Meta-learning from Learning Curves Challenge: Lessons learned from the First Round and Design of the Second Round

Anonymous Authors[1]

Abstract

Meta-learning from learning curves is an important yet often neglected research area in the Machine Learning community. We introduce a series of Reinforcement Learning-based meta-learning challenges, in which an agent searches for the best suited algorithm for a given dataset, based on feedback of learning curves from the environment. The first round attracted participants both from academia and industry. This paper analyzes the results of the first round (accepted to the competition program of WCCI 2022), to draw insights into what makes a meta-learner successful at learning from learning curves. With the lessons learned from the first round and the feedback from the participants, we have designed the second round of our challenge with a new protocol and a new meta-dataset. The second round of our challenge is accepted at the AutoML-Conf 2022 and currently on-going.

1. Background and motivation

Meta-learning has been playing an increasingly important role in Automated Machine Learning. While it is a natural capability of living organisms, who constantly transfer acquired knowledge across tasks to quickly adapt to changing environments, artificial learning systems are still in their meta-learning "infancy". They are only capable, so far, to transfer knowledge between very similar tasks. At a time when society is pointing fingers at AI for being wasteful with computational resources, there is an urgent need for learning systems, which **recycle their knowledge**. To achieve that goal, some research areas have been widely studied, including few-shot learning (Wang et al., 2020), transfer learning (Zhuang et al., 2020), representation learning (Bengio et al., 2013), continual learning (Delange et al., 2021), life-long learning (Chen et al., 2018), and meta-learning (Vanschoren, 2018). However, **meta-learning from learning curves**, an essential sub-problem in meta-learning (Mohr & van Rijn, 2022), is still under-studied. This motivates the design of this new challenge series we are proposing.

Learning curves evaluate algorithm incremental performance improvement, as a function of training time, number of iterations, and/or number of examples. Our challenge design builds on top of previous challenges and work, which considered other aspects of the problem: meta-learning as a recommendation problem, but not from learning curves (Guyon et al., 2019; Liu et al., 2020) and few-shot learning (El Baz et al., 2021). Analysis of past challenges revealed that top-ranking methods often involve switching between algorithms during training, including "freeze-thaw" Bayesian techniques (Swersky et al., 2014). However, our previous protocols did not allow us to separately evaluate such methods, due to inter-dependencies between various heuristics. Furthermore, we want to study the potential benefit of **learned policies**, as opposed to applying hand-crafted black-box optimization methods.

Our challenge took inspiration from MetaREVEAL (Nguyen et al., 2021), ActivMetal (Sun-Hosoya et al., 2018), REVEAL (Sun-Hosoya, 2019), and Freeze-Thaw Bayesian Optimization (Swersky et al., 2014). A meta-learner needs to learn to solve two problems at the same time: **algorithm selection** and **budget allocation**. We are interested in meta-learning strategies that leverage information on **partially trained algorithms**, hence reducing the cost of training them to convergence. We offer pre-computed learning curves as a function of time, to facilitate benchmarking. Meta-learners must "pay" a cost emulating computational time (negative reward) for revealing their next values. Hence, meta-learners are expected to learn the **exploration-exploitation trade-offs** between continuing "training" an already tried good candidate algorithm and checking new candidate algorithms.

In this paper, we first recap the design of the first round of our challenge. Then, we present the first round results and compare the participants' solutions with our baselines. After the end of the first round, we have identified some limitations in our design and proposed a new design and a

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

new meta-dataset for the second round.

2. Design of the first round

In this section, we describe the design of the first round of our challenge, including: data, challenge protocol, and evaluation. More details can be found on our Codalab competition page [1].

2.1. Learning curve data

Despite the fact that learning curves are widely used, there were not many learning curve meta-datasets available in the Machine Learning community at the time of organizing this challenge. Taking advantage of 30 cross-domain datasets used in the AutoML challenge (Guyon et al., 2019), we computed *de novo* learning curves (both on the validation sets and the test sets) for 20 algorithms, by submitting them to the AutoML challenge, as post-challenge submissions. These algorithms are created from two base algorithms, Random Forest and Gradient Boosting, but with different values of several hyperparameters. We also provided meta-features of each dataset, such as type of learning task, evaluation metric, time budget, etc. In this first round, we considered a **learning curve as a function of time**. Each point on the learning curve corresponds to the algorithm performance at a certain time.

In addition to the aforementioned real-world meta-dataset, we synthetically generated 4000 learning curves for the participants to practice and get familiar with our starting kit. The points on each synthetic learning curve are sampled from a parameterized sigmoid function with its hyperparameters generated from matrix factorizations. This allows us to create some hidden relationships between algorithms and datasets (i.e. some algorithms perform well particularly for some datasets). Details of how this synthetic meta-dataset was created can be found in (Nguyen et al., 2021).

2.2. Challenge protocol

We organized a novel two-phase competition protocol:

- **Development phase**: participants make as many submissions as they want, which are evaluated on the *validation learning curves*.
- **Final test phase**: no new submissions are accepted in this phase. The last submission of each participant in the Development phase is transferred automatically to this phase. It is then evaluated on the *test learning curves*.

Note that the meta-datasets are never exposed to the participants in neither phase, because this is a challenge with code submission (only the participants' agent sees the data).

This setting is novel because it is uncommon in reinforcement learning to have a separate phase for agent "development" and agent "testing". Validation learning curves are used during the development phase and test learning curves during the final phase, to prevent agent overfitting. Moreover, we implemented the k-fold meta-cross-validation (with k=6) to reduce variance in the evaluations of the agents (i.e. 25 datasets for meta-training and 5 datasets for meta-testing). The final results are averaged over datasets in the test folds.

During meta-training, learning curves and meta-data collected on 25 datasets are passed to the agent for meta-learning in any possible ways implemented by the agent. Then during meta-testing, one dataset is presented to the agent at a time. The agent interacts back and forth with an "environment", similarly to a Reinforcement Learning setting. It keeps suggesting to reveal algorithms' validation learning curves and choosing the current best performing algorithm based on observations of the partially revealed learning curves.

2.3. Evaluation

In this challenge series, we want to search for agents with high "any-time learning" capacities, which means the ability to have good performances if they were to be stopped at any point in time. Hence, the agent is evaluated by the Area under the agents' Learning Curve (ALC) which is constructed using the learning curve of the best algorithms chosen at each time step (validation learning curve in the Development phase, and the test learning curves in the Final phase). The computation of the ALC is explained in (Nguyen et al., 2021). The results will be averaged over all meta-test datasets and shown on the leaderboards. The final ranking is made according to the average test ALC.

As indicated in the competition rules, participants should make efforts to guarantee the reproducibility of their methods (e.g. by fixing all random seeds involved). In the Final Phase, all submissions were run **three times**, and the run with the **worst performance** is used for the final ranking[2].

On each dataset $\mathcal{D}_i$, we evaluated an agent $\mathcal{A}_j$ by the Area under the Learning curve ALC_i^j of the agent on the dataset. In the final ranking, the agent is ranked based on its average ALC over all datasets ($\mathcal{N} = 30$ datasets):

$$\mu_j = \frac{\sum_{i=1}^{\mathcal{N}} ALC_i^j}{\mathcal{N}} \tag{1}$$

[1] https://codalab.lisn.upsaclay.fr/competitions/753

[2] The output of each run can be found in this Google Drive folder: [link]

To measure the variability of an agent $\mathcal{A}_j$, we computed the standard deviation of ALC scores obtained by the agent over all datasets:

$$\sigma_j = \sqrt{\frac{\sum_{i=1}^{\mathcal{N}} (ALC_i^j - \mu_j)^2}{\mathcal{N}}} \tag{2}$$

3. Analyses of the first round results

Results, final rankings, and prizes in the Final phase of the first round are shown in Table 1. The 1st, 2nd, and 3rd ranked teams qualifying for prizes, as per the challenge rules[3], were awarded prizes of 500$, 300$, and 200$, respectively. In the remainder of this section, we give a more in-depth view of how each team performed on each dataset in this round, compared to our baselines.

In this round, the participants are asked to solve two tasks simultaneously: *algorithm selection* and *budget allocation*. Both of them are crucial to achieving our goal of maximizing the area under an agent's learning curve. We found approaches submitted by the participants using a wide range of methods, from simple (e.g. using algorithm ranking and pre-defined values for Δt) to more sophisticated (e.g. predicting scores and timestamps of unseen learning curve points). The results first indicate that using both learned policies (models) for choosing algorithms and spending time budget (used by team *MoRiHa* and *neptune*) yields better ALC scores than hard-coded ones (e.g. using a fixed pre-defined list of Δt in *AIpert* and our *DDQN* baseline).

According to Table 2, team *MoRiHa* obtained the highest average ALC of 0.43 in the final phase. It succeeded in achieving the highest ALC score on 21 out of 30 datasets. In addition, it performed notably better than other teams in some datasets, such as *tania*, *robert*, *newsgroups*, and *marco*. They are datasets of either multi-label or multi-class classification tasks with a very high number of features, which can be considered difficult tasks in the AutoML challenge.

Team *neptune* has a score of 0.42 which is very close to the winner's, followed by team *AIpert* with a score of 0.40. Team *automl-freiburg*, which was ranked 4th, achieved a slightly lower score (0.37) than our DDQN baseline (0.38), and so did team *automl-hannover* (0.32).

The successes of the top-ranked team can be explained by the strategies they implemented. Team *MoRiHa*, which finished in the 1st place, uses a simple yet efficient approach that explores the most promising algorithms and avoids wasting time switching between too many different algorithms. Interestingly, team *neptune* learns a policy for allocating time budget using learning curve convergence speed. The Reinforcement Learning-based approach of team *AIpert* is very intuitive as our competition borrows the RL framework to formulate our problem, which also explains our baseline choice of DDQN. However, by complementing it with the K-means clustering method, *AIpert*'s approach achieved higher performance than our baseline. Both *AIpert* and *automl-freiburg* share the same idea of suggesting algorithms based on dataset similarity.

[3] https://codalab.lisn.upsaclay.fr/competitions/753

4. Winning solutions

In this section, we briefly introduce strategies of the winning solutions. More details on their implementations can be found in our factsheet summary (Appendix A) or in individual factsheets provided by the winning teams in Table 1.

4.1. Team *MoRiHa* (1st place)

According to team *MoRiHa*, they focus on "doing the right thing at the right time". Their agent is very goal-oriented (i.e. maximizing the area under the learning curve) and does not rely on complex models or expensive computations. They emphasize the importance of having an accurate schedule regarding the invested time (choosing Δt for each query) in order to avoid wasting time budget. One of their key findings is that switching between algorithms during exploration is very costly and it is rarely beneficial to switch the explored algorithm more than once. They build a **time model** for each algorithm to predict the time when the **first point** on the algorithm's learning curve is available. In addition, they keep a **list of algorithms ranked descending** based on their ALC in the meta-training phase. In meta-testing their algorithm scheduler explores algorithms in an order starting from the best algorithm. If an algorithm's learning curve stales, the next algorithm will be chosen. Time allocation is done using the time models and a **heuristic** procedure. This is the only team having an assumption that the learning curves are monotonically increasing.

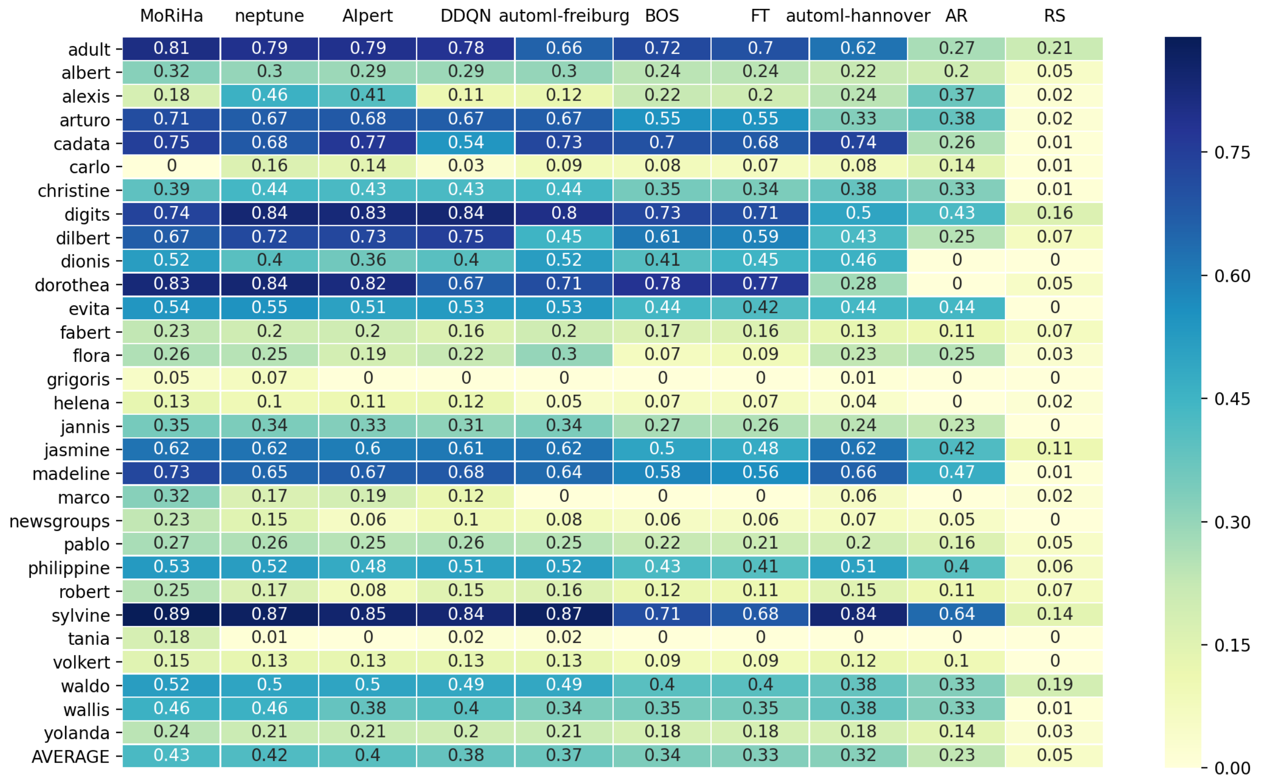
4.2. Team *neptune* (2nd place)

As described by team *neptune*, they train a learning curve predictor to **predict unseen points** on the learning curves (i.e. by interpolating the original scores) in order to find the best algorithm for a new dataset. In addition, they train an algorithm classifier to categorize algorithms into three groups based on their **learning curve convergence speed**: Fast/Median/Slow. Different budget allocation strategies will be selected according to the algorithm's convergence type. Regarding their implementation, they train MLP networks to perform both tasks: learning curve prediction and algorithm classification.

Table 1. **Final phase ranking of the first round**. Teams are ranked based on their average ALC scores, which were recorded on the worst run among three runs. Team *MoRiHa* was ranked 1st, but not qualified for a monetary prize. Teams *neptune*, *AIpert*, and *automl-freiburg* are qualified for the prizes.

Rank	**Team (username)**	**ALC score**	**Monetary Prize qualification**	**Comments/ source code**
1	**MoRiHa** (username: MoRiHa) Felix Mohr et al.	0.43	NO	This team is not qualified for a monetary prize due to close relation with the organizers [CODE URL] [FACTSHEET]
2	**neptune** (username: neptune) Xiangzhen Kong	0.42	OK	[CODE URL] [FACTSHEET]
3	**AIpert** (username: AIpert) Giorgia Franchini et al.	0.40	OK	[CODE URL] [FACTSHEET]
4	**automl-freiburg** (username: automl-freiburg) Sebastian Pineda-Arango et al.	0.37	OK	[CODE URL] [FACTSHEET]
5	**automl-hannover** (username: amsks) Aditya Mohan et al.	0.32		[CODE URL] [FACTSHEET]
6	**pprp** (username: pprp)	0.24		
7	**arushsharma24** (username: arushsharma24)	0.23		
8	**Xavier** (username: Xavier)	0.17		

Table 2. **ALC scores of the top 5 methods: MoRiHa, neptune, AIpert, automl-freiburg, automl-hannover, and our baselines: Double Deep Q Network (DDQN), Best on Samples (BOS), Freeze-Thaw BO (FT), Average Rank (AR), Random Search (RS)**. The reported scores correspond to the worst of 3 runs for each method. The last row shows the average ALC scores (in descending order, from left to right) over 30 datasets.


0.75
0.60
0.45
0.30
0.15
0.00

	MoRiHa	neptune	AIpert	DDQN	automl-freiburg	BOS	FT	automl-hannover	AR	RS
adult	0.81	0.79	0.79	0.78	0.66	0.72	0.7	0.62	0.27	0.21
albert	0.32	0.3	0.29	0.29	0.3	0.24	0.24	0.22	0.2	0.05
alexis	0.18	0.46	0.41	0.11	0.12	0.22	0.2	0.24	0.37	0.02
arturo	0.71	0.67	0.68	0.67	0.67	0.55	0.55	0.33	0.38	0.02
cadata	0.75	0.68	0.77	0.54	0.73	0.7	0.68	0.74	0.26	0.01
carlo	0	0.16	0.14	0.03	0.09	0.08	0.07	0.08	0.14	0.01
christine	0.39	0.44	0.43	0.43	0.44	0.35	0.34	0.38	0.33	0.01
digits	0.74	0.84	0.83	0.84	0.8	0.73	0.71	0.5	0.43	0.16
dilbert	0.67	0.72	0.73	0.75	0.45	0.61	0.59	0.43	0.25	0.07
dionis	0.52	0.4	0.36	0.4	0.52	0.41	0.45	0.46	0	0
dorothea	0.83	0.84	0.82	0.67	0.71	0.78	0.77	0.28	0	0.05
evita	0.54	0.55	0.51	0.53	0.53	0.44	0.42	0.44	0.44	0
fabert	0.23	0.2	0.2	0.16	0.2	0.17	0.16	0.13	0.11	0.07
flora	0.26	0.25	0.19	0.22	0.3	0.07	0.09	0.23	0.25	0.03
grigoris	0.05	0.07	0	0	0	0	0	0.01	0	0
helena	0.13	0.1	0.11	0.12	0.05	0.07	0.07	0.04	0	0.02
jannis	0.35	0.34	0.33	0.31	0.34	0.27	0.26	0.24	0.23	0
jasmine	0.62	0.62	0.6	0.61	0.62	0.5	0.48	0.62	0.42	0.11
madeline	0.73	0.65	0.67	0.68	0.64	0.58	0.56	0.66	0.47	0.01
marco	0.32	0.17	0.19	0.12	0	0	0	0.06	0	0.02
newsgroups	0.23	0.15	0.06	0.1	0.08	0.06	0.06	0.07	0.05	0
pablo	0.27	0.26	0.25	0.26	0.25	0.22	0.21	0.2	0.16	0.05
philippine	0.53	0.52	0.48	0.51	0.52	0.43	0.41	0.51	0.4	0.06
robert	0.25	0.17	0.08	0.15	0.16	0.12	0.11	0.15	0.11	0.07
sylvine	0.89	0.87	0.85	0.84	0.87	0.71	0.68	0.84	0.64	0.14
tania	0.18	0.01	0	0.02	0.02	0	0	0	0	0
volkert	0.15	0.13	0.13	0.13	0.13	0.09	0.09	0.12	0.1	0
waldo	0.52	0.5	0.5	0.49	0.49	0.4	0.4	0.38	0.33	0.19
wallis	0.46	0.46	0.38	0.4	0.34	0.35	0.35	0.38	0.33	0.01
yolanda	0.24	0.21	0.21	0.2	0.21	0.18	0.18	0.18	0.14	0.03
AVERAGE	0.43	0.42	0.4	0.38	0.37	0.34	0.33	0.32	0.23	0.05

4.3. Team *AIpert* (3rd place)

According to team *AIpert*, their method aims at uncovering good algorithms as fast as possible, using a low-computational cost and simple process. The novelty lies in the combination of an off-policy Reinforcement Learning method (**Q-learning**) and **K-means clustering** model. As similar datasets usually have the same *learning behavior*, organizing similar datasets into groups based on their meta-features is essential. They thus build a Q-learning matrix for each dataset cluster (12 clusters in total). In meta-training, each dataset is seen multiple times and the corresponding Q-learning matrix is updated. This allows the agents to be exposed to more situations (different observations and rewards) on a dataset. In meta-testing, they first determine which cluster the given dataset belongs to. Then, the Q-learning matrix associated with the cluster is utilized as a policy to guide the agent. Δt is not chosen as an absolute value but from **a fixed list of time portions** of the total time budget.

4.4. Team *automl-freiburg* (4th place)

As described by team *automl-freiburg*, their algorithm selection policy is based on a **Deep Gaussian Processes** (DGP) surrogate, in a similar way to FSBO (Wistuba & Grabocka, 2021). The surrogate aims to predict the performance of an algorithm at time $t + \Delta t$, with Δt chosen by a **trained budget predictor**. The DGP is trained during the meta-training phase, and fine-tuned in the meta-testing phase. During meta-training, they store the best algorithm for each dataset to be used as the first algorithm to query during meta-testing. The “best” algorithm is defined as the one that has the highest $\frac{y_0}{t_0}$, which means they favor algorithms that achieve **high performance early**. Given a new dataset in meta-testing, the best algorithm of the **closest dataset** (previously seen in meta-training, based on the euclidean distance) will be selected. During meta-testing, they keep updating the observed algorithms and fine-tune the DGP surrogate.

5. Lessons learned and new design

In this section, we describe the limitations of our original design and how we addressed them in the second round.

First, the time when a new point on a learning curve appears quite randomly to the participants, making it difficult to predict and learn how to spend the given time budget wisely. Second, if the time budget Δt allocated by an agent is not enough to query a new point on a model’s learning curve, Δt will be subtracted from the total time budget $\mathcal{T}$ without returning any new learning curve information. This may not be realistic because in a real-life scenario, if one decided to train a model in Δt, a new model should be returned along with a new accuracy score.

We thus provide a new type of learning curve for the second round: learning curve as a function of training data size, as opposed to learning curves as a function of time. The time budget for querying a point on a learning curve will be returned by the environment and not chosen by the agent, which means that the agent has to pay whatever it costs.

Second, the test learning curves were highly correlated with the validation curves. Therefore, one could overfit the former by simply overfitting the latter. In the second round, the agent will always be evaluated using the test learning curves but on a completely different set of datasets in each phase (Development phase and Final phase).

The second round of our challenge was accepted at the AutoML-Conf 2022. Participation in the first round is not a prerequisite for the second round. The second round comes with some new features, including:

- **Learning curve**: we focus on learning curves as functions of training data size. We thus collected a new large meta-dataset of such learning curves.
- **Competition protocol**: Given a portfolio of algorithms, an agent suggests which algorithm and the amount of training data to evaluate the algorithm on a new task (dataset) efficiently. The agent observes information on both the *training learning curves* and *validation learning curves* to plan for the next step. Test learning curves, which are kept hidden, will be used for evaluating the agent.
- **Data split**: We use half of the meta-dataset for the Development phase and the other “fresh” half to evaluate the agent in the Final phase.

The second round is split into three phases:

- **Public phase (1 week)**: participants practice with the given starting kit and sample data
- **Development phase (6 weeks)**: participants submit agents that are meta-trained and meta-tested on the platform. 15 datasets will be used in this phase.
- **Final phase (1 week)**: no further submissions are made in this phase. Your last submission in the Development phase will be forwarded automatically to this phase and evaluated on 15 fresh datasets (not used in the Development phase).

Like in the first round, this is a competition with code submission and the participants do not see the data in either phase (only their submitted agent is exposed to the meta-datasets).

We created a new meta-dataset of pre-computed learning curves of 40 algorithms with different hyperparameters on 30 datasets used in the AutoML challenge. We added meta-features of datasets and hyperparameters of algorithms. We respected the data split of the AutoML challenge to produce three sets of learning curves for each task, from the **training, validation, and test sets**. The type of metric used to compute the learning curves of the meta-dataset is provided in the meta-features of the dataset. We also generated a new synthetic meta-dataset that contains 12000 learning curves, in a similar way used in the first round, but with a new type of learning curve as explained above (learning curve as a function of training data size).

In meta-training, the following data is given to the agent to meta-learn: meta-features of datasets, hyperparameters of algorithms, training learning curves, validation learning curves, and test learning curves. While in meta-testing, the agent interacts with an environment in a Reinforcement Learning style. Given a portfolio of algorithms, an agent suggests which algorithm and the amount of training data to evaluate the algorithm on a new task (dataset) efficiently. The agent observes information on both the training learning curve and validation learning curve to plan for the next step. An episode ends when the given time budget is exhausted. The following two lines of code demonstrate the interactions between the agent and the environment:

$$action = trained\_agent.suggest(observation)$$
$$observation, done = env.reveal(action)$$

with:

observation : a tuple of $(A, p, t, R\_train\_A\_p, R\_validation\_A\_p)$, with:

- A: index of the algorithm provided in the previous action,
- p: decimal fraction of training data used, with the value of p in [0.1, 0.2, 0.3, ..., 1.0]
- t: the amount of time it took to train A with training data size of p, and make predictions on the training/validation/test sets.
- $R\_train\_A\_p$: performance score on the training set
- $R\_validation\_A\_p$: performance score on the validation set

action: a tuple of (A, p), with:

- A: index of the algorithm to be trained and tested
- p: decimal fraction of training data used, with the value of p in [0.1, 0.2, 0.3, ..., 1.0]

The scoring program automatically chooses the best algorithm at each time step (i.e. the algorithm with the highest validation score found so far, which is different from the first round where it was chosen by the agent) to compute the agent's test learning curve (as a function of time spent). The metric used for ranking on the leaderboard is the Area under the agent's Learning Curve (ALC).

6. Conclusion

The first round results of our challenge have revealed that agents that learn both policies for selecting algorithms and allocating time budget are more successful in our challenge setting. Team *MoRiHa*, who finished in 1st place, outperformed all other teams and our baselines on two-thirds of the datasets. We are looking forward to see whether the findings of the first round will be reinforced in the second round, which presents a novel setting and a new meta-dataset.

For our future work, we want to perform more post-challenge analyses to verify whether progress was made in meta-learning from learning curves. First, we would like to do a point-by-point comparison of the winning methods, based on their fact sheets. Second, to investigate further the winning methods and see what contributed the most to their success, we want to perform systematic experiments in collaboration with the winners. More concretely, we will build a common workflow and ask participants to conduct ablation studies. Lastly, we are also interested in examining the effect of changes in our reward function hyper-parameters on participants' performances.

Software and Data

All software (including starting kit and winning solutions) are open-sourced on our website (https://metalearning.chalearn.org/). The meta-datasets will remained private on the challenge platform (Codalab) to serve as a long-lasting benchmark for research in meta-learning.

Acknowledgements

...

References

Bengio, Y., Courville, A., and Vincent, P. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35: 1798–1828, 08 2013. doi: 10.1109/TPAMI.2013.50.

Chen, Z., Liu, B., Brachman, R., Stone, P., and Rossi, F. *Lifelong Machine Learning*. Morgan amp; Claypool Publishers, 2nd edition, 2018. ISBN 1681733021.

Delange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., and Tuytelaars, T. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021. doi: 10.1109/TPAMI.2021.3057446.

El Baz, A., Guyon, I., Liu, Z., van Rijn, J. N., Treguer, S., and Vanschoren, J. Advances in metadl: Aaai 2021 challenge and workshop. In Guyon, I., van Rijn, J. N., Treguer, S., and Vanschoren, J. (eds.), *AAAI Workshop on Meta-Learning and MetaDL Challenge*, volume 140 of *Proceedings of Machine Learning Research*, pp. 1–16. PMLR, 09 Feb 2021. URL https://proceedings.mlr.press/v140/el-baz21a.html.

Guyon, I., Sun-Hosoya, L., Boullé, M., Escalante, H. J., Escalera, S., Liu, Z., Jajetic, D., Ray, B., Saeed, M., Sebag, M., Statnikov, A., Tu, W.-W., and Viegas, E. *Analysis of the AutoML Challenge Series 2015–2018*, pp. 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.

Liu, Z., Pavao, A., Xu, Z., Escalera, S., Ferreira, F., Guyon, I., Hong, S., Hutter, F., Ji, R., Nierhoff, T., Niu, K., Pan, C., Stoll, D., Treguer, S., Wang, J., Wang, P., Wu, C., and Xiong, Y. Winning solutions and post-challenge analyses of the ChaLearn AutoDL challenge 2019. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 17, 2020.

Mohr, F. and van Rijn, J. N. Learning curves for decision making in supervised machine learning – a survey, 2022.

Nguyen, M. H., Grinsztajn, N., Guyon, I., and Sun-Hosoya, L. Metareveal: Rl-based meta-learning from learning curves. In *Workshop on Interactive Adaptive Learning co-located with European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2021)*, Bilbao/Virtual, Spain, September 2021. URL https://hal.inria.fr/hal-03502358.

Sun-Hosoya, L. *Meta-Learning as a Markov Decision Process*. Theses, Université Paris Saclay (COmUE), December 2019. URL https://hal.archives-ouvertes.fr/tel-02422144.

Sun-Hosoya, L., Guyon, I., and Sebag, M. Activmetal: Algorithm recommendation with active meta learning. In *IAL@PKDD/ECML*, 2018.

Swersky, K., Snoek, J., and Adams, R. Freeze-thaw bayesian optimization. 06 2014.

Vanschoren, J. Meta-learning: A survey, 10 2018.

Wang, Y., Yao, Q., Kwok, J., and Ni, L. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys*, 53:1–34, 06 2020. doi: 10.1145/3386252.

Wistuba, M. and Grabocka, J. Few-shot bayesian optimization with deep kernel surrogates, 2021. URL https://arxiv.org/abs/2101.07667.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., and He, Q. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, PP:1–34, 07 2020. doi: 10.1109/JPROC.2020.3004555.

A. Factsheet Summary

In this section, we summarize the information provided in the participants' factsheets. Only top 5 teams submitted their factsheets, including: *MoRiHa, neptune, AIpert, automl-freiburg,* and *automl- hannover.*

A.1. PRE-PROCESSING & FEATURE ENGINEERING

Question 1: Did you perform any data pre-processing methods?

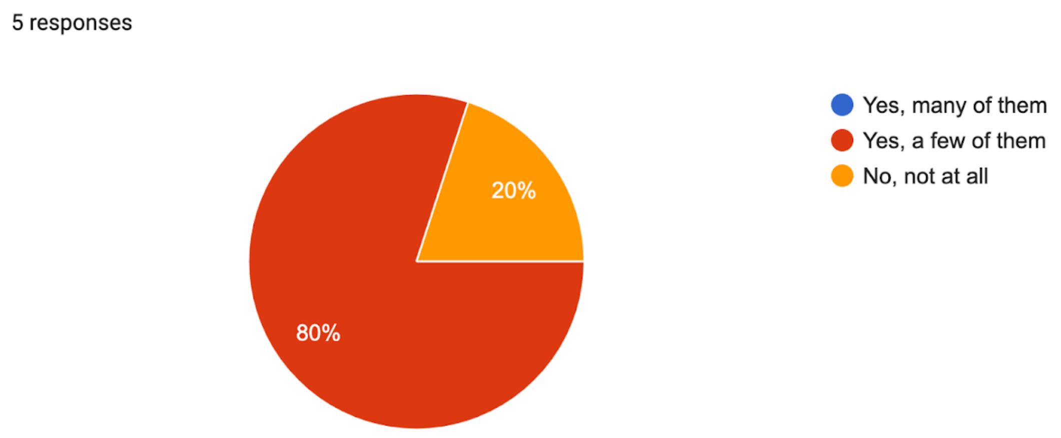


Figure 1. Pre-processing methods include: Standard Scaler, Feature Scaling, One-hot Encoding, extracting the 90% convergence time and final performances.

Question 2: Did you perform feature engineering methods?

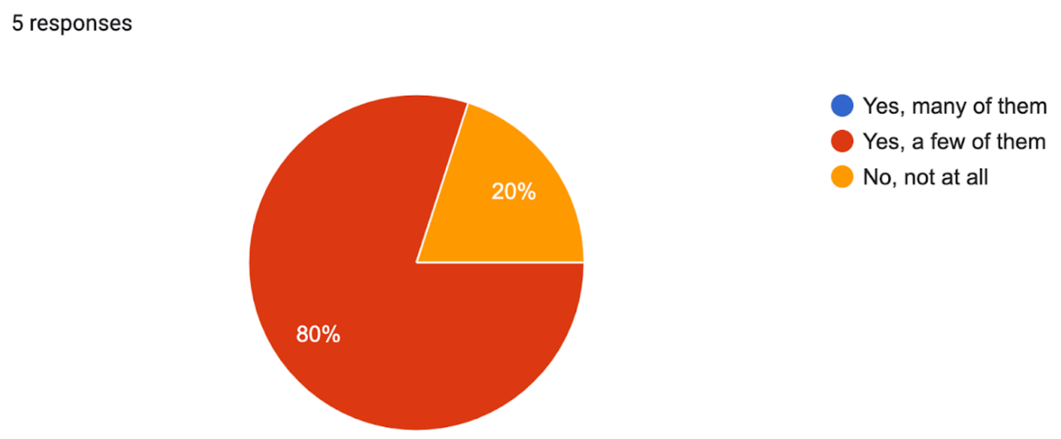


Figure 2. Feature engineering methods include: Clustering; Polynomial features of train_num, feat_num; Multi-step forecasting for data augmentation.

A.2. DATA USED FOR LEARNING

Question 3: Did you use all points on the learning curves or only some of them?

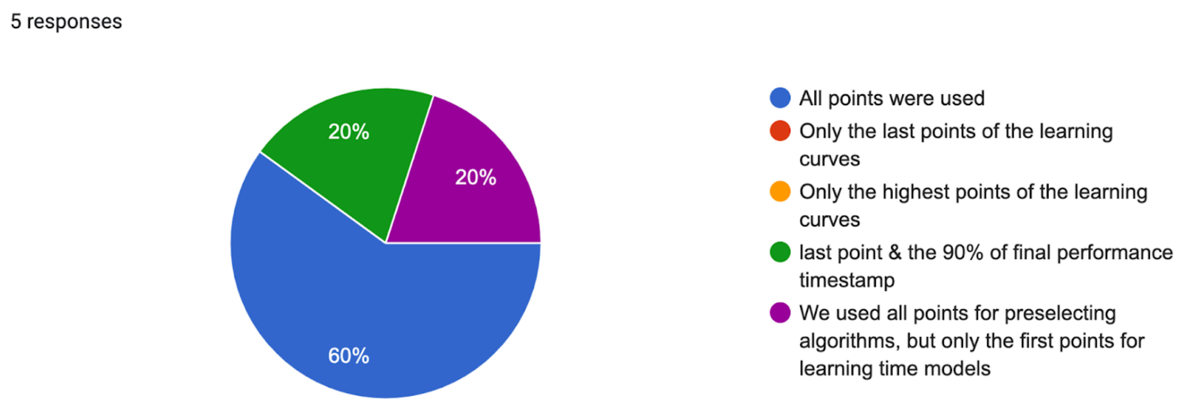


Figure 3. Most of the methods use all points on the learning curves for learning.

Question 4: Did you make use of meta-features of datasets?

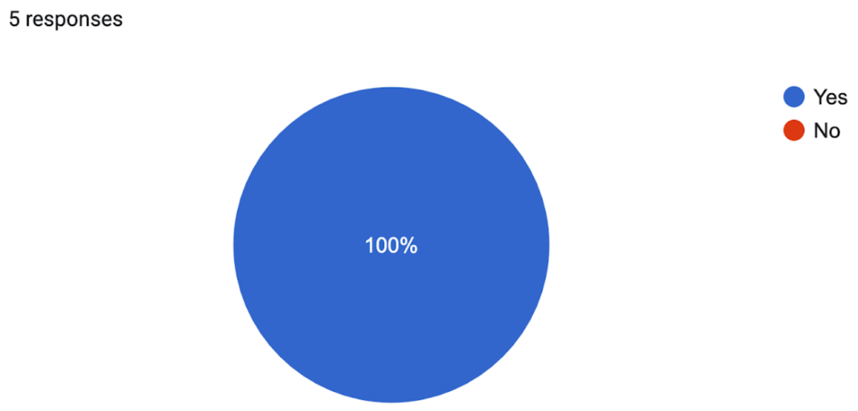


Figure 4. All methods took advantage of meta-features of datasets.

Question 5: Did you implement a Hyperparameter Optimization component for your agent using the provided hyperparameters of algorithms?

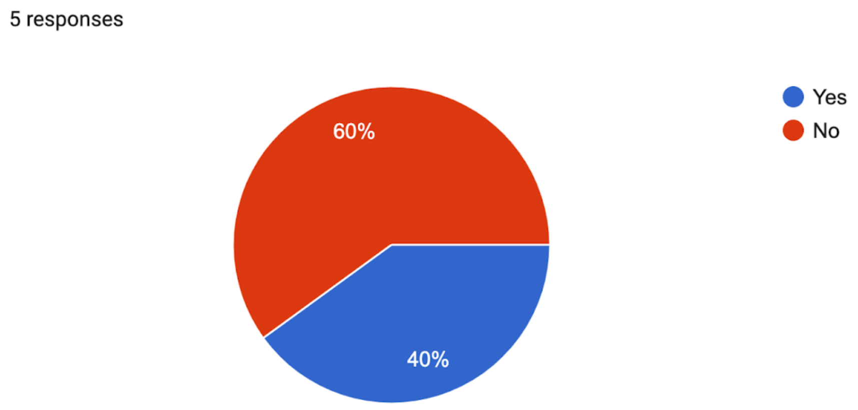


Figure 5. Some HPO tools were used, such as Hyperband and FSBO.

Question 6: In case you used either or both meta-features of datasets and algorithms, did it improve the performance

of your method?

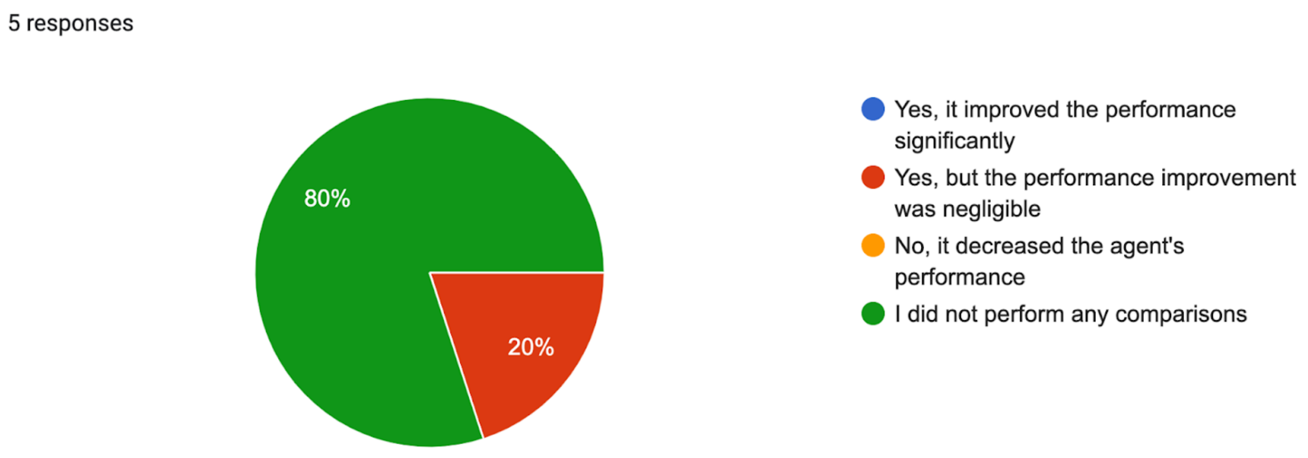


Figure 6. More experiments need to be done to confirm whether meta-features of datasets and algorithms help.

Question 7: In any case, did you find the meta-features useful in our meta-learning setting?

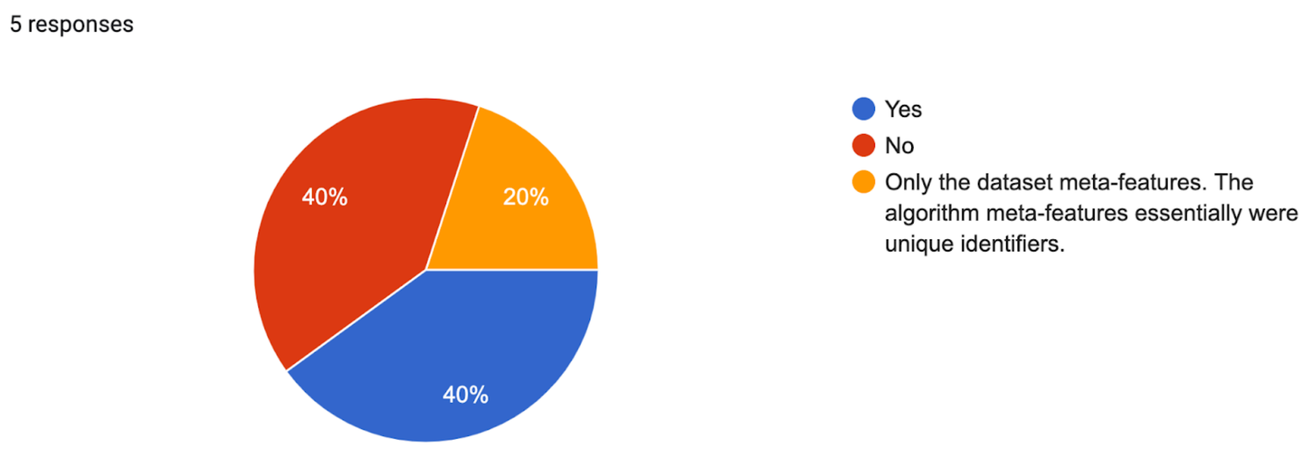


Figure 7. Not all participants find the provided meta-features useful.

A.3. POLICY CHARACTERISTICS

Question 8: Does your agent learn a policy from datasets in the meta-training phase?

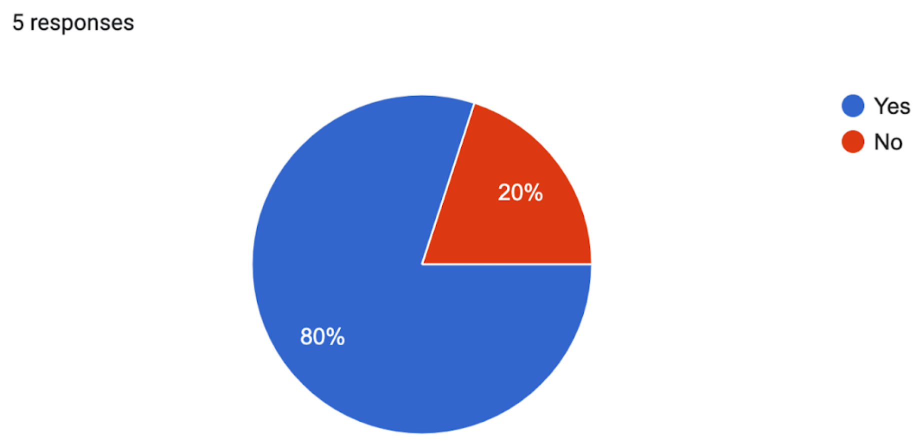


Figure 8. Most of the agents use a learned policy from the meta-training phase.

Question 9: How does your agent manage the exploration-exploitation trade-offs (revealing a known good algorithm's learning curve vs. revealing a new algorithm candidate's learning curve)?

1. With an ϵ greedy policy, in a Reinforcement Learning framework, only in meta- training we create different Q-matrices. In the meta-testing phase, we perform the choice of the new algorithm with the computed Q-matrices.
2. We are very restrictive with switching the explored algorrithm. We preselect the single best performing algorithm from the validation learning curves statically and only explore other algorithms, if its learning curve is stale. So we strongly emphasize exploiting the single best algorithm.
3. A modified Round Robin on the top-k performing algorithms. The incumbent i.e. the value of the top performing algorithm on the test dataset is challenged by zero budget allocated algorithms. Since the training on the validation data allows for immediate look up in the test dataset.
4. Bayesian Optimization
5. We find the best algorithm by a learning curve predictor.

Question 10: Does your agent switch between learning curves during an episode (i.e. switching between training different algorithms on the dataset at hand)?

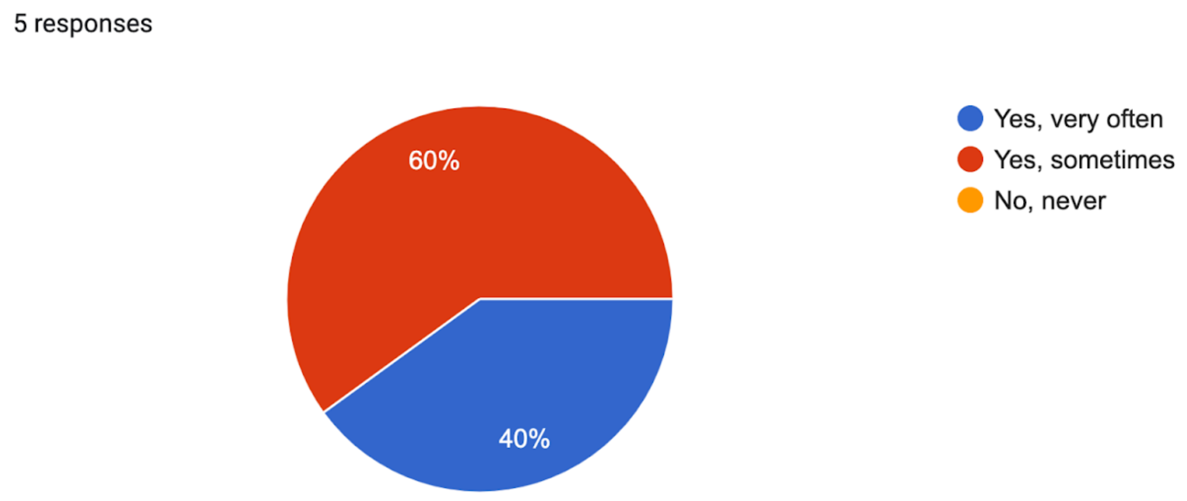


Figure 9. All agents switch between learning curves to find the best algorithm for the task at hand.

Question 11: Does your agent leverage partially revealed learning curves on the dataset at hand?

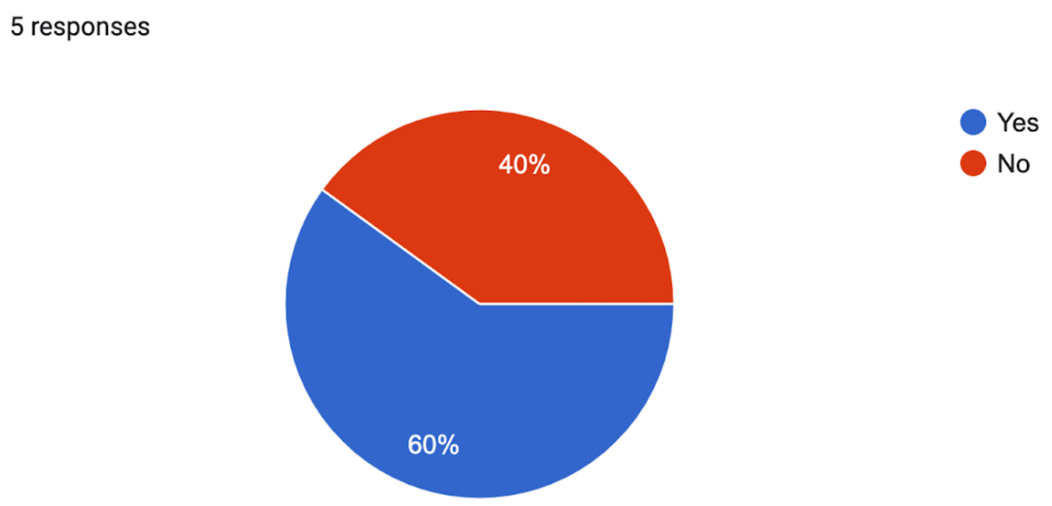


Figure 10. Some agents do not take into account information of partial learning curves on the task at hand for deciding their actions.

Question 12: Did you make any assumptions about the shapes of the learning curves?

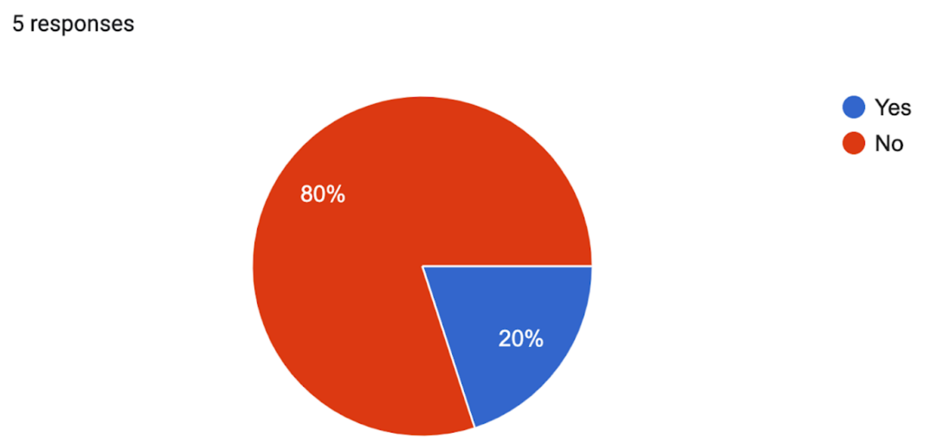


Figure 11. Only one team made an assumption that the learning curves are *monotonically increasing*.

Question 13: Does your agent predict unseen points on a learning curve?

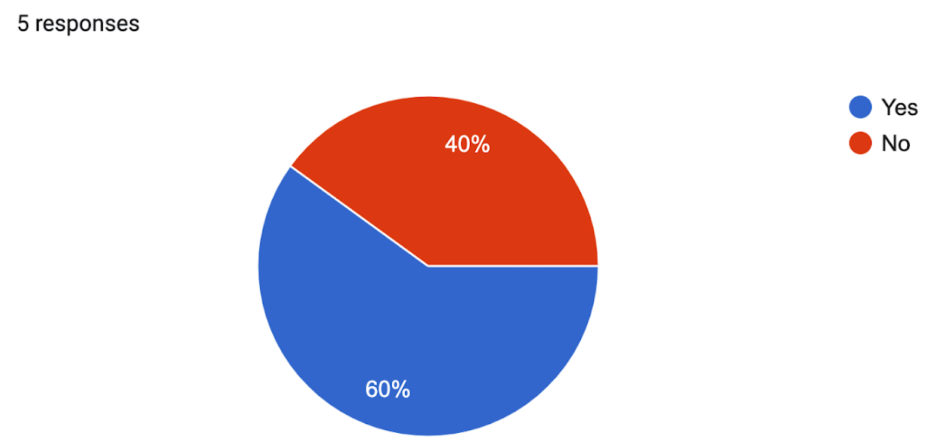


Figure 12. 3 out of 5 agents make decisions based on predicted performance scores.

Question 14: Does your agent perform pairwise comparisons of algorithms' learning curves?

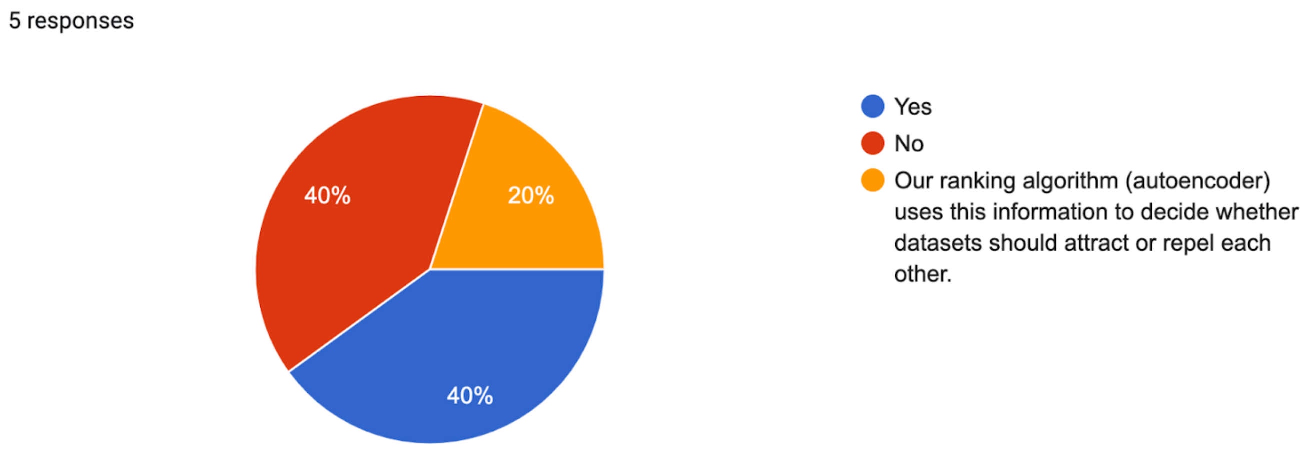


Figure 13. Pairwise comparisons of algorithms' learning curves have been exploited in 60% of the agents.

Question 15: How does your agent spend the given time budgets (i.e. choosing delta_t at each step)?

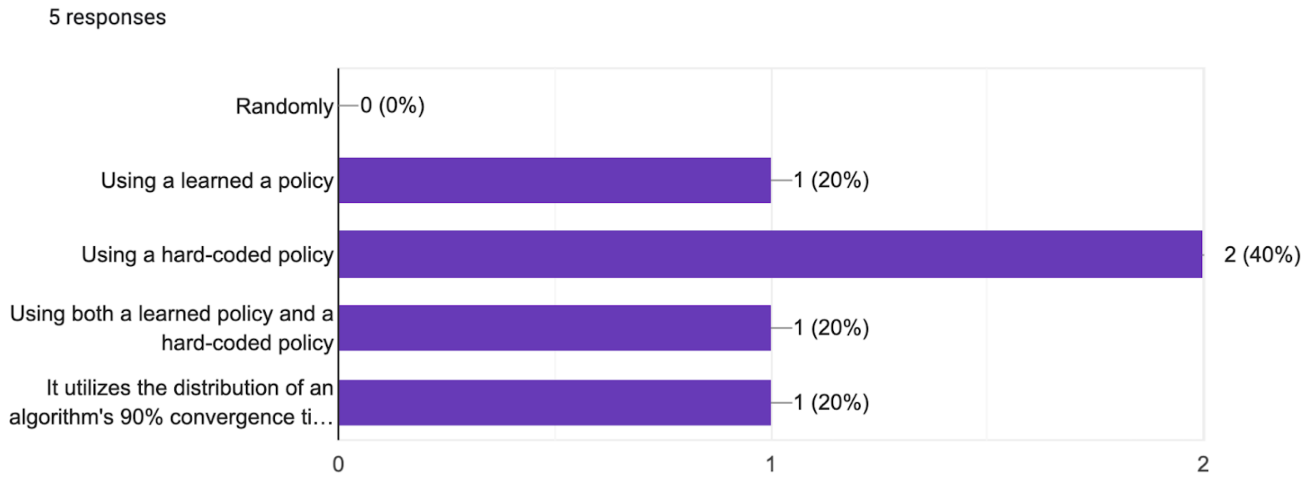


Figure 14. Participants use either or both hard-coded policy and learned policy to distribute a given time budget (no one does it randomly).

Question 16: How does your agent choose which algorithm to contribute to its learning curve at each step (i.e. choosing A_star)?

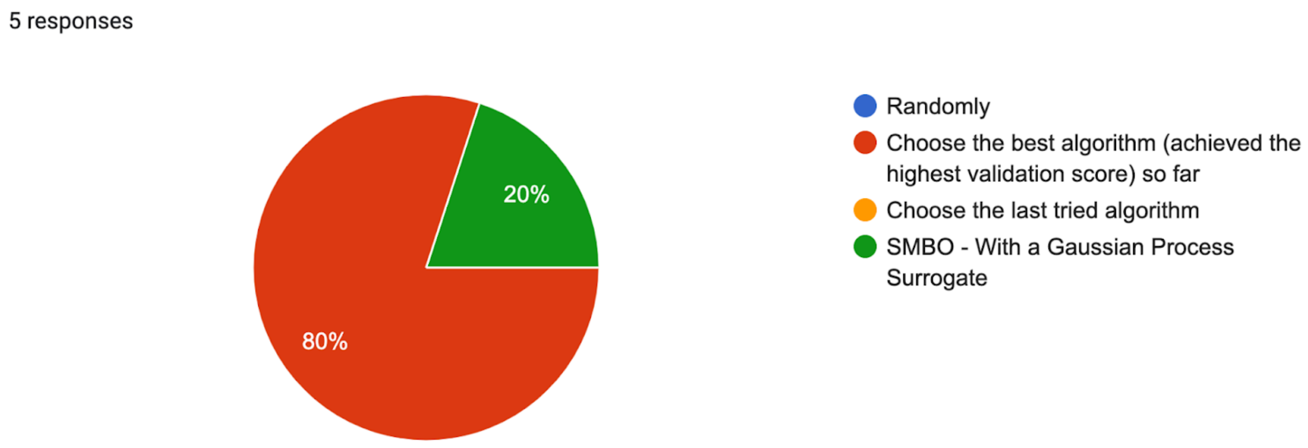


Figure 15. Most of the agents choose the best algorithm so far as A^*, with only one exception that uses Sequential Model Based Optimisation (SMBO) with a Gaussian Process Surrogate.

Question 17: Which phase did you focus on more to improve your agent's performance?

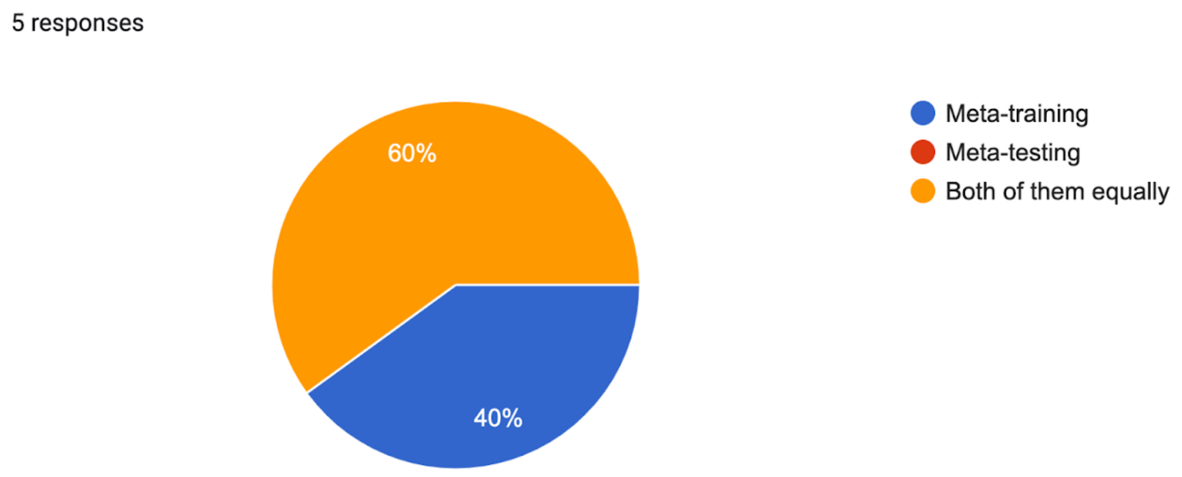


Figure 16. Participants give a slightly higher importance weight to meta-training than meta-testing.

Question 18: Did you build an algorithm ranking?

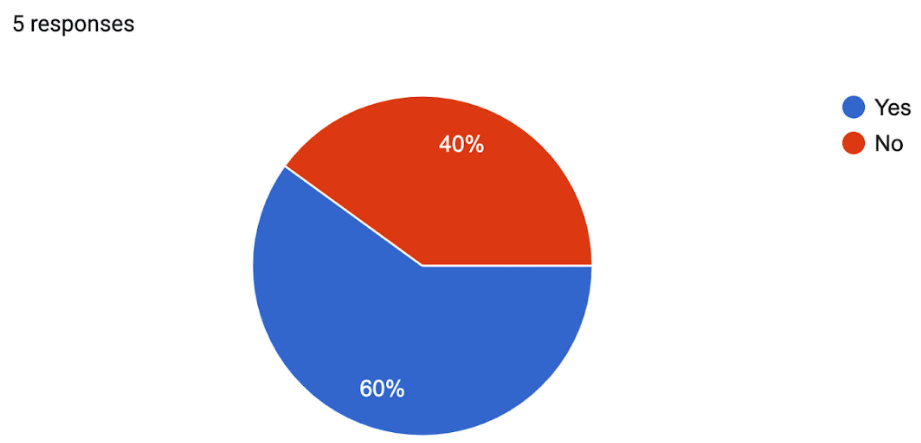


Figure 17. More than half of the participants build an algorithm ranking and use it as a tool for selecting algorithms.

Question 19: Did you use Reinforcement Learning to train your agent?

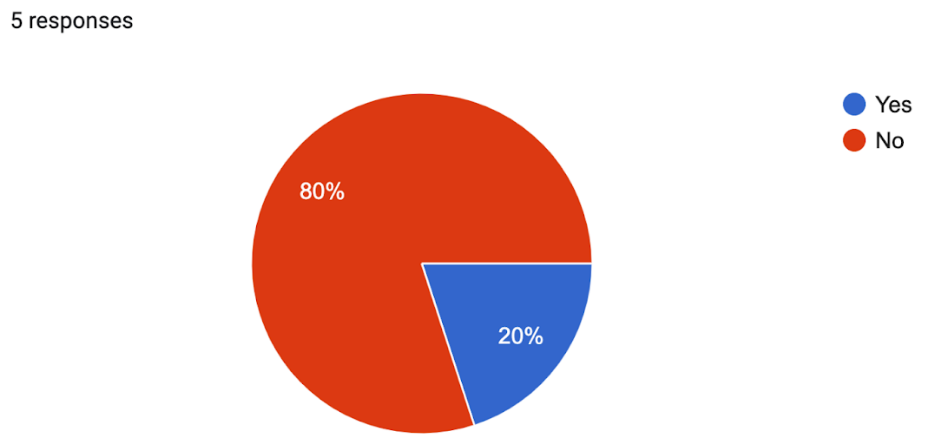


Figure 18. Only one participant applies Reinforcement Learning to train the agent.

A.4. METHOD IMPLEMENTATION

Question 20: What is the percentage of originality of your method/implementation?

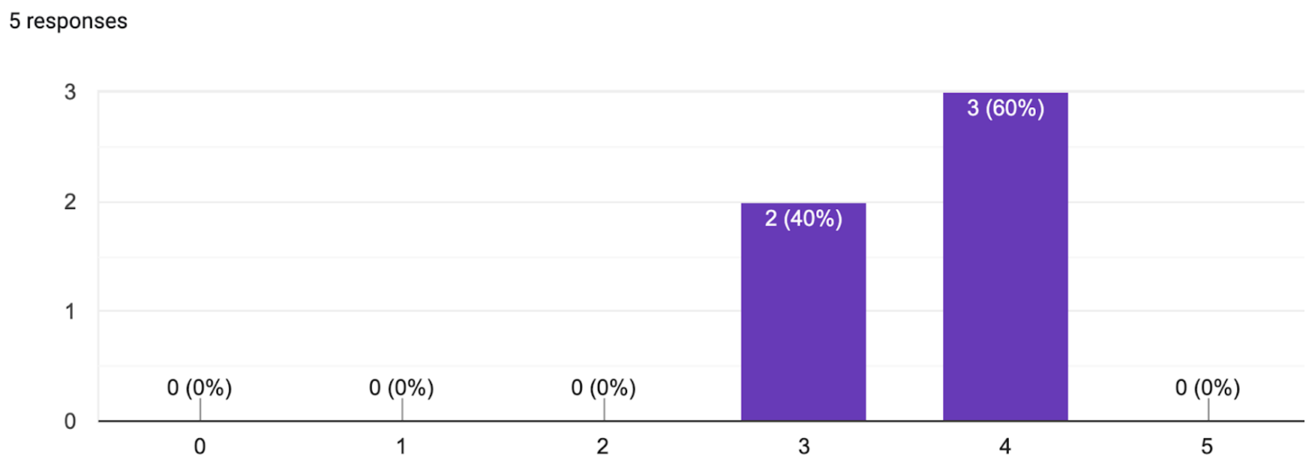


Figure 19. The percentage of originality of their methods/implementations ranges from 60% to 80%.

Question 21: Did you use a pre-trained agent?

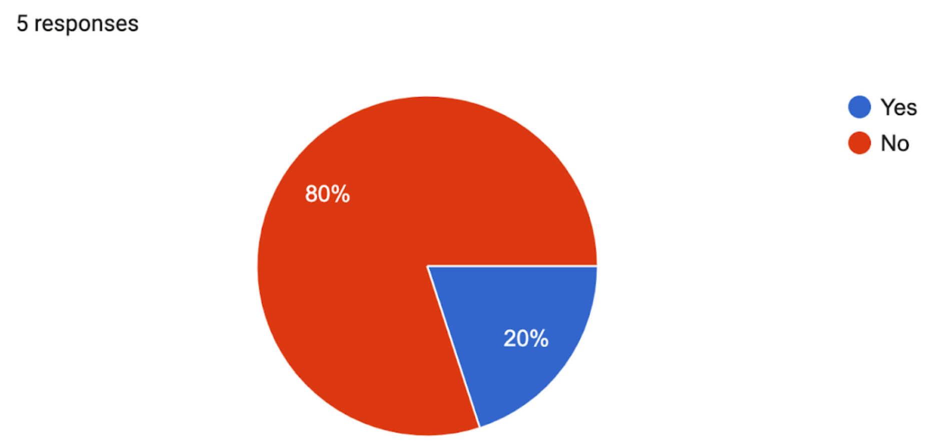


Figure 20. Only one participant pre-train their agent using all the provided datasets in meta-training.

Question 22: Is your method strongly based on an existing solution? Which one(s)?

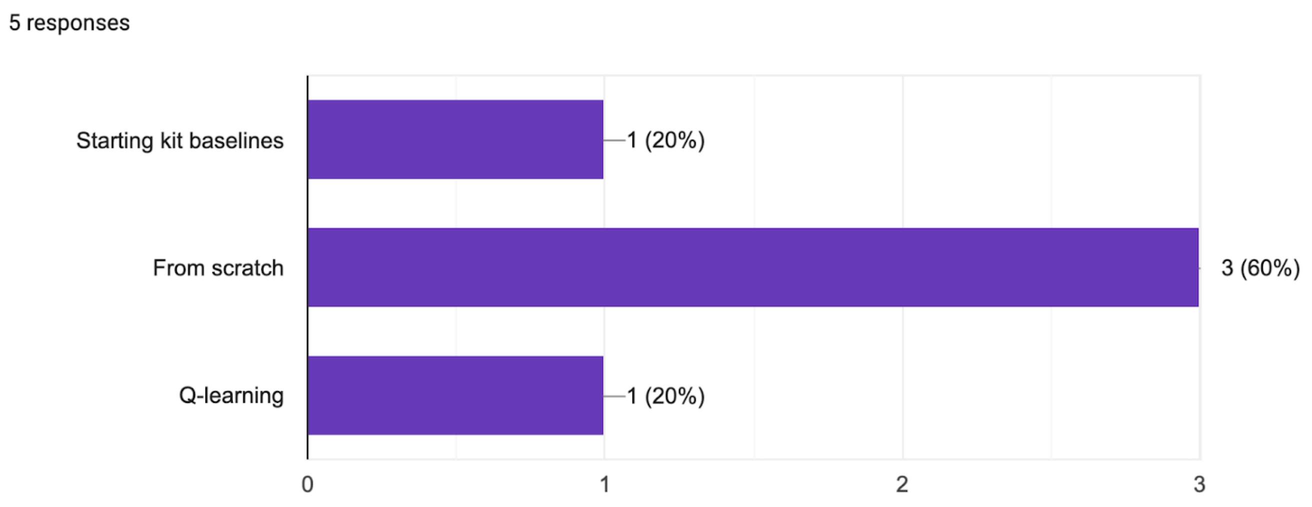


Figure 21. Q-learning and starting kit baselines served as bases for 2 methods, while the other methods were built from scratch.

Question 23: Did you use Neural Networks for your agent?

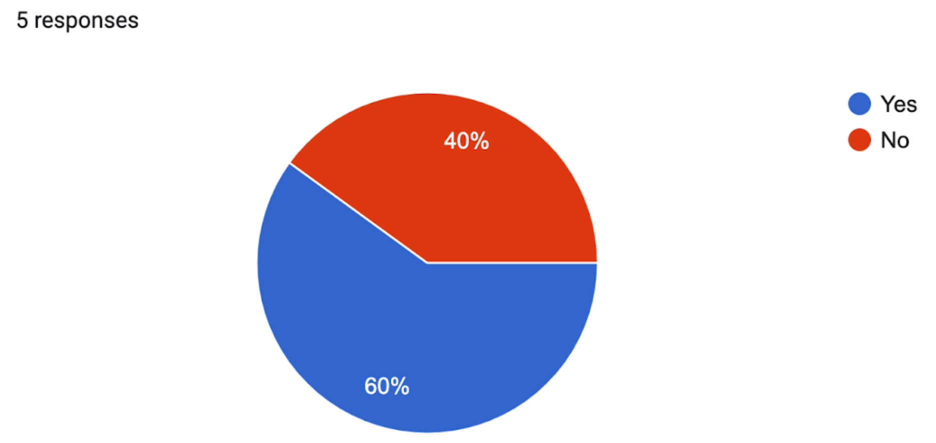


Figure 22. Neural Networks were implemented in 3 out of 5 methods.

Question 24: Do you find the provided libraries / packages / frameworks sufficient?

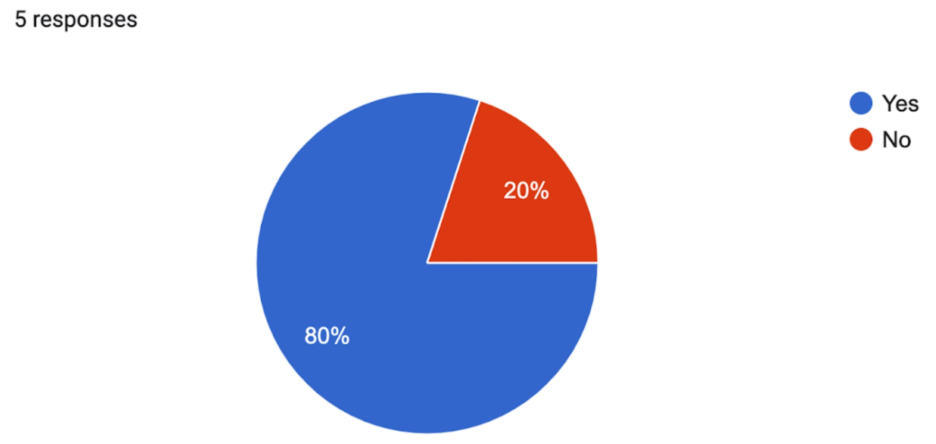


Figure 23. The provided libraries/packages/frameworks were sufficient for most of the participants.

Question 25: Check all Python packages/frameworks you used.

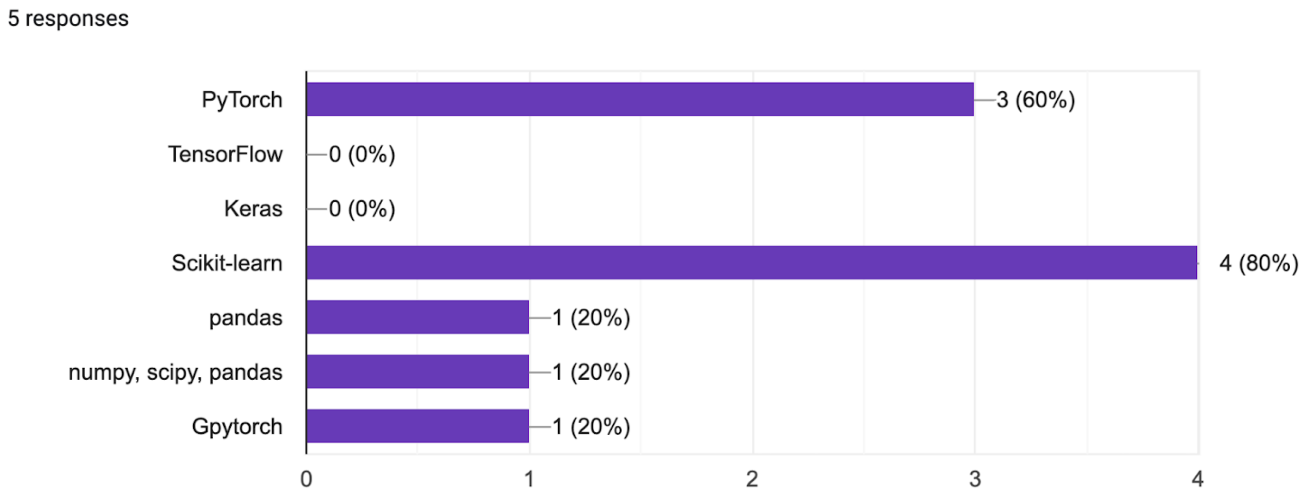


Figure 24. Scikit-learn and Pytorch are the most used packages by participants.

Question 26: Did you use any specific AutoML / Meta-learning / Hyperparameter Optimization libraries?

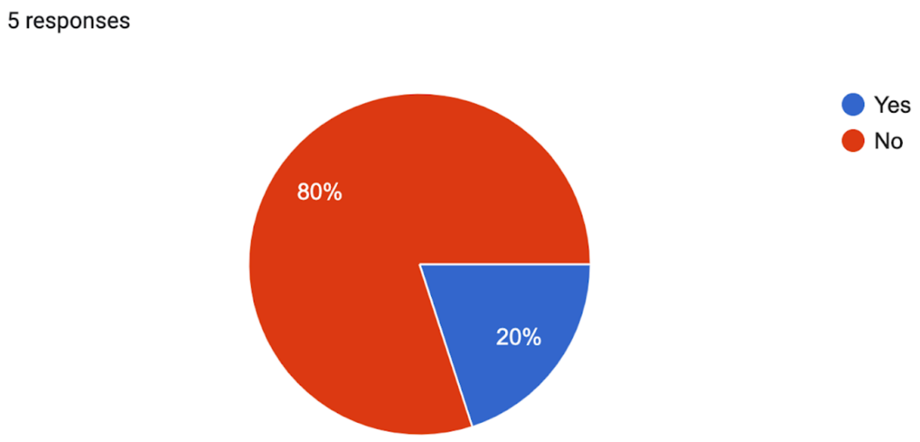


Figure 25. Only one participant uses SMAC for hyperparameter optimization.

Question 27: Was it difficult for you to deal with the provided data format of the learning curves and meta-features?

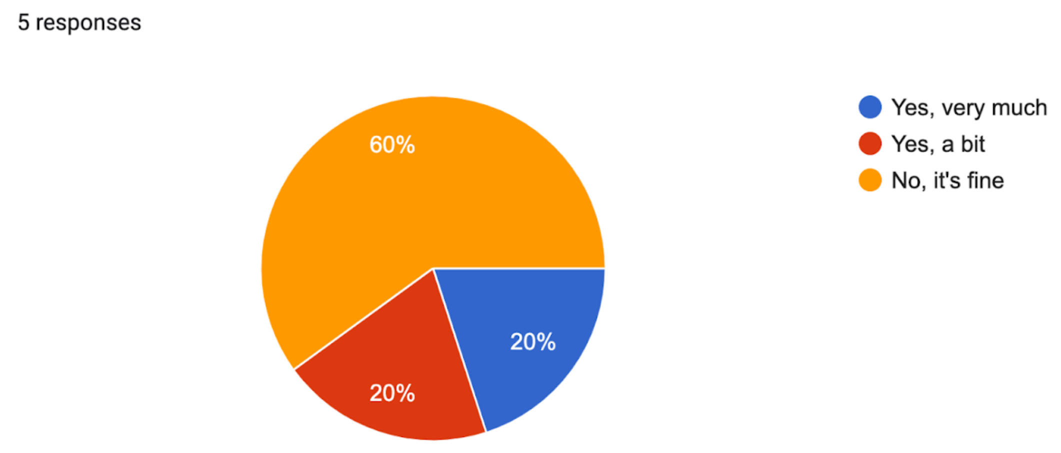


Figure 26. 40% of participants struggled with the provided data format. Some comments include: rather than nested dictionaries create a single dictionary with a tuple identifier (dataset, algorithm).

Question 28: How much time did you spend developing your agents?

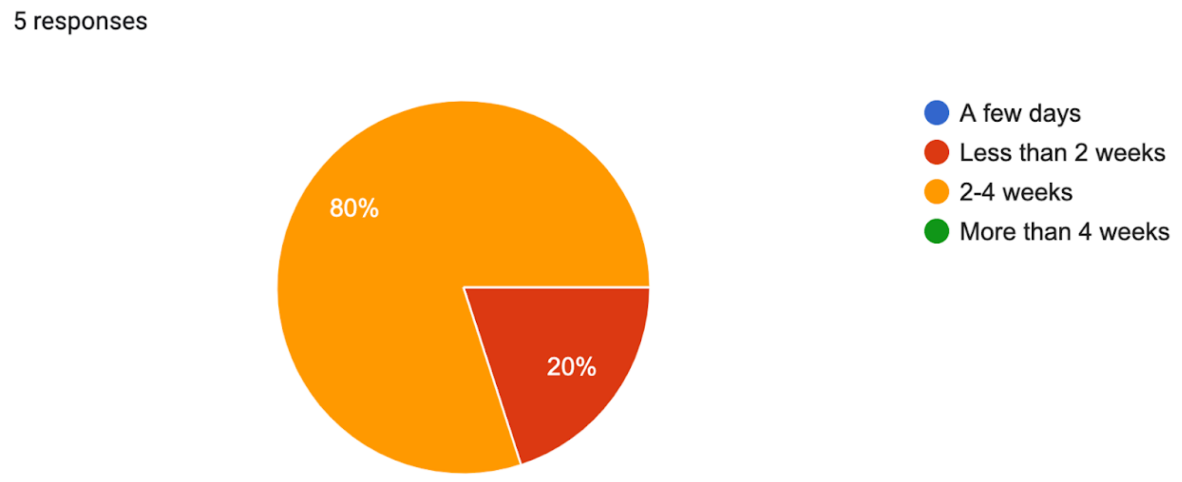


Figure 27. All participants completed their solutions within 4 weeks.

Question 29: What's the difficulty induced by the computation resource (memory, time budget, etc) constraints?

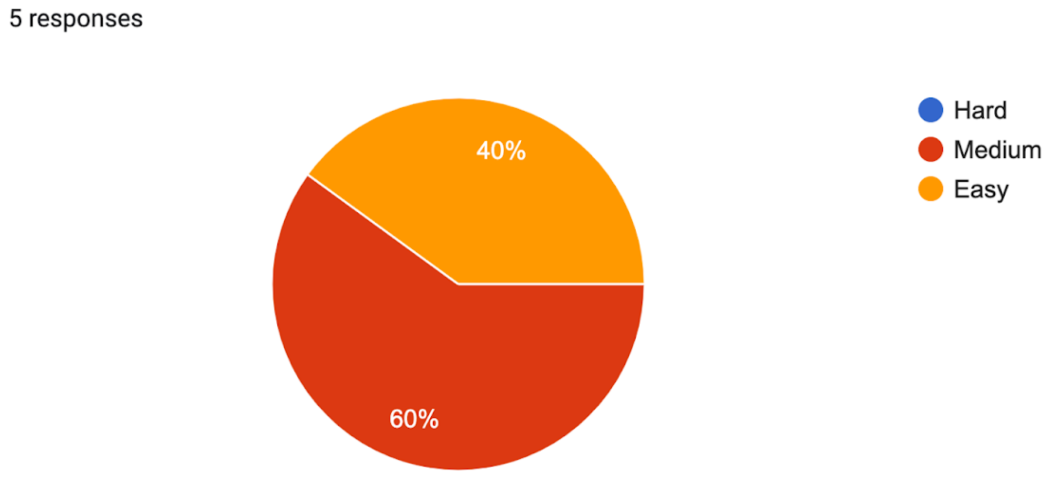


Figure 28. The provided computation resource was reasonable to participants.

A.5. USER EXPERIENCE

Question 30: Was the challenge duration enough for you to develop your methods?

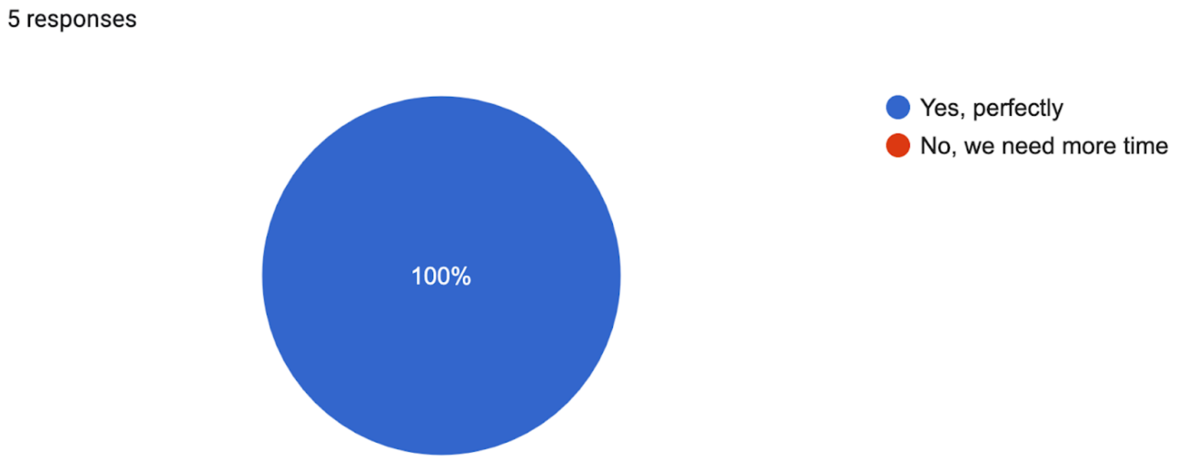


Figure 29. The challenge duration was enough for the participants.

Question 31: Your evaluation on the starting kit

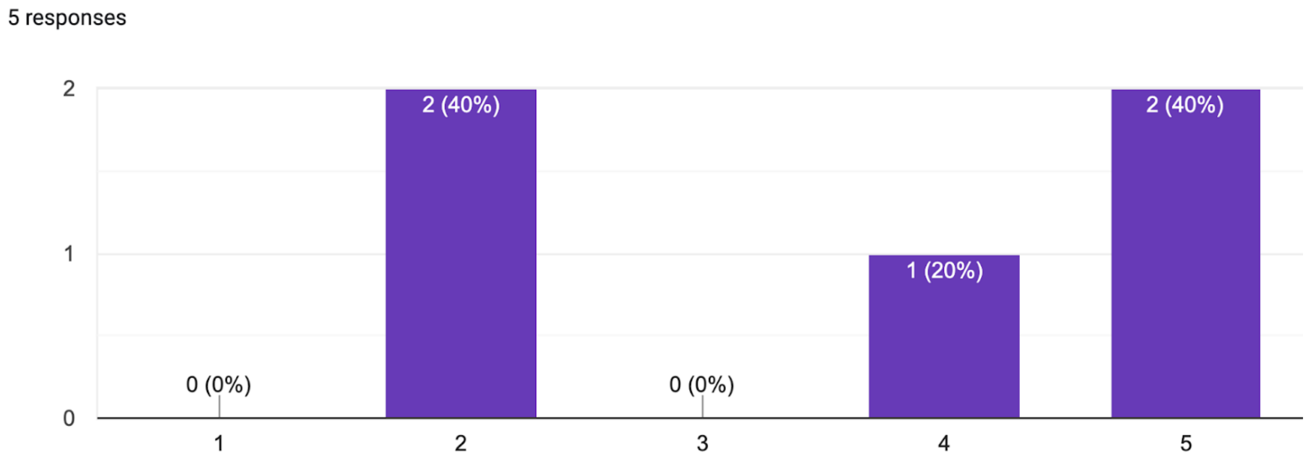


Figure 30. Some improvements on the starting kit need to be made (see below).

Question 32: Which improvements can be made for the starting kit? You are welcome to list all issues and bugs you ran into.

1. The algorithm meta features were non-informative, the dataset meta features' categorical columns at some point were non informative and the categories were not entirely represented in the validation dataset. Also the environment allows to exploit 0 budgets to improve the agent's performance in a non realistic way. Also we wanted to use many libraries, that were not available (e.g pytorch geometric or networkx) or with other versions (e.g. sklearn's quantile regression) was not available initially. The validation & test set (on server) were slightly different. In particular, the distribution of timestamps was dramatically different.

2. Return the whole learning curve after a suggestion, not only the last observed budget and the last observed performance.

3. It may be useful to query the learning curve at specific iterations or explain clearly the meaning of the timestamps of the observed samples. If they are random or there is not an underlying pattern or structure, it is difficult to predict the next budget.

4. The final rank should be obtained by running the methods on other meta-train dataset. In the current set-up, the test curve is highly correlated to the validation curve (seen in the development stage), therefore simply overfitting the latter will probably help to obtain good results in the former.

5. Decreasing the budget when the suggested algorithm+budget tuple is not enough to query a new point in the learning curve is unrealistic. In the real world, once we decide to run some algorithm for a given time-range or number of epochs, we will obtain some change in the accuracy (unless the learning curve is in a plateau).

Question 33: Your evaluation on the challenge website: https://codalab.lisn.upsaclay.fr/competitions/753

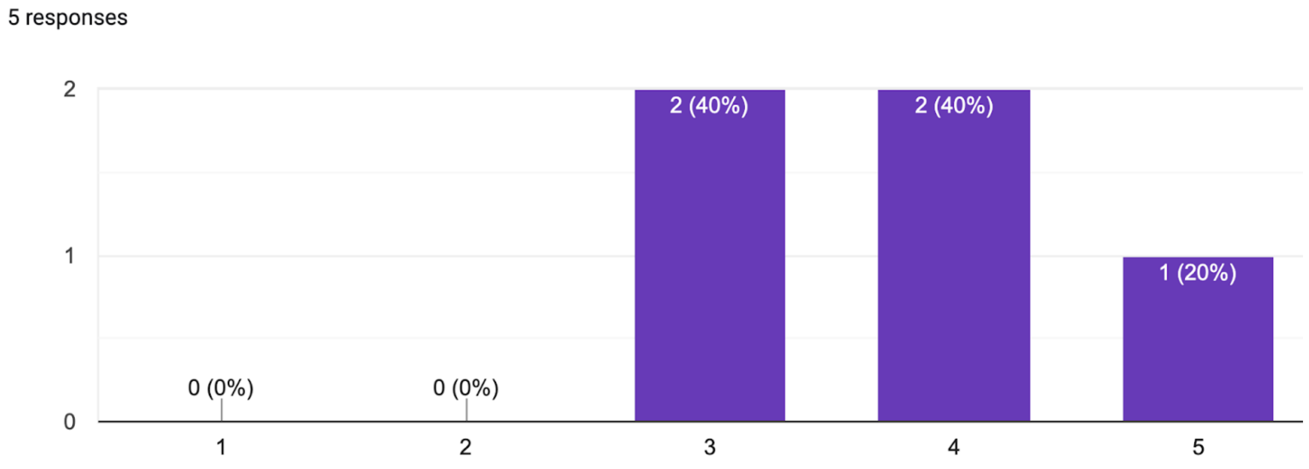


Figure 31. One comment for the challenge website is: the way the ALC is explained is not very straightforward.