NVIDIA V100s, and $50,000 in cloud credit resources. These servers were the compute backend that connected to the worker queue on CodaLab. When demand peaked approaching the end date of the competition, HPE also allocated up to 18 Google Cloud instances with the same V100 GPUs.

To ensure fairness, the CodaLab Docker images were modified to provide a virtual partition of the Apollo server that matches the same setup of the cloud instance. Each submission ran on a virtual machine equipped with an NVIDIA Tesla V100 GPU, eight vCPUs, and 30 GB of memory. This provided a powerful computing environment for participants to develop their methods. Additionally, one VM is dedicated entirely to the job of one participant during its execution, with minimum interference from other participants.

2.4. Development phase hackathons

With the goals of introducing university students at all levels to the problem of AutoML for diverse tasks and giving them initial exposure to the competition, we hosted three separate hackathons at Carnegie Mellon University, the University of Wisconsin-Madison, and the AutoML Fall School in Freiburg, Germany.[3] These hackathons were in-person and took place during consecutive weeks in October 2022.

Each hackathon was run separately over the course of roughly 36 hours. An introductory presentation to the hackathon, the overarching competition, and the topic and motivation of AutoML was delivered before participants were made free to begin their development. Due to the brief time frames of the hackathons, their focus was to offer initial engagement with the competition and AutoML as whole, and ideally garner continuing interest until the competition's conclusion from some teams. As an additional incentive for engagement, we offered Amazon gift cards as prizes to teams that submitted a method to the official competition that surpassed certain baselines, or gave a presentation on the methods they implemented or researched. We provided hackathon participants with a self-contained Jupyter notebook that simulated and walked through the real competition pipeline and automatically downloaded a subset of the development tasks. Additionally, as many students were unfamiliar with AutoML, we provided a list of basic suggested approaches. The hackathons resulted in several submissions to the competition that explored a wide range of approaches. Some teams from the hackathons became serious competitors after improving their methods for the remainder of the competition.

2.5. Test phase

On the last day of the development phase, we encountered a platform bug which prevented submissions from being automatically moved to the test phase. To account for this, we allotted extra time for participants to make their final submissions via email, and we proceeded with the test phase manually. Consequently, we worked with teams to migrate their code to our manual test phase runs. We provide full details about how this process was conducted in Appendix D.

Results post-processing During the test phase, two teams—Team TrueFit and Team TEG—both achieved perfect test performance on the HumanGait task, which resulted in scores of 0.0. Several other entries achieved similarly high scores, differing only by 1-2 misclassified examples, and were affected by the magnitudes of their confidence scores. However, perfect scores of 0.0 are not supported by performance profiles because these scores appear as a divisor—therefore, we could not directly compute the final AUP scores using these entries. We considered several options for fixing this issue: adding a small epsilon to the perfect scores $\epsilon_{\text{perfect}} \in (0, \text{next\_best}]$, adding a small epsilon to all of the scores $\epsilon_{\text{all}} > 0$, removing the task entirely, or setting a nonzero best possible score for the task $\epsilon_{\text{min\_score}} > 0$. Here, next_best refers to the next best nonzero score. We found that under reasonable choices of $\epsilon_{\text{perfect}}$, ϵ_{all}, or $\epsilon_{\text{min\_score}}$, all of the solutions resulted in the same winner. Ultimately, we set a minimum possible score for the task, and used a value of $\epsilon_{\text{min\_score}} = 1\text{e} - 7 < \text{next\_best}$.

3. https://sites.google.com/view/automl-fall-school-2022/home

3. Results and analysis

In the test phase, nine submissions ultimately outperformed the provided baselines. In this section, we provide details about the top-5 highest performing methods on the test tasks.

Table 2: Development and test task leaderboards. These have been trimmed to show only the top-11 methods to include all methods that achieved the best score on any of the tasks—the complete leaderboards are given in Appendix A. Methods are ranked according to their AUP scores (not shown). Lower is better for all tasks.

(a) Development phase leaderboard.

Team	Navier Stokes	Spherical	NinaPro	Cosmic	ECG	Deep-SEA	Nott-ingham	Crypto	Ember	FSD50K
euxhenh	0.1659	0.7332	0.1290	**0.0243**	0.3792	0.3767	0.0241	**0.0020**	0.0462	0.8560
Freiburg-AutoML-Lab	**0.0858**	0.7928	0.1290	0.0477	0.8910	**0.3208**	**0.0148**	0.9990	0.0392	0.7211
TEG-AutoML	0.1892	0.9771	0.1426	0.4983	**0.3205**	0.4992	0.1168	0.0039	0.0685	0.7702
KaiWu	0.1892	0.9771	0.1426	0.4983	**0.3205**	0.4992	0.1168	0.0039	0.0685	0.7702
Team 42	0.9999	0.9016	0.1244	0.4607	0.4685	0.4277	0.0486	0.0270	**0.0247**	**0.6150**
TrueFit	0.6648	0.8809	**0.1168**	0.4983	0.6730	0.3873	0.0177	0.0638	0.0257	0.9738
Minions	0.2925	0.9671	0.2367	0.4716	0.4864	0.4989	0.2607	0.0058	0.0547	0.9598
XGBBaseline	0.2848	0.9660	0.2109	0.4689	0.5400	0.4992	0.2607	0.0055	0.0686	0.9636
dragon_bra	0.9999	0.9443	0.1973	0.4983	0.5576	0.4990	0.2674	0.0021	0.0775	0.9587
miracle-flex	0.2538	0.8947	0.2367	0.4983	0.8910	0.4992	0.0548	0.0111	0.4043	0.9738
automl	0.9999	**0.7316**	0.1684	0.4540	0.8910	0.4992	0.0506	0.0074	0.4043	0.9738

(b) Test phase leaderboard.

Team	Shallow Water	Satellite	Spherical Tiny ImageNet	JSB Chorales	Human Gait	Spoken MNIST	Stock	MYO	Year	HPA
TrueFit	0.0697	0.3764	**0.9798**	0.1228	**1e-7***	0.0067	0.0325	0.0279	0.0042	0.0219
TEG-AutoML	0.0474	0.3003	0.9903	0.1110	**1e-7***	0.0067	0.0822	0.1331	0.0033	**0.0190**
Freiburg-AutoML-Lab	**0.0004**	**0.2256**	0.9896	0.1017	2.281e-6	0.1200	0.0412	0.0400	0.0031	0.3883
euxhenh	0.0964	0.4059	0.9965	0.1273	1.171e-5	0.0100	**0.0208**	0.0226	0.0034	0.0302
tak	0.0408	0.2423	0.9895	0.4735	1.087e-7	0.0333	0.0490	0.3742	0.0033	0.2198
automl	0.0461	0.2378	0.9950	0.5013	9.417e-7	**0.0033**	0.0519	0.3742	0.0034	0.2227
42	0.0343	0.2460	0.9836	0.5160	7.244e-7	0.0100	0.0660	0.3742	0.0033	0.2091
Paris-Saclay	0.0119	0.2334	0.9949	**0.0973**	0.9192	0.0433	0.0620	**0.0082**	**0.0030**	0.1793
dragon_bra	0.0006	0.3330	0.9945	3.8154	0.0068	0.1167	0.0597	0.0461	0.0032	0.6127
XGBBaseline	**0.0004**	0.6072	0.9832	4.1342	0.2557	0.1333	0.0832	0.0362	0.0033	0.3871
miracle-flex	0.0293	0.5721	0.9995	0.1134	0.0010	0.4333	0.0570	0.3742	0.0710	0.2059

*Post-processed to avoid scores of 0.0, which are not supported by AUP.

3.1. Top-5 approaches

Team TrueFit Team TrueFit finished in first place. They focused on architectural range, parameter tuning, and efficient experiments. They began with a transformer with time-based tokenization for sequence and signal data, and included a CNN for image classification, a U-Net for segmentation, and LightGBM for tabular data. They sampled hyperparameters from wide yet reasonable distributions, with random then evolutionary sampling during training. All models were tuned over 6-15 cross-validation runs, followed by full retraining. They note that including an MLP, time-series module, and similar techniques would further extend their winning solution. Further discussion and source code is available on GitHub.[4]

Team TEG-AutoML The second place team, Team TEG-AutoML, designed a solution based on DASH (Shen et al., 2022). This solution involved tailoring various templates to potential downstream tasks to guide a combination of differentiable NAS and HPO.

4. https://github.com/truefit-ai/auto-ml

Their NAS approach used a wide ResNet (Zagoruyko and Komodakis, 2016) backbone with attention modules (Hu et al., 2018) and self-distillation heads could transform into a specialized network for the given task. Normalization was applied to the inputs in order to improve optimization. Their solution also incorporated a trial scheduling mechanism, allowing for multiple runs of their full AutoML pipeline within the time budget, attempting several random seeds and matched templates. They included system-level optimizations such as automatic mixed precision, pin-memory, and persistent workers to speed up the data loading and training process. Their solution achieved a strong second-place ranking, demonstrating the versatility of morphism-based AutoML for the problem of diverse tasks.

Team Freiburg-AutoML-Lab The third place team, Team Freiburg-AutoML-Lab, created a taxonomy that determines the task type based on the input shape and trains multiple models that work well for this task. Afterward, they created ensembles for all possible model combinations and they selected the best-performing ensemble. Their handcrafted decision tree maps datasets to a subset of the model portfolio, which consists primarily of neural networks. Furthermore, their solution involved the use of pre-trained models, early stopping, and an "LR range test" (Smith, 2017), to find learning rates for the neural networks. This solution could be extended by using Meta-Learning or applying HPO. The implementation of their approach is available on GitHub.[5]

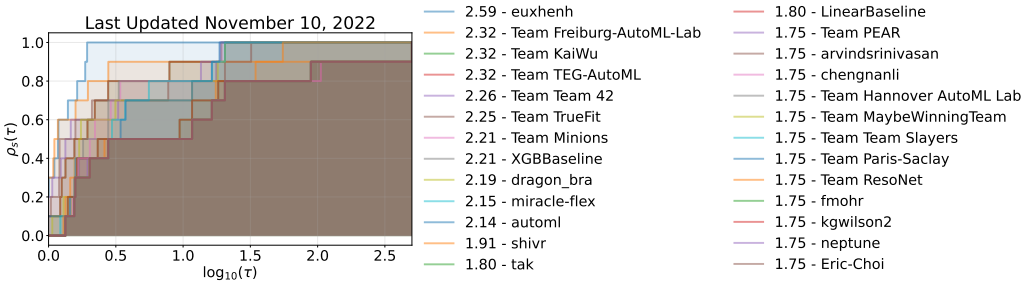
Team euxhenh Team euxhenh finished in fourth place with a solution that aimed to pick the right model family for a given task type. This was determined a hand-designed decision tree by considering the input dimensions of the task. Apart from tasks whose input consist of 1D feature vectors (which use XGBoost), all other task types were assigned deep learning-based methods. These varied from gated recurrent unit (GRU) networks for time series data to wide ResNets for channeled inputs, with special consideration given to 2D time-series and tasks with 2D outputs. The solution used simple model architectures (some containing only two layers) and it used fixed hyperparameters. This could be further improved by incorporating HPO or NAS techniques. Overall, this simple approach managed to beat the baselines by a large margin. The implementation is available on GitHub.[6]

Team tak Team tak finished in fifth place with an ensemble-based solution containing 2 main components. The first component included an HPO phase. Inspired by the H2O AutoML approach,[7] for each algorithm they executed a grid search to find the best hyperparameters for each algorithm they consider. The second component was a weighted average of different algorithms with the selected hyperparameters. They observed that no single algorithm worked best in all scenarios. They conclude that they needed to execute enough experiments to choose the best algorithms, hyperparameters, and ensemble weights.

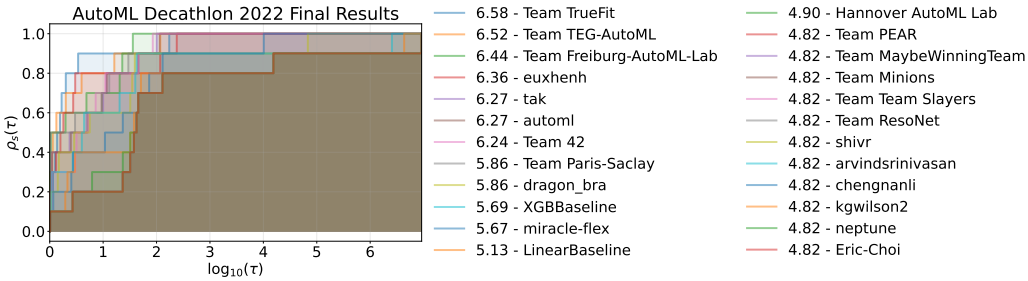
Discussion of top methods The top-5 teams combined existing HPO and/or portfolio selection methods with a wide variety of modern, specialized neural network architectures, pretrained models for transfer learning, and NAS techniques in their solutions. Team TrueFit used transformers, CNNs, and a U-Net architecture. Team TEG-AutoML used a variant of the wide ResNet architecture with attention modules in conjunction with DASH, while

5. https://github.com/automl/AutoML-Lab_Decathlon
6. https://github.com/euxhenh/automl-decathlon-2022-eh
7. https://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html

Team Freiburg-AutoML-Lab uses pretrained models for transfer learning with ensemble selection. Team euxhenh used GRU, ConvGRU, and wide ResNet architectures, and Team tak used an ensemble method inspired by H2O, an existing AutoML package. Four of the five top teams used some form of HPO in their solutions, and all of the teams included simpler ML models as fallback methods. *We conclude that a key feature of the top solutions was the combination of modern machine learning methods—modern architectures, transfer learning, and differentiable NAS—with advanced HPO or ensembling.*



(*a*) Results on dev tasks, as of original competition deadline.



(*b*) Final results on the test tasks.

Figure 1: Final AUP plots. The ordering of the legend from top-left to bottom-right indicates the ranking according to the AUP scores (where higher is better).

3.2. Comparison to baselines

We included the source code for two baselines at the beginning of the competition: a linear model and a stronger XGBoost (Chen and Guestrin, 2016) baseline. At the conclusion of the competition, we compared the final leaderboard results to two additional baselines: DASH (Shen et al., 2022) and AutoGluon (Erickson et al., 2020), which are both targeted specifically at the diverse tasks problem.

Linear baseline The linear baseline consisted of a single fully-connected layer with appropriate input and output dimensions.

XGBoost baseline The XGBoost baseline was implemented using the XGBoost package. XGBoost uses gradient boosted tree models for classification and regression tasks, and it supports multiple outputs.

DASH Baseline We evaluated DASH on the test tasks using 1D, 2D, and 3D wide ResNet (Zagoruyko and Komodakis, 2016) backbones. DASH uses a search space of several convolutional kernel sizes and dilation rates and an efficient differentiable search method.

AutoGluon Baseline Finally, we evaluated an AutoGluon baseline. Because AutoGluon's Tabular Predictors are designed for single-label tasks, this baseline falls back to XGBoost for multi-output tasks.

Discussion of baselines We ran these baselines with minimal tuning. Of these methods, DASH and AutoGluon achieved competitive performance—ranking in second and third place, respectively. We provide a more in-depth discussion of baseline performance in Appendix B. *We conclude that these two methods, both of which were designed specifically for the diverse tasks problem, are strong baselines that should be considered in future work on AutoML for diverse tasks.*

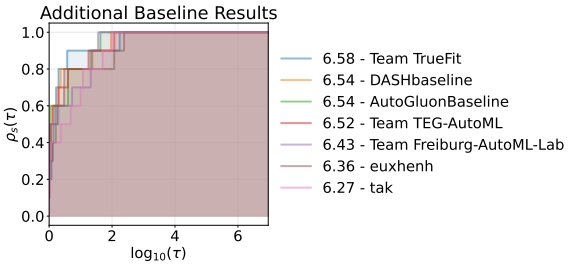
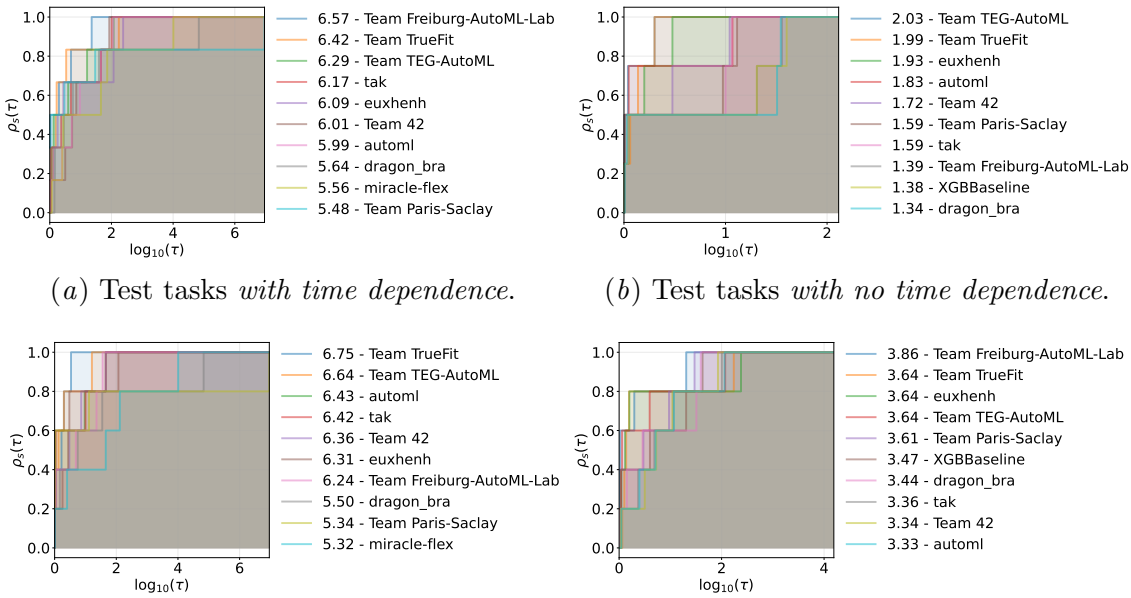


Figure 2: AUP plot comparing additional baselines to the top-5 submissions. DASH and AutoGluon achieve competitive performance.

3.3. Results by task type

We compare the top-10 rankings of submissions among different subsets of the test tasks. In particular, we compare submissions among the following categories of tasks: time-dependent, non-time-dependent, single-label classification, and multilabel or continuous test tasks. We chose these categories to compare across different input (Figures 3(a), 3(b)) and output characteristics (Figures 3(c), 3(d)) of the tasks.

The winning submission, Team TrueFit, performs the best on single-label classification tasks (Figure 3(c)), and is ranked second in all other categories. Team TEG-AutoML achieves the highest performance on tasks with no time dependence (Figure 3(b)), and is in the top-5 for all other subsets (Figures 3(a), 3(c), 3(d)). Team Freiburg-AutoML-Lab is first for time-dependent and multilabel/continuous tasks (Figures 3(a), 3(d)), but is less performant on single-label classification tasks and tasks with no time dependence (Figures 3(b), 3(c)). *We conclude that some submissions were more specialized for tasks with certain input or output characteristics, while submissions that performed strongly across all task types ultimately performed best overall.*



(*a*) Test tasks *with time dependence.* (*b*) Test tasks *with no time dependence.*

(*c*) *Single-label classification* test tasks. (*d*) *Mutlilabel or continuous* test tasks.

Figure 3: Top-10 rankings across different types of test tasks. Top overall submissions perform well on each subset, although the rankings differ across subsets.

4. Conclusions and future directions

We ran the AutoML Decathlon 2022 competition to stoke interest in AutoML for diverse tasks—an emerging and promising research area within AutoML. We received a total of 24 submissions during the development phase, with nine of them outperforming our two baselines in the test phase of the competition. We also ran a series of hackathons to prepare participants for the competition and to further promote the AutoML for diverse tasks problem. In our post-hoc analysis, we found that the top submission even outperformed existing state-of-the-art AutoML methods targeted at diverse tasks. *We conclude that combining advanced HPO with modern ML methods—including transfer learning, modern architectures, and NAS—is a promising direction for the field of AutoML for diverse tasks.*

In future work, we hope to run a follow-up competition that explores diverse tasks beyond standard supervised learning in diverse domains—e.g., limited access to labeled data. More generally in the longer term, we hope to continue to promote and explore this emerging research area via competitions, blogs, benchmarks, and new methods.

Authors' Note The first three authors contributed equally. Co-first authors may prioritize their names when referencing this work.

Acknowledgments

Funding for cloud compute resources was generously provided by Hewlett Packard Enterprise and Morgan Stanley. Hewlett Packard Enterprise also provided funding for the competition and hackathon prizes. Morgan Stanley helped prepare datasets in the competition phases. The submission portal and development and test phases were hosted on the CodaLab Competitions platform. We would also like to thank Isabelle Guyon and Adrien Pavao for their feedback and timely assistance with CodaLab and feedback on the competition. These contributions were essential to the formation and success of the competition.

This work was supported in part by the National Science Foundation grants IIS1705121, IIS1838017, IIS2046613, IIS2112471, and funding from Meta, Morgan Stanley, Amazon, and Google. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of any of these funding agencies.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

Hyrum S. Anderson and Phil Roth. Ember: An open dataset for training static pe malware machine learning models, 2018. URL https://arxiv.org/abs/1804.04637.

Manfredo Atzori, Arjan Gijsberts, Simone Heynen, Anne-Gabrielle Mittaz Hager, Olivier Deriaz, Patrick van der Smagt, Claudio Castellini, Barbara Caputo, and Henning Müller. Building the ninapro database: A resource for the biorobotics community. In *2012 4th IEEE RAS EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pages 1258–1265, 2012. doi: 10.1109/BioRob.2012.6290287.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv, 2018.

Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

Tianqi Chen and Carlos Guestrin. XGBoost. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, aug 2016. doi: 10.1145/2939672.2939785. URL https://doi.org/10.1145%2F2939672.2939785.

Gari D Clifford, Chengyu Liu, Benjamin Moody, Li-wei H. Lehman, Ikaro Silva, Qiao Li, A E Johnson, and Roger G. Mark. Af classification from a short single lead ecg recording: The physionet/computing in cardiology challenge 2017. In *2017 Computing in Cardiology (CinC)*, pages 1–4, 2017. doi: 10.22489/CinC.2017.065-469.

Taco S. Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical cnns. In *International Conference on Learning Representations*, 2018.

Ulysse Côté-Allard, Cheikh Latyr Fall, Alexandre Drouin, Alexandre Campeau-Lecours, Clément Gosselin, Kyrre Glette, François Laviolette, and Benoit Gosselin. Deep learning for electromyographic hand gesture signal classification using transfer learning. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 27(4):760–771, 2019. doi: 10.1109/TNSRE.2019.2896269.

Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de las Casas, Craig Donner, Leslie Fritz, Cristian Galperti, Andrea Huber, James Keeling, Maria Tsimpoukelli, Jackie Kay, Antoine Merle, Jean-Marc Moret, Seb Noury, Federico Pesamosca, David Pfau, Olivier Sauter, Cristian Sommariva, Stefano Coda, Basil Duval, Ambrogio Fasoli, Pushmeet Kohli, Koray Kavukcuoglu, Demis Hassabis, and Martin Riedmiller. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022. doi: 10.1038/s41586-021-04301-9. URL https://doi.org/10.1038/s41586-021-04301-9.

Elizabeth D. Dolan and Jorge J. Moré. Benchmarking optimization software with performance profiles. 2001. doi: 10.48550/ARXIV.CS/0102001. URL https://arxiv.org/abs/cs/0102001.

Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. Autogluon-tabular: Robust and accurate automl for structured data, 2020. URL https://arxiv.org/abs/2003.06505.

ENCODE Project Consortium et al. 306(5696):636–640, 2004.

Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J. R. Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, David Silver, Demis Hassabis, and Pushmeet Kohli. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610 (7930):47–53, 2022. doi: 10.1038/s41586-022-05172-4. URL https://doi.org/10.1038/s41586-022-05172-4.

Eduardo Fonseca, Jordi Pons, Xavier Favory, Frederic Font, Dmitry Bogdanov, Andres Ferraro, Sergio Oramas, Alastair Porter, and Xavier Serra. Freesound datasets: A platform for the creation of open audio datasets. In Sally Jo Cunningham, Zhiyao Duan, Xiao Hu, and Douglas Turnbull, editors, *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*, pages 486–493, 2017. URL https://ismir2017.smcnus.org/wp-content/uploads/2017/10/161_Paper.pdf.

Isabelle Guyon, Lisheng Sun-Hosoya, Marc Boullé, Hugo Jair Escalante, Sergio Escalera, Zhengying Liu, Damir Jajetic, Bisakha Ray, Mehreen Saeed, Michèle Sebag, Alexander Statnikov, Wei-Wei Tu, and Evelyne Viegas. *Analysis of the AutoML Challenge Series 2015–2018*, pages 177–219. Springer International Publishing, Cham, 2019. ISBN 978-3-030-05318-5. doi: 10.1007/978-3-030-05318-5_10. URL https://doi.org/10.1007/978-3-030-05318-5_10.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

Zohar Jackson, César Souza, Jason Flaks, Yuxin Pan, Hereman Nicolas, and Adhish Thite. Jakobovski/free-spoken-digit-dataset: v1.0.8, August 2018. URL https://doi.org/10.5281/zenodo.1342401.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873): 583–589, 2021. doi: 10.1038/s41586-021-03819-2. URL https://doi.org/10.1038/s41586-021-03819-2.

Ya Le and Xuan S. Yang. Tiny imagenet visual recognition challenge. 2015.

Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=c8P9NQVtmnO.

Zhengying Liu, Adrien Pavao, Zhen Xu, Sergio Escalera, Fabio Ferreira, Isabelle Guyon, Sirui Hong, Frank Hutter, Rongrong Ji, Julio C. S. Jacques Junior, Ge Li, Marius Lindauer, Zhipeng Luo, Meysam Madadi, Thomas Nierhoff, Kangning Niu, Chunguang Pan, Danny Stoll, Sebastien Treguer, Jin Wang, Peng Wang, Chenglin Wu, Youcheng Xiong, Arbër Zela, and Yang Zhang. Winning solutions and post-challenge analyses of the ChaLearn AutoDL challenge 2019. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 17, 2020.

Adamantios Ntakaris, Martin Magris, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Benchmark dataset for mid-price forecasting of limit order book data with machine learning methods. http://urn.fi/urn:nbn:fi:csc-kata20170601153214969115. N/A.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy,

Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Adrien Pavao, Isabelle Guyon, Anne-Catherine Letournel, Xavier Baró, Hugo Escalante, Sergio Escalera, Tyler Thomas, and Zhen Xu. Codalab competitions: An open source platform to organize scientific challenges. *Technical report*, 2022. URL https://hal.inria.fr/hal-03629462v1.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

François Petitjean, Jordi Inglada, and Pierre Gancarski. Satellite image time series analysis under time warping. *IEEE Transactions on Geoscience and Remote Sensing*, 50(8):3081–3095, 2012. doi: 10.1109/TGRS.2011.2179050.

Esteban Real, Chen Liang, David R. So, and Quoc V. Le. Automl-zero: Evolving machine learning algorithms from scratch. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Nicholas Roberts, Mikhail Khodak, Tri Dao, Liam Li, Christopher Ré, and Ameet Talwalkar. Rethinking neural operations for diverse tasks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 15855–15869. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/84fdbc3ac902561c00871c9b0c226756-Paper.pdf.

Nicholas Roberts, Xintong Li, Tzu-Heng Huang, Dyah Adila, Spencer Schoenberg, Cheng-Yu Liu, Lauren Pick, Haotian Ma, Aws Albarghouthi, and Frederic Sala. AutoWS-bench-101: Benchmarking automated weak supervision with 100 labels. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL https://openreview.net/forum?id=nQZHEunntbJ.

Junhong Shen, Mikhail Khodak, and Ameet Talwalkar. Efficient architecture search for diverse tasks. In *Advances in Neural Information Processing Systems*, 2022.

Junhong Shen, Liam Li, Lucio M. Dery, Corey Staten, Mikhail Khodak, Graham Neubig, and Ameet Talwalkar. Cross-modal fine-tuning: Align then refine, 2023. URL https://arxiv.org/abs/2302.05738.

Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 464–472. IEEE, 2017.

Devin P Sullivan, Casper F Winsnes, Lovisa Åkesson, Martin Hjelmare, Mikaela Wiking, Rutger Schutten, Linzi Campbell, Hjalti Leifsson, Scott Rhodes, Andie Nordgren, Kevin Smith, Bernard Revaz, Bergur Finnbogason, Attila Szantner, and Emma Lundberg. Deep

learning is combined with massive-scale citizen science to improve large-scale image classification. *Nature Biotechnology*, 36(9):820–828, 2018. doi: 10.1038/nbt.4225. URL https://doi.org/10.1038/nbt.4225.

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Dan MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. PDEBench: An Extensive Benchmark for Scientific Machine Learning. In *36th Conference on Neural Information Processing Systems (NeurIPS 2022) Track on Datasets and Benchmarks*, 2022. URL https://arxiv.org/abs/2210.07182.

Renbo Tu, Nicholas Roberts, Mikhail Khodak, Junhong Shen, Frederic Sala, and Ameet Talwalkar. NAS-bench-360: Benchmarking neural architecture search on diverse tasks. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL https://openreview.net/forum?id=xUXTbq6gWsB.

Amir Vajdi, Mohammad Reza Zaghian, Nazli Rafei Dehkordi, Elham Rastegari, Kian Maroofi, Saman Farahmand, Shaohua Jia, Marc Pomplun, Nurit Haspel, and Akram Bayat. Human gait database for normal walk collected by smartphone accelerometer, 2019. URL https://arxiv.org/abs/1905.03109.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference 2016*. British Machine Vision Association, 2016.

Keming Zhang and Joshua S. Bloom. deepCR: Cosmic ray rejection with deep learning. *The Astrophysical Journal*, 889(1):24, jan 2020. doi: 10.3847/1538-4357/ab3fa6.

Appendix A. Development and Test Leaderboards

In this section, we provide the complete development and test leaderboards. These are shown in Figure 3.

Table 3: Development and test task leaderboards.

(a) Development phase leaderboard.

Team	Navier Stokes	Spherical	NinaPro	Cosmic	ECG	Deep-SEA	Nott-ingham	Crypto	Ember	FSD50K
euxhenh	0.1659	0.7332	0.1290	**0.0243**	0.3792	0.3767	0.0241	**0.0020**	0.0462	0.8560
Freiburg-AutoML-Lab	**0.0858**	0.7928	0.1290	0.0477	0.8910	**0.3208**	**0.0148**	0.9990	0.0392	0.7211
TEG-AutoML	0.1892	0.9771	0.1426	0.4983	**0.3205**	0.4992	0.1168	0.0039	0.0685	0.7702
KaiWu	0.1892	0.9771	0.1426	0.4983	**0.3205**	0.4992	0.1168	0.0039	0.0685	0.7702
Team 42	0.9999	0.9016	0.1244	0.4607	0.4685	0.4277	0.0486	0.0270	**0.0247**	**0.6150**
TrueFit	0.6648	0.8809	**0.1168**	0.4983	0.6730	0.3873	0.0177	0.0638	0.0257	0.9738
Minions	0.2925	0.9671	0.2367	0.4716	0.4864	0.4989	0.2607	0.0058	0.0547	0.9598
XGBBaseline	0.2848	0.9660	0.2109	0.4689	0.5400	0.4992	0.2607	0.0055	0.0686	0.9636
dragon_bra	0.9999	0.9443	0.1973	0.4983	0.5576	0.4990	0.2674	0.0021	0.0775	0.9587
miracle-flex	0.2538	0.8947	0.2367	0.4983	0.8910	0.4992	0.0548	0.0111	0.4043	0.9738
automl	0.9999	**0.7316**	0.1684	0.4540	0.8910	0.4992	0.0506	0.0074	0.4043	0.9738
shivr	0.9955	0.9760	0.2352	0.4258	0.8340	0.4138	0.8135	0.0687	0.4043	0.8906
tak	0.9999	0.9771	0.1897	0.4442	0.7430	0.4460	1.3154	0.9990	0.2326	0.9715
LinearBaseline	0.9999	0.9771	0.1897	0.4442	0.7430	0.4460	1.3154	0.9990	0.2326	0.9738
PEAR	0.9999	0.9771	0.2276	0.4983	0.8910	0.4992	1.3154	0.9990	0.4043	0.9738
arvindsrinivasan	0.9999	0.9771	0.1942	0.4983	0.8910	0.4992	1.5607	0.9990	0.4043	0.9738
chengnanli	0.9999	0.9771	0.1942	0.4983	0.8910	0.4992	1.5607	0.9990	0.4043	0.9738
Hannover AutoML Lab	0.9999	0.9771	0.2367	0.4983	0.8910	0.4992	1.3154	0.9990	0.4043	0.9738
Slayers	0.9999	0.9771	0.2367	0.4983	0.8910	0.4992	1.3154	0.9990	0.4043	0.9738
fmohr	0.9999	0.9771	0.2367	0.4983	0.8910	0.4992	1.3154	0.9990	0.4043	0.9738
Eric-Choi	0.9999	0.9771	0.2367	0.4983	0.8910	0.4992	1.3154	0.9990	0.4043	0.9738
neptune	0.9999	0.9771	0.2367	0.4983	0.8910	0.4992	1.3154	0.9990	0.4043	0.9738
ResoNet	0.9999	0.9771	0.2367	0.4983	0.8910	0.4992	1.3154	0.9990	0.4043	0.9738
MaybeWinningTeam	0.9999	0.9771	0.2367	0.4983	0.8910	0.4992	1.3154	0.9990	0.4043	0.9738
kgwilson2	0.9999	0.9771	0.2367	0.4983	0.8910	0.4992	1.3154	0.9990	0.4043	0.9738
Paris-Saclay	0.9999	0.9771	0.2367	0.4983	0.8910	0.4992	1.3154	0.9990	0.4043	0.9738

(b) Test phase leaderboard.

Team	Shallow Water	Satellite	Spherical Tiny ImageNet	JSB Chorales	Human Gait	Spoken MNIST	Stock	MYO	Year	HPA
TrueFit	0.0697	0.3764	**0.9798**	0.1228	**1e-7***	0.0067	0.0325	0.0279	0.0042	0.0219
TEG-AutoML	0.0474	0.3003	0.9903	0.1110	**1e-7***	0.0067	0.0822	0.1331	0.0033	**0.0190**
Freiburg-AutoML-Lab	**0.0004**	**0.2256**	0.9896	0.1017	2.281e-6	0.1200	0.0412	0.0400	0.0031	0.3883
euxhenh	0.0964	0.4059	0.9965	0.1273	1.171e-5	0.0100	**0.0208**	0.0226	0.0034	0.0302
tak	0.0408	0.2423	0.9895	0.4735	1.087e-7	0.0333	0.0490	0.3742	0.0033	0.2198
automl	0.0461	0.2378	0.9950	0.5013	9.417e-7	**0.0033**	0.0519	0.3742	0.0034	0.2227
42	0.0343	0.2460	0.9836	0.5160	7.244e-7	0.0100	0.0660	0.3742	0.0033	0.2091
Paris-Saclay	0.0119	0.2334	0.9949	**0.0973**	0.9192	0.0433	0.0620	**0.0082**	**0.0030**	0.1793
dragon_bra	0.0006	0.3330	0.9945	3.8154	0.0068	0.1167	0.0597	0.0461	0.0032	0.6127
XGBBaseline	**0.0004**	0.6072	0.9832	4.1342	0.2557	0.1333	0.0832	0.0362	0.0033	0.3871
miracle-flex	0.0293	0.5721	0.9995	0.1134	0.0010	0.4333	0.0570	0.3742	0.0710	0.2059
LinearBaseline	6.2550	0.4844	0.9927	0.2899	0.4331	0.1700	0.7873	0.3742	0.0059	0.6127
Hannover AutoML Lab	6.2550	0.6072	0.9943	4.1342	0.9192	0.4333	0.1292	0.3742	0.0710	0.6127
Eric-Choi	6.2550	0.6072	0.9995	4.1342	0.9192	0.4333	0.7873	0.3742	0.0710	0.6127
arvindsrinivasan	6.2550	0.6072	0.9995	4.1342	0.9192	0.4333	0.7873	0.3742	0.0710	0.6127
MaybeWinningTeam	6.2550	0.6072	0.9995	4.1342	0.9192	0.4333	0.7873	0.3742	0.0710	0.6127
kgwilson2	6.2550	0.6072	0.9995	4.1342	0.9192	0.4333	0.7873	0.3742	0.0710	0.6127
Minions	6.2550	0.6072	0.9995	4.1342	0.9192	0.4333	0.7873	0.3742	0.0710	0.6127
chengnanli	6.2550	0.6072	0.9995	4.1342	0.9192	0.4333	0.7873	0.3742	0.0710	0.6127
neptune	6.2550	0.6072	0.9995	4.1342	0.9192	0.4333	0.7873	0.3742	0.0710	0.6127
ResoNet	6.2550	0.6072	0.9995	4.1342	0.9192	0.4333	0.7873	0.3742	0.0710	0.6127
PEAR	6.2550	0.6072	0.9995	4.1342	0.9192	0.4333	0.7873	0.3742	0.0710	0.6127
shivr	6.2550	0.6072	0.9995	4.1342	0.9192	0.4333	0.7873	0.3742	0.0710	0.6127
Slayers	6.2550	0.6072	0.9995	4.1342	0.9192	0.4333	0.7873	0.3742	0.0710	0.6127

*Post-processed to avoid scores of 0.0, which are not supported by AUP.

Appendix B. Additional Baseline Performance

The DASH baseline maintains strong results both in overall performance as well as individual task performance, claiming the top method spot in a couple tasks. As mentioned, the AutoGluon baseline substitutes XGBoost for multioutput tasks: ShallowWater, JSB-Chorales, Stock, and HPA. The AutoGluon baseline owes it's excellent ShallowWater result

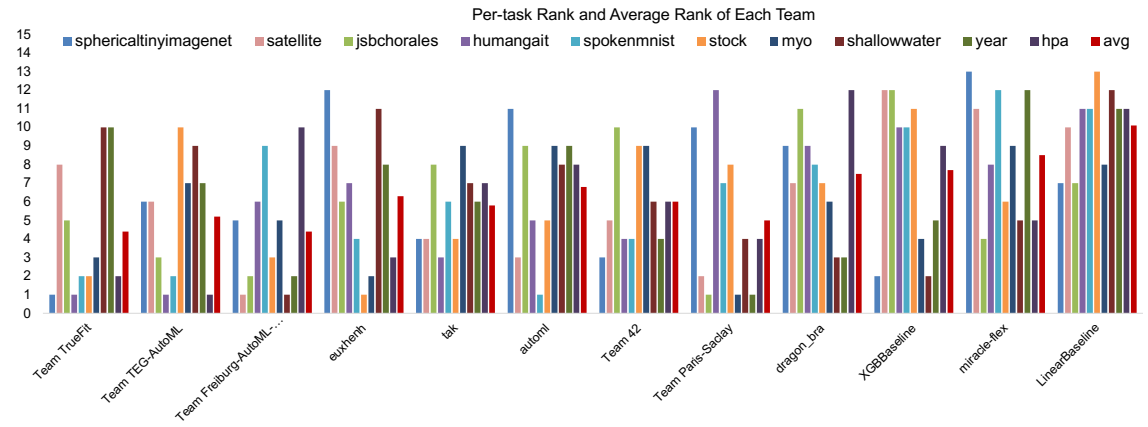
Table 4: Test leaderboard with additional post-competition baseline comparisons. **Bold red** numbers represent the best performance for a given task if it tied or outperformed the best performing method from the competition, and the previous-best results as of the end of the competition are shown in **bold**.

Team	Shallow Water	Satellite	Spherical Tiny ImageNet	JSB Chorales	Human Gait	Spoken MNIST	Stock	MYO	Year	HPA
TrueFit	0.0697	0.3764	**0.9798**	0.1228	**1e-7***	0.0067	0.0325	0.0279	0.0042	0.0219
DASHbaseline	0.0367	0.2689	0.9943	0.1198	**1e-7***	0.0800	0.0485	0.0146	0.0033	**0.0185**
AutoGluonBaseline	**0.0004**	0.2445	0.9949	4.1342	**1e-7***	0.0133	0.0832	**0.0074**	0.0033	0.3871
TEG-AutoML	0.0474	0.3003	0.9903	0.1110	**1e-7***	0.0067	0.0822	0.1331	0.0033	**0.0190**
Freiburg-AutoML-Lab	**0.0004**	**0.2256**	0.9896	0.1017	2.281e-6	0.1200	0.0412	0.0400	0.0031	0.3883
euxhenh	0.0964	0.4059	0.9965	0.1273	1.171e-5	0.0100	**0.0208**	0.0226	0.0034	0.0302
tak	0.0408	0.2423	0.9895	0.4735	1.087e-7	0.0333	0.0490	0.3742	0.0033	0.2198
automl	0.0461	0.2378	0.9950	0.5013	9.417e-7	**0.0033**	0.0519	0.3742	0.0034	0.2227
42	0.0343	0.2460	0.9836	0.5160	7.244e-7	0.0100	0.0660	0.3742	0.0033	0.2091
Paris-Saclay	0.0119	0.2334	0.9949	**0.0973**	0.9192	0.0433	0.0620	**0.0082**	**0.0030**	0.1793
dragon_bra	0.0006	0.3330	0.9945	3.8154	0.0068	0.1167	0.0597	0.0461	0.0032	0.6127
XGBBaseline	**0.0004**	0.6072	0.9832	4.1342	0.2557	0.1333	0.0832	0.0362	0.0033	0.3871
miracle-flex	0.0293	0.5721	0.9995	0.1134	0.0010	0.4333	0.0570	0.3742	0.0710	0.2059
LinearBaseline	6.2550	0.4844	0.9927	0.2899	0.4331	0.1700	0.7873	0.3742	0.0059	0.6127
Hannover AutoML Lab	6.2550	0.6072	0.9943	4.1342	0.9192	0.4333	0.1292	0.3742	0.0710	0.6127
Eric-Choi	6.2550	0.6072	0.9995	4.1342	0.9192	0.4333	0.7873	0.3742	0.0710	0.6127
arvindsrinivasan	6.2550	0.6072	0.9995	4.1342	0.9192	0.4333	0.7873	0.3742	0.0710	0.6127
MaybeWinningTeam	6.2550	0.6072	0.9995	4.1342	0.9192	0.4333	0.7873	0.3742	0.0710	0.6127
kgwilson2	6.2550	0.6072	0.9995	4.1342	0.9192	0.4333	0.7873	0.3742	0.0710	0.6127
Minions	6.2550	0.6072	0.9995	4.1342	0.9192	0.4333	0.7873	0.3742	0.0710	0.6127
chengnanli	6.2550	0.6072	0.9995	4.1342	0.9192	0.4333	0.7873	0.3742	0.0710	0.6127
neptune	6.2550	0.6072	0.9995	4.1342	0.9192	0.4333	0.7873	0.3742	0.0710	0.6127
ResoNet	6.2550	0.6072	0.9995	4.1342	0.9192	0.4333	0.7873	0.3742	0.0710	0.6127
PEAR	6.2550	0.6072	0.9995	4.1342	0.9192	0.4333	0.7873	0.3742	0.0710	0.6127
shivr	6.2550	0.6072	0.9995	4.1342	0.9192	0.4333	0.7873	0.3742	0.0710	0.6127
Slayers	6.2550	0.6072	0.9995	4.1342	0.9192	0.4333	0.7873	0.3742	0.0710	0.6127

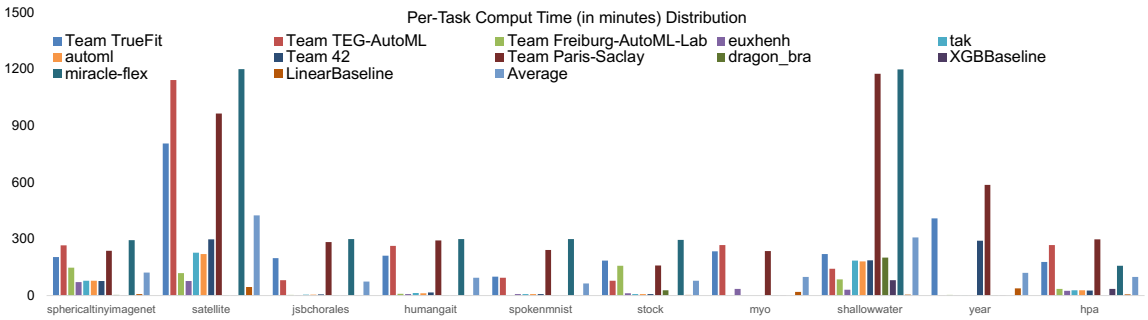
*Post-processed to avoid scores of 0.0, which are not supported by AUP.

to XGBoost, but on JSBChorales and HPA the performance is not as strong. On the remaining tasks, AutoGluon performs well.

Appendix C. Analysis of Per-task Rank and Compute Time



(*a*) Per-task ranks and average rank of each team that outperforms the linear baseline.



(*b*) Compute time distribution of each team on test tasks.

Figure 4: Per-task analyses.

Figure 4(*a*) shows the per-task rank and average ranks across tasks for each team that outperformed the linear baseline. Team TrueFit achieves top performance on SphericalTinyImagenet and Humangait with an average rank of 4.4. Team TEG-AutoML wins on HPA and Humangait with an average rank of 5.2. Finally, team Freiburg-AutoML-Lab wins on Satellite and ShallowWater with an average rank of 4.4. If we used average rank instead of AUP, both TrueFit and Freiburg-AutoML-Lab would be the winning teams. However, AUP was able to identify the significant performance gap between Freiburg-AutoML-Lab and optimal solutions on Humangait and HPA.

Figure 4(*b*) shows the compute time distribution of the top-10 teams for each task. Both inter-task and intra-task variations were observed. Some teams, such as Team Paris-Saclay and miracle-flex, tended to use all of their compute time budget for each task to exhaustively search for better solutions. The NAS-based solution from Team TEG-AutoML also takes

significantly longer time on average compared to most teams. The Satellite task was the most time-consuming, with an average compute time of 7 hours, followed by ShallowWater at 5 hours. The SpokenMINST task is the least time-consuming with an average compute time of 1 hour.

Appendix D. Test Phase Migration

Due to technical issues with CodaLab, participants were unable to make submissions on the final day of the competition. To address this, participants were asked to email their final submissions according to the following rules:

- no major changes to the method were allowed,
- no new files can be added,
- no new functions can be implemented,
- modifications or deletions of the existing code should be within 20 lines of code, and
- updating the submission was allowed only once.

Additionally, we ensured that no hints of the test tasks were given. The submissions were checked on the development tasks before being promoted to the test phase. This period also served as a debugging phase, during which minor code issues were resolved. Small edits of up to 20 lines of code were allowed for issues such as file path discrepancies, numerical errors, or environment problems. The organizers were able to fix all minor issues in the final submissions within a day. For example, a function name typo was found for Team TrueFit in their submitted "model.py," and a floating-point stability issue was solved for Team TEG in their submitted "dataloaders.py" within 3 lines of code change.