The unreasonable effectiveness of early discarding after one epoch in neural network hyperparameter optimization

Egele, Romain; Mohr, Felix; Viering, Tom; Balaprakash, Prasanna

DOI
10.1016/j.neucom.2024.127964

Publication date
2024

Document Version
Final published version

Published in
Neurocomputing

Citation (APA)
Egele, R., Mohr, F., Viering, T., & Balaprakash, P. (2024). The unreasonable effectiveness of early discarding after one epoch in neural network hyperparameter optimization. *Neurocomputing*, *597*, Article 127964. https://doi.org/10.1016/j.neucom.2024.127964

Neurocomputing 597 (2024) 127964

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

The unreasonable effectiveness of early discarding after one epoch in neural network hyperparameter optimization

Romain Egele [a,b,*], Felix Mohr [c], Tom Viering [d], Prasanna Balaprakash [e]

[a] Argonne National Laboratory, USA
[b] Université Paris-Saclay, France
[c] Universidad de La Sabana, Colombia
[d] Delft University of Technology, Netherlands
[e] Oak Ridge National Laboratory, USA

ARTICLE INFO

Keywords:
Hyperparameter optimization
Multi-fidelity optimization
Deep neural network
Learning curve

ABSTRACT

To reach high performance with deep learning, hyperparameter optimization (HPO) is essential. This process is usually time-consuming due to costly evaluations of neural networks. Early discarding techniques limit the resources granted to unpromising candidates by observing the empirical learning curves and canceling neural network training as soon as the lack of competitiveness of a candidate becomes evident. Despite two decades of research, little is understood about the trade-off between the aggressiveness of discarding and the loss of predictive performance. Our paper studies this trade-off for several commonly used discarding techniques such as successive halving and learning curve extrapolation. Our surprising finding is that these commonly used techniques offer minimal to no added value compared to the simple strategy of discarding after a constant number of epochs of training. The chosen number of epochs mostly depends on the available compute budget. We call this approach i-Epoch (i being the constant number of epochs with which neural networks are trained) and suggest to assess the quality of early discarding techniques by comparing how their Pareto-Front (in consumed training epochs and predictive performance) complement the Pareto-Front of i-Epoch.

1. Introduction

Optimizing the configuration of a deep learning pipeline is a complex task that involves properly configuring the data preprocessing, training algorithm, and neural architecture. A configuration is a specification of so-called hyperparameters [1], which control the behavior of pipeline elements and hence can greatly influence its final predictive performance. The objective is to identify the configuration of hyperparameters that achieves the best predictive performance, usually referred to as hyperparameter optimization (HPO).

As HPO is often done from a black-box optimization point of view, that is by observation of input configuration and output performance, a major challenge is the required computation to evaluate candidate hyperparameters by training deep neural networks. This greatly limits the number of testable hyperparameter configurations within a practical time frame. This is why multi-fidelity hyperparameter optimization with early discarding was proposed to switch the black-box problem to a "gray-box" optimization problem by observing the intermediate training performance of neural networks and using it as an estimate of the final performance. Such estimates can in principle be obtained at a computationally cheaper training stage and therefore save overall computation. In deep neural networks, the training epochs are usually used to perform early discarding. An epoch usually refers to making a full pass over the training data. The predictive performance versus the number of epochs is also known as a "learning curve" [2,3].

HPO with early discarding trades-off computation with quality of extrapolated performance. For example, if the neural network is trained for a few epochs, it can save computation but it also means we have little (noisy) training information and therefore increase the chances of mistaking the extrapolation. It is important to note that extrapolated performance is not always absolute but it can also be relative to other candidates such as by predicting a ranking.

A shortcoming of the HPO early discarding literature is the multi-objective ((1) predictive performance, (2) overall computation) optimization viewpoint that such techniques are trying to solve. Therefore, experimental evaluations lack comparison to proper baselines and sometimes present over-optimistic results. For example, it is common to compare early discarding techniques with complete training discarding [4] and, only rare works consider the baseline performance,

* Corresponding author at: Université Paris-Saclay, France.
E-mail address: romain.egele@universite-paris-saclay.fr (R. Egele).

https://doi.org/10.1016/j.neucom.2024.127964
Received 30 January 2024; Received in revised form 3 April 2024; Accepted 29 May 2024
Available online 1 June 2024

which minimizes computation by stopping the training after a single epoch [5,6] during HPO and possibly selecting from the top-k models after further training. We call this baseline "1-Epoch" or more generally i-Epoch when the training is stopped after epoch i.

In this work, we evaluate the computation optimal policy 1-Epoch and show its surprising effectiveness in detecting top-ranked hyperparameter configurations. In addition, we look at the set of trade-offs between computation and predictive performance offered by different early discarding methods among which is the i-Epoch baseline. We do this by spanning different levels of early discarding aggressiveness of each technique. Being more aggressive (i.e., stopping training earlier) reduces computation but also generally sacrifices predictive performance. Therefore, we evaluate the multi-objective optimal frontier, also known as the Pareto-front, achieved by the different early discarding techniques. Ideally, varying the aggressiveness parameters of the different techniques, leads to a large Pareto-front, offering different trade-offs between aggressiveness (training epochs used) and predictive performance.

To simplify our experiments and avoid confounding factors, we do not use advanced HPO solvers but instead perform a random sampling of hyperparameter configurations, for which we can compare several early discarding techniques. We compare i -Epoch to asynchronous successive halving (SHA), parametric learning curve extrapolation (LCE), and the recently introduced LC-PFN model [7] for learning curve extrapolation. We study these techniques in various classification and regression tasks for the class of fully connected deep neural networks.

Against all expectations, our findings are:

1. dynamically allocating resources as done by successive halving or learning curve extrapolation offers minimal (and oftentimes no) utility compared to a constant number of training epochs, and
2. one can often early discard models after only one epoch without losing significant final predictive performance, indicating that perhaps learning curves are more well-behaved than one may expect.

We believe these findings highlight the necessity to incorporate 1-Epoch in future studies since it achieves such good predictive performance for minimal computation while being extremely simple to implement. The software used for our experiments is made publicly available.[1]

2. Related work

Our study focuses on methods, that train only a single model at a time, but keep all checkpoints for further reference. Early discarding means switching to training a model with another HP configuration before attaining the maximum number of epochs allowed. Such strategies are sometimes referred to as "vertical" model selection [3].

One well-known example is Asynchronous Successive Halving [8] (SHA). Hyperband [9] can also be adapted to this setting, which can explore different trade-offs for SHA hyperparameters. Note that since we try different hyperparameters of SHA, Hyperband cannot improve over SHA in terms of the Pareto front, because Hyperband must incur some overhead. After all, it runs multiple versions of SHA inside, which is why it is not included in this comparison.

Learning Curve Extrapolation [10] (LCE) observes early performances and extrapolates them to decide whether training should continue. Learning Curve with Support Vector Regression [11] predicts the final performance based on the configuration and early observations. Learning Curve with Bayesian neural networks [12] instead uses a Bayesian neural network. Trace Aware Knowledge-Gradient [13] leverages an observed curve to update the posterior distribution of a Gaussian process more efficiently. [7] uses a prior-fitted network [14], which is a transformer, to extrapolate learning curves, which is a sped-up and improved version of [10]. [15] extrapolates learning curves using a transformer to larger fidelities to predict the best algorithm from a portfolio, but does not perform regression. [16] uses a purely linear extrapolation, which is a conservative technique that is guaranteed to not prune the optimal candidate given the convexity of the learning curve. The latter, however, is usually not the case for learning curves of neural networks.

FABOLAS [17] uses a similar technique, where correlations are learned in the candidates' ranking between different levels of fidelity. Bayesian Optimization Hyperband [4] embeds Bayesian optimization in Hyperband to sample candidates more efficiently.

Some previous works sometimes implicitly make strong assumptions about the learning curve. For example, methods based on SH or SHA (implicitly) assume that the discarded learning curves will not cross in the future, since only Top-K models are allowed to continue at any given step. In this context, models that start slowly are often discarded. This phenomenon is known as the "short-horizon bias" [18], and this is one of the most pressing reasons to introduce more complex models to deal with learning curves and their possibility of crossing. That is essentially what LCE methods aim to achieve. They either assume a parametric model [7,10], Gaussian process model [13], complex hierarchical Bayesian models [10], or Bayesian neural network models [12] to model the learning curves, to name a few examples. These models can make quite strong assumptions about the learning curves.

It is not clear how often learning curves cross in general [2,19] and what kind of problem this poses for HPO. This work investigates how often curves cross: if curves often cross, 1-epoch cannot perform well, generally, because it would discard too many slow-starting models. Our method further avoids making any assumptions about the learning curves, in the same spirit as SHA. One can see 1-Epoch as SHA to the extreme: where the reduction factor is set in such a way as to prune all models in one go.

Benchmarks play a critical role in the design and development of HPO methods. We have surveyed several recent benchmarks for continuously evolving learning curves, such as HPOBench [20,21], LCBench [22], JAHS-Bench-201 [23], and YAHPO-Gym [24]. In a preliminary version of this study [5], we have already provided visualization and early elimination experiments for these different benchmarks that are consistent with this study. However, as LCBench only had learning curves of 50 epochs and performance estimates on a test set, JAHS-Bench-201 and YAHPO-Gym are using a surrogate model which makes learning curves smoother, we prefer to use actual learning curve data to improve reliability. Therefore, we have chosen to only use learning curves from 4 regression tasks in HPOBench [20,21], and we resort to generating our own learning curves for classification with an experimental setup close to HPOBench.

3. Methods

We consider a function $f(\theta, i) \in \mathbb{R}$ that returns (empirical) generalization error of a deep neural network pipeline configured with hyperparameters $\theta \in \Theta$ (i.e., a vector of mixed variables) after $i \in \mathcal{I}$ training epochs. In our setting we bound the number of training epochs $i_{\min} \leq i \leq i_{\max}$. Next consider a hyperparameter optimization algorithm $a \in \mathcal{A}$ such that $a(f, \Theta, \mathcal{I}) = (y_L, y_I)^T$ where $y_L = f(\theta^*, i_{\max}) \in \mathbb{R}$ is the generalization error of the returned trained deep neural network pipeline configured with hyperparameters θ^* and $y_I \in \mathbb{N}$ is the total number of training epochs used by a to complete the hyperparameter optimization process. Then, the multi-objective problem that hyperparameter optimization with early discarding algorithms aims to solve is:

$$\min_{a \in \mathcal{A}} \quad (y_L, y_I) \tag{1}$$

[1] Code: https://github.com/fmohr/lcdb/blob/ce96fa3768da94d222644883a11403119844f241/publications/2024-neurocom/multi-fidelity-hpo/README.md.

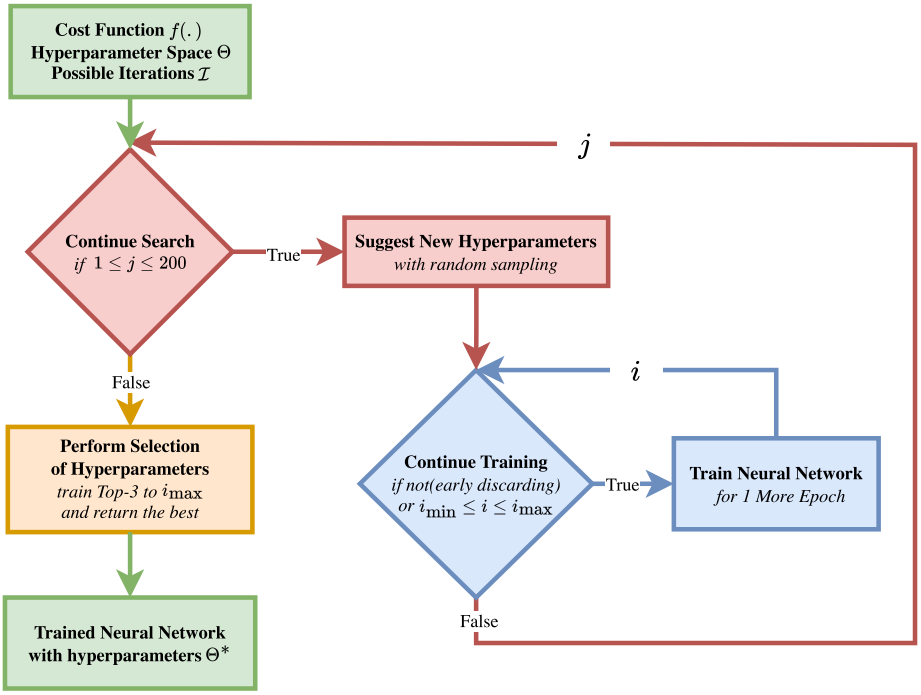


Fig. 1. Hyperparameter optimization and its components including input/output, outer optimization loop exploring new hyperparameter configurations, inner optimization loop incrementally allocating training iterations (what we study in this work) and selection of hyperparameters to return. In *italic* we specify the blocks to match with our experimental study.

s.t. $(y_L, y_I)^T = a(L, \Theta, \mathcal{I})$

In Fig. 1 we provide a flowchart diagram of the hyperparameter optimization with early discarding algorithm class $\mathcal{A}$ that we consider. The HPO process comprises an outer open cycle (red parts), in which an optimizer decides whether optimization should be continued or not. If so, it picks a candidate hyperparameter (HP) configuration (or various if parallelization is supported) for evaluation. Then, the performance of the chosen configurations is computed (blue parts). Since we only consider training of neural networks one can think of the candidate evaluation as an inner cycle in which an empirical learning curve is constructed, with one entry per epoch. In the orange box, a set of final candidates is selected (possibly of size 1) and trained to convergence (if not done already). Among these, the candidate with the best performance is returned and serves as a trained model for predictions.

In the interest of separation of concerns, this paper focuses only on the aspect of early discarding (blue diamond). The other components are fixed as follows: The outer cycle simulates a random search with an evaluation limit of 200 pipelines, which are sampled offline to make sure that all early discarding methods decide upon the same pipelines. Since no model of the performance landscape is built in the random search, the evaluation module simply returns the prediction performance of the network at the time when training is being stopped (no matter whether prematurely or because it has converged). The orange component selects the 3 best configurations found during optimization and trains them to convergence (if not yet). It then returns the best of these models.

This being said, our study focuses only on early discarding techniques for *single candidates* as opposed to *candidate portfolios*. Many popular optimizers consider entire portfolios of candidates, which are then reduced at some predefined ratio [4,25–27]. We are interested in a more flexible class of early discarding techniques that do not need to know all the candidates upfront but decide only upon one candidate at a time based on the score of the best candidate seen so far. This is also referred to as the difference between horizontal optimization (simultaneously growing learning curves of a portfolio) and vertical optimization (evaluating candidates one by one, possibly without even knowing the whole set of candidates to be evaluated) [3].

Among these early discarding techniques for single candidates, we consider three state-of-the-art approaches from different research branches and an approach that simply trains the networks for a previously defined constant number of epochs. First, for the idea of Successive Halving, which is used in many horizontal optimizers [4,25–27], there is a sequential variant [8] that can be used as an independent early discarding module. The second and third approaches discard candidates based on extrapolated learning curves using Monte Carlo Markov Chains (MCMC) [5] and Prior Fitted Networks (PFN) [7], respectively. Another approach for extrapolation, learning curve-based cross-validation (LCCV) [28] with state-of-the-art results in early discarding is not considered in the evaluation because it is based on the assumption of convexity (or concavity) of the learning curves, which is the typical case for sample-wise learning curves but not iteration-wise learning curves as created during the training of a neural network [3].

3.1. Vertical version of successive halving

Successive Halving (SHA) [25] is an *optimization* technique that receives a *set* of candidates, which is successively reduced while granting more resources to candidates that are being retained. A common approach is to eliminate 50% of the candidates and double the amount of resources for the remaining candidates until only one candidate remains; thereby, all iterations consume roughly the same quantity of compute resources.

It is possible to isolate the idea of SHA in order to use it as an early discarding module such as shown in blue in Fig. 1 [8]. To do this, one can test at epoch i if the currently observed score is among the top-$100/r\%$ already observed in the past for other candidates at the same epoch i, where r is called the reduction factor (e.g., $r = 2$ for a reduction of 50%). If this is the case, then the training is continued otherwise it is stopped. This condition is not checked at every training epoch but follows a geometric schedule.

3.2. Parametric learning curve extrapolation with adapted MCMC variant

Parametric Learning Curve Extrapolation (LCE) [10] uses a parametric model to predict the continuation of a learning curve. The parametric functions used for this task are mostly power laws originating from physics research [3]. It is also common to consider linear combinations of such functions [10].

To enable the reconfigurability of greediness, we are interested in *probabilistic extrapolations*. That is, the extrapolation technique should output a *distribution* over learning curves rather than just a single one (usually the likelihood maximizer). These distributions can be obtained by sampling from the posterior distribution, usually using a Bayesian approach [10,12].

However, we found that the above techniques suffer from instabilities, which is why we here use a technique called RoBER (Robust Bayesian Early Rejection) [5]. Instead of considering a linear combination of several parametric models, we only consider one, that is MMF4 which was found to work well in general for extrapolation by [19]. In addition, we do not use a pure Bayesian approach but instead combine optimization with Bayesian inference. That is, first, we fit the parametric model using Levenberg–Marquardt, which minimizes the mean squared error on the observed anchors of the learning curve. Afterward, we use these fitting parameters $\hat{\theta}$ to derive a data-driven prior of the form $\theta \sim N(\hat{\theta}, 1)$. We use a Gaussian likelihood on the observed learning curve anchors with an exponential prior with scale parameter 1. This completely defines the posterior, which is sampled using Markov-Chain-Monte-Carlo. This allows us to sample the distribution of extrapolated values at the largest anchor. We compute this distribution for each currently observed learning curve. If this distribution indicates for a learning curve candidate that we are with probability larger than ρ worse than the current best-observed learning curve value, the candidate is eliminated. The larger ρ, the more conservative: for example if $\rho = 0.9$, a candidate is only discarded if the probability that it under-performs the currently best one at the horizon is greater than or equal to 90%.

3.3. Extrapolation via Prior Fitted Networks (PFN)

Prior Fitted Networks (PFN) are transformer networks that are being trained on synthetic tasks sampled from a so-called prior distribution [29]. For a new task, the PFN does not only output a single prediction for each test point but a *distribution*.

Due to their general nature, PFNs can also be used to predict distributions over learning curves. A recent approach that reports results comparable to or better than the MCMC approach of [10] was presented in [7]. In this approach, synthetic learning curves are sampled from a prior distribution over linear combinations of model classes; a subset of those suggested in [10] is used. The authors of this network offer a pre-trained implementation,[2] which comes with an API that allows extrapolations of learning curves out of the box. Our experiments are based on this implementation.

Regarding LCE, one can define a confidence level ρ and discard candidates only if the probability that the limit performance is worse than the best currently known solution is at least ρ.

3.4. i-Epoch: Constant number of epochs

The last and simplest method is one of a constant number of epochs. In this case, the number of epochs is defined a priori and does not depend on any observations made during the evaluation of the candidate. In our experiments, we consider all numbers of epochs between 1 and 100 as possible limits.

[2] LC-PFN code: https://github.com/automl/lcpfn.

This method is different from the others in that it does not necessarily train *any* model to convergence *during* the evaluation. In all the other approaches, at least one network, namely the one that is believed to be best, is trained until convergence. On the contrary, in the case of a constant number of epochs, even the best network is not (necessarily) trained to convergence during evaluation but only in the final selection phase (orange box in Fig. 1). Of course, if the number of epochs configured is high, it can *implicitly* happen that the networks converge during evaluation. In particular, no early *stopping* (mind the difference to early discarding) is used to stop training once the curve has flattened out, so training can even take more epochs than what would be observed with a standard early stopping approach.

4. Experimental design

Our experiments were designed to answer the following research questions (RQs) for the hyperparameter optimization of deep neural networks:

- **RQ1**: What is the anytime performance of the HPO process (i.e., when stopped at any iteration of the red loop in Fig. 1) when run with the different early discarding techniques for extreme configurations of discarding aggressiveness (i.e., when stopping training at the earliest and at the latest)?
- **RQ2**: For each early discarding technique, what is its Pareto-frontier in terms of (1) final predictive performance (of the selected and trained hyperparameter configuration) and (2) total training epochs consumed in the HPO process, obtained when testing different settings of the method?
- **RQ3**: What does each method contribute to the Pareto frontier resulting from all techniques? This aims to see if methods complement each other in terms of attainable trade-offs and which algorithm offers the most diverse trade-offs.
- **RQ4**: How does 1-Epoch compare to other methods and how can we understand its surprisingly good performance?

Preempting the detailed results, we already summarize at this point that the answers to these questions might be in contrast to the expectations in two ways:

1. While it is clear that 1-Epoch is Pareto-optimal (since one cannot be faster), one would expect that i-Epoch tends to develop a sub-optimal Pareto frontier (compared to other early discarding methods) as i grows. This is because, since i-Epoch does not react to the previous performance observations, there is an increasing risk that (unpromising) candidates are trained unnecessarily long so that the number of total training epochs in the HPO increases without generating any benefit. In other words, for pretty much any $i > i_{\min}$ for some small $i_{\min}$, e.g., 5 or 10, one would expect that there are configurations of the other early discarding methods that Pareto-dominate i-Epoch. The surprising insight of our experiments is that the simple i-Epoch policy is rarely ever Pareto-dominated by any other method.
2. While one would generally expect the maximally aggressive strategy 1-Epoch to deliver significantly sub-optimal results in predictive performance y_L, we show that generally there is little and sometimes *no* possible improvement in predictive performance y_L over the 1-Epoch baseline. In several cases, 1-Epoch is not only Pareto-optimal but strictly optimal.

4.1. Learning curves benchmarks

To be able to generalize conclusions from this work, we answer the questions on several datasets, both regression and classification, which displayed noticeable differences in the observed learning curves. However, we limited our study to the class of fully connected deep

Table 1
Hyperparameter search space for regression benchmarks defined in HPOBench [21,30].

Hyperparameters	Choices
Initial LR	$\{0.0005, 0.001, 0.005, 0.01, 0.05, 0.1\}$
Batch Size	$\{8, 16, 32, 64\}$
LR Schedule	{cosine, fix}
Activation/Layer 1	{relu, tanh}
Activation/Layer 2	{relu, tanh}
Layer 1 Size	$\{16, 32, 64, 128, 256, 512\}$
Layer 2 Size	$\{16, 32, 64, 128, 256, 512\}$
Dropout/Layer 1	$\{0.0, 0.3, 0.6\}$
Dropout/Layer 2	$\{0.0, 0.3, 0.6\}$

neural networks, still including a variety of hyperparameters (e.g., preprocessing, residual connections, regularization).

All learning curves used to benchmark early discarding techniques were computed and stored *prior* to the experimentation. We now describe this generating process. All evaluated deep neural networks are trained for 100 epochs, which fixes $i_{\min} = 1$ and $i_{\max} = 100$. For *regression* tasks, we used an external benchmark of pre-computed learning curves from HPOBench [20,21]. The deep neural networks from this benchmark are similar to ours but were generated from 9 hyperparameters listed in Table 1 and 4 datasets were used.

Datasets were split into 3 folds. The training split was used to optimize the neural network weights for a fixed hyperparameter configuration. The validation split was used to optimize the hyperparameter configurations and serves as an estimate of generalization performance. The test split was used as a final set of data to provide an unbiased report of our results. The data split was 60% for training, 20% for validation, and 20% for testing in the regression tasks, which was dictated by the setup of [20]. In the classification tasks, we chose the split to be 80% for training, 10% for validation, and 10% for testing.

For classification tasks, we generated a set of 1000 randomly sampled hyperparameter configurations from a search space of 17 hyperparameters listed in Table 2. The learning curve generation for each classification task required about 1 h of computation on 400 parallel NVIDIA A100 GPUs on the Polaris Supercomputer at the Argonne Leadership Computing Facility.

For all these configurations we compute the training, validation, and test learning curves by collecting confusion matrices on predictions. Accounting for hyperparameter configurations that resulted in failures (e.g., "`nan`" loss with overflow or underflow) we end up with about 850 correct learning curves for each classification dataset. The diversity of evaluated tasks is provided through the number of samples, features, classes or targets, and the type of features (real or categorical) in Table 3.

4.2. Experimental protocol

As we are interested in evaluating early discarding techniques (blue diamond in Fig. 1) isolated from the process which suggests hyperparameter configurations, we propose the following experimental protocol. The simulated process that suggests hyperparameter configurations (red rectangle in Fig. 1) is a random sampling from the set of precomputed learning curves. This process is fixed by an initial random seed to simulate the same stream of candidate learning curves to different early discarding techniques. The number of search iterations (red diamond in Fig. 1) is fixed to 200 (main constant which makes outcomes of all experiments comparable). Once the (red) loop of 200 candidates is over, the Top-3 models observed are selected and trained to completion if not already done. A model that was not trained to completion during the previous 200 iterations will be retrained from scratch. Of course, these additional training epochs are accounted for in the total number of training epochs used by the method. For example, in 1-Epoch after 200 iterations we select the Top-3 candidates based on the observed scores y_L, we train them to completion so it consumes an additional 3×100 epochs, then we return the best from these 3. For 100-Epoch, as all evaluated models are already trained to completion no additional training is required. The performance we report corresponds to the score reached by "*Method + Top-3*" at any iteration of the search. This corresponds to looking at the "*any-time*" performance of each early discarding method, that is looking at what would be the outcome of the method if we were to stop after k hyperparameter search iterations (red loop) for all $k \in [i_{\min} = 1, i_{\max} = 100]$. A fixed set of 10 random seeds is set to perform 10 repetitions for each method. This protocol ensured that each method was exposed to the same streams of candidates. Therefore the different outcomes observed are only coming from differences in the decisions taken by each method to stop or continue the training.

4.3. Performance indicators

In this section, we describe the two performance indicators of importance in our study. First, we detail the R^2 metric (generalized to both regression and classification) used to assess the predictive performance of evaluated hyperparameter configurations. Then, we detail the hypervolume indicator (HVI) metric used to assess the quality of the solutions for multi-objective optimization.

First, we introduce the coefficient of determination R^2 in the case of regression tasks, where the target prediction is a real value, and, then we extend the notion to the case of classification tasks, where the target prediction is a categorical value in the spirit of [31], also called the Prediction Advantage. This metric is useful as it standardizes both regression and classification similarly which helps us homogenize regression and classification learning curves. A dataset $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ is composed of i.i.d. variables from the joint distribution $P(X, Y)$. In *regression*, the usual definition of R^2 (a.k.a., coefficient of determination) is:

$$R^2 := 1 - \frac{SS_{res}}{SS_{tot}} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}(x_i))^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{2}$$

where SS_{res} is the residual sum of squares, SS_{tot} is the total sum of squares, $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ is the empirical mean of the marginal distribution $P(Y)$ and, $\hat{y}(x_i)$ is a prediction from our model. This definition can also written as:

$$R^2 = 1 - \frac{\frac{1}{n}\sum_{i=1}^{n} L_2(y_i, \hat{y}(x_i))}{\frac{1}{n}\sum_{i=1}^{n} L_2(y_i, \bar{y})} \approx 1 - \frac{E[(Y - E[Y|X])^2|X]}{E[(Y - E[Y])^2]} \tag{3}$$

In the form given by Eq. (3), the expectations $E[Y]$ and $E[Y|X]$ correspond to the optimal Bayes predictors for the squared loss $L_2(Y, \hat{Y}) = (Y - \hat{Y})^2$ respectively on the marginal and conditional distributions. Therefore R^2 corresponds to the normalization of the expected error of the optimal Bayes predictor on the conditional $P(Y|X)$ distribution by the expected error of the optimal Bayes predictor on the marginal distribution $P(Y)$ (a.k.a., constant or "dummy" predictor). In *classification*, we replace the squared-loss with the $0-1$ loss $L_{0\text{-}1}(Y, \hat{Y}) = 1$ if $Y \neq \hat{Y}$ else 0. The optimal Bayes predictor becomes the mode instead of the mean (i.e., the class with the highest probability). We then obtain a new definition of R^2 for classification:

$$R^2 = 1 - \frac{\frac{1}{n}\sum_i L_{0\text{-}1}(y_i, \hat{y}(x_i))}{\frac{1}{n}\sum_i L_{0\text{-}1}(y_i, \dot{y})} \tag{4}$$

where $\dot{y}$ is the mode on the marginal distribution $P(Y)$. This is also known as the Prediction Advantage [31]. For both regression and classification, we have that performance of zero means that the model is as bad as the optimal constant predictor that only uses information from the marginal $P(Y)$ and ignores the input X. If the R^2 is 1 the prediction is "*perfect*" (which also means that there is no presence of random noise on the target). In our study, the goal is to maximize the R^2 score for improved predictive performance which is equivalent to

Table 2
Hyperparameter search space for classification benchmarks.

Hyperparameters	Choices
Activation Function	{none, relu, sigmoid, softmax, softplus, softsign, tanh, selu, elu, exponential}
Activity Regularizer	{none, L1, L2, L1L2}
Batch Normalization	{True, False}
Batch Size	$[1, 512]$ (log-scale)
Bias Regularizer	{none, L1, L2, L1L2}
Dropout Rate	$[0.0, 0.9]$
Kernel Initializer	{random-normal, random-uniform, truncated-normal, zeros, ones, glorot-normal, glorot-uniform, he-normal, he-uniform, orthogonal, variance-scaling}
Kernel Regularizer	{none, L1, L2, L1L2}
Learning Rate	$[10^{-5}, 10^{1}]$ (log-scale)
Number of Layers	$[1, 20]$
Number of Units	$[1, 200]$ (log-scale)
Optimizer	{SGD, RMSprop, Adam, Adadelta, Adagrad, Adamax, Nadam, Ftrl}
Regularizer Factor	$[0.0, 1.0]$
Shuffle Each Epoch	{True, False}
Skip Connections	{True, False}
Transform Categorical	{onehot, ordinal}
Transform Real	{minmax, std, none}

Table 3
Characteristics of datasets used for our experiments. On Top, the 4 datasets were used for regression, and on the bottom, the 6 datasets were used for classification. The datasets are sorted by decreasing number of samples.

Dataset (OpenML-Id)	#Features	#Samples	#Classes or #Targets	Real Features	Categorical Features
Slice Localization (42973)	380	53,500	1	True	False
Protein Structure (44963)	9	45,730	1	True	False
Naval Propulsion (44969)	14	11,934	1	True	False
Parkinson's Telemonitoring (4531)	20	5875	2	True	True
MNIST (554)	784	70,000	10	True	False
Australian Electricity Market (151)	8	45,312	2	True	True
Bank Marketing (1461)	16	45,211	2	True	True
Letter Recognition (6)	16	20,000	26	True	False
Letter Speech Recognition (300)	617	7797	26	True	False
Robot Navigation (1497)	24	5456	4	True	False
Chess End-Game (3)	36	3196	2	False	True
Multiple Features (Karhunen) (14)	76	2000	10	True	False
Multiple Features (Fourier) (16)	64	2000	10	True	False
Steel Plates Faults (40982)	27	1941	7	True	False
QSAR Biodegradation (1494)	41	1055	2	True	False
German Credit (31)	20	1000	2	True	True
Blood Transfusion (1464)	4	748	2	True	False

minimizing $y_L := 1 - R^2(\theta, i_{\max})$ in Eq. (1) (replacing the $\mathcal{L}$ by our R^2 score).

Now that we have discussed the performance indicator for prediction we will present the metric used to assess the quality of multi-objective optimization (MOO). For the sake of brevity, we will not recall the formal definitions related to the notion of Pareto-optimality in MOO. However, shortly we recall that Pareto-Front refers to the solution set in the objective space (i.e., 2-dimensional in our case as we have 2 objectives y_L and y_I). As these objectives are (supposedly) conflicting, y_L the predictive performance and y_I the total number of training iterations used, the Pareto-Front is a one-dimensional space (i.e., a line) unless the problem is "degenerated", meaning there is no real conflict between objectives and the solution set is therefore containing a single point. Among the possible metrics used in MOO [32] and as we do not know the true Pareto-Front of our problem we decide to use the hypervolume indicator (HVI). As we are in 2-D it corresponds to measuring the area defined by an estimated Pareto-Front and a reference point (fixed for all experiments on the same dataset). The HVI is compliant with the notion of Pareto-optimality and also known to measure the compare the diversity of solutions (i.e., trade-offs) between different Pareto-Fronts. In our study, the goal is to identify the early discarding technique which maximizes the Hypervolume indicator when evaluated at different levels of aggressiveness.

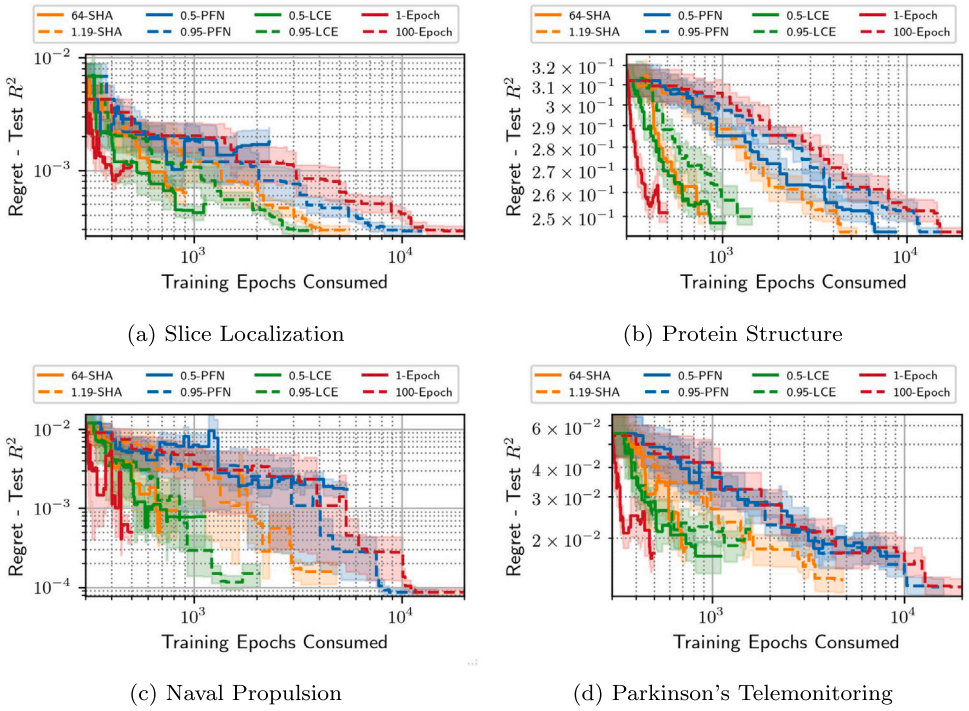
5. Results

In this section, we present the results that helped us answer the research questions introduced in Section 4.

5.1. RQ1 — *What is the anytime performance of early discarding techniques?*

To understand the anytime performance of early discarding techniques we plot the $1 - R^2$ *test* performance as a function of overall training epochs realized so far. That is, a curve that passes the point (t, l) in the plot means that the test score of the model that *would* have been picked if the HPO process had stopped after t total training epochs would have been l. This type of performance curve weighs training epochs equally for all hyperparameter configurations, which may be deceiving since they can vary in computational cost (e.g., large and small neural networks). Still, it is a convenient simple method abstracting from implementation details. We present the performance curves in Figs. 2 and 3 for classification and regression respectively.

The most important insight from the plots is that the sensitivity of the early discarding techniques with respect to their aggressiveness parameter varies a lot. While the i-Epoch and r-SHA algorithms are very sensitive to the aggressiveness (as expected), the learning curve extrapolation-based methods (i.e., ρ-LCE and ρ-PFN) are surprisingly



(a) Slice Localization

(b) Protein Structure

(c) Naval Propulsion

(d) Parkinson's Telemonitoring

Fig. 2. Comparing the any-time performance of various early discarding techniques during a random search (mean and one standard error over 10 repetitions) of 200 iterations (4 regression tasks). The two baseline strategies 1-Epoch and 100-Epoch method bound the trade-offs that can be achieved. The predictive performance of 1-Epoch is at least of the same order of magnitude as other strategies while consuming a significantly smaller (the minimum in training epochs) number of training epochs.

less sensitive to aggressiveness parameter ρ. This can be observed especially on the set of classification tasks shown in Fig. 3. In other words, for ρ-LCE and ρ-PFN, it almost makes no difference in consumed training epochs whether the user requires almost certainty ($\rho = 0.95$) or whether the certainty is just as good as a coin flip ($\rho = 0.5$). This could indicate that the models express too little uncertainty about the extrapolated learning curve.

Another observation is that the ρ-PFN method hardly reduces the overall training epochs used by 100-Epoch as can be seen for all datasets. It means that the learning curve extrapolation of this method is probably over-optimistic. It even seems to perform worse than 100-Epoch for both predictive performance and overall training epochs used on learning curves which are very noisy and increasing. These failures can be observed in Figs. 3(c), 3(l) and 3(m).

A third observation is that ρ-LCE, while being a more robust version of LCE, can still under-perform predictive performance even when being set to be conservative ($\rho = 0.95$). This can be seen in Figs. 2(d), 3(d), 3(h) and 3(k). This confirms our belief that such models express too little uncertainty about the extrapolation.

From the practical viewpoint, no utopia method has yet been found. A utopia method would achieve a strict and consistent dominance compared to the 100-Epoch baseline. That is a method that achieves, on all tasks, better predictive performance while being faster than the base full training evaluation. Such a method seems not to exist currently and may not exist if both objectives y_L and y_I are truly conflicting.

Finally, the presented performance curve plots also show the importance of considering the 1-Epoch baseline to contextualize results and avoid an overly optimistic presentation of the methods. Without the solid red line which corresponds to 1-Epoch, ρ-LCE might appear a quite dominant approach in this experimental setting. While it is true that learning curve extrapolation-based methods are very convincing in many cases, there are some datasets, such as Protein Structure (Fig. 2(b)), Parkinson's Telemonitoring (Fig. 2(d)), MNIST (Fig. 3(a)), QSAR-Biodegradation (Fig. 3(j)), or German Credit (Fig. 3(l)), in which the 1-Epoch baseline can reduce the number of epochs of LCE again by about 50% without losing significant or any predictive performance.

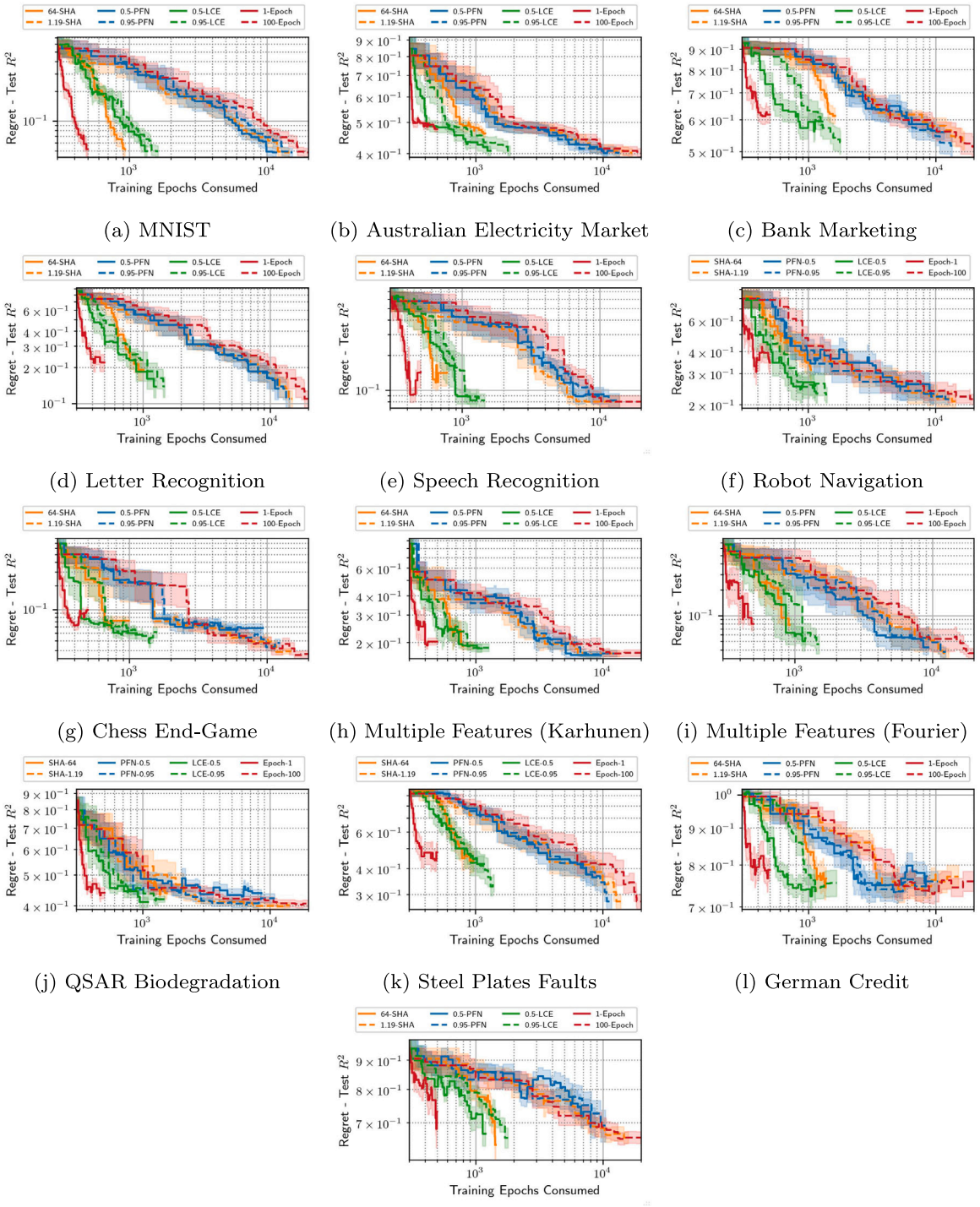
5.2. RQ2 — Multi-objective trade-offs and pareto-fronts

While the previous question only considers two extreme configurations to understand the sensitivity of the HPO process with respect to the aggressiveness of the early discarding technique, we now want to better understand the actual trade-offs that each method can span. At this point, we no longer look at any-time performance but instead, we look at the final predictive performance and overall consumed training epochs for one aggressiveness setting. Once all methods and all aggressiveness levels are collected we compute the Pareto-Front of each early discarding method which does not always contain all evaluated points.

From the results presented in the previous section, we already know that the Pareto-Fronts of ρ-PFN will be strictly dominated by other techniques (i.e., the area/hypervolume it defines will be strictly included in the area of other methods). Since even the difference between minimum ($\rho = 0.95$) and maximum aggressiveness ($\rho = 0.5$) had only minimal effect, one expects the area covered by ρ-PFN in the multi-objective profile to be narrow.

The multi-objective profiles and the corresponding Pareto-Fronts are presented in Figs. 4 and 5. For i-Epoch, a value was computed for each $1 \leq i \leq 100$. For ρ-LCE and ρ-PFN, we used values of $\rho \in \{0.5, 0.7, 0.8, 0.9, 0.95\}$, and for r-SHA we used values of $r \in \{\sqrt{\sqrt{2}} = 1.19, \sqrt{2} = 1.41, 2, 4, 8, 16, 32, 64\}$. For each approach, the Pareto-optimal points are connected with a step function to indicate the respective Pareto frontier. The shaded areas show the hypervolume of each approach.

Some plots, like in Fig. 4(a), suggest a certain inconsistency in the trade-off logic of i-Epoch in the sense that many points of a single method do not lie on the same method's Pareto frontier. However, this can often be attributed to noise on rather small scales. For example, in the mentioned plot, differences are on a scale below 10^{-3}, i.e., less than 0.1% difference in performance in terms of the constant predictor baseline. For the other methods, this effect is less pronounced or does not occur because much fewer points are generated and the change in



(a) MNIST (b) Australian Electricity Market (c) Bank Marketing

(d) Letter Recognition (e) Speech Recognition (f) Robot Navigation

(g) Chess End-Game (h) Multiple Features (Karhunen) (i) Multiple Features (Fourier)

(j) QSAR Biodegradation (k) Steel Plates Faults (l) German Credit

(m) Blood Transfusion

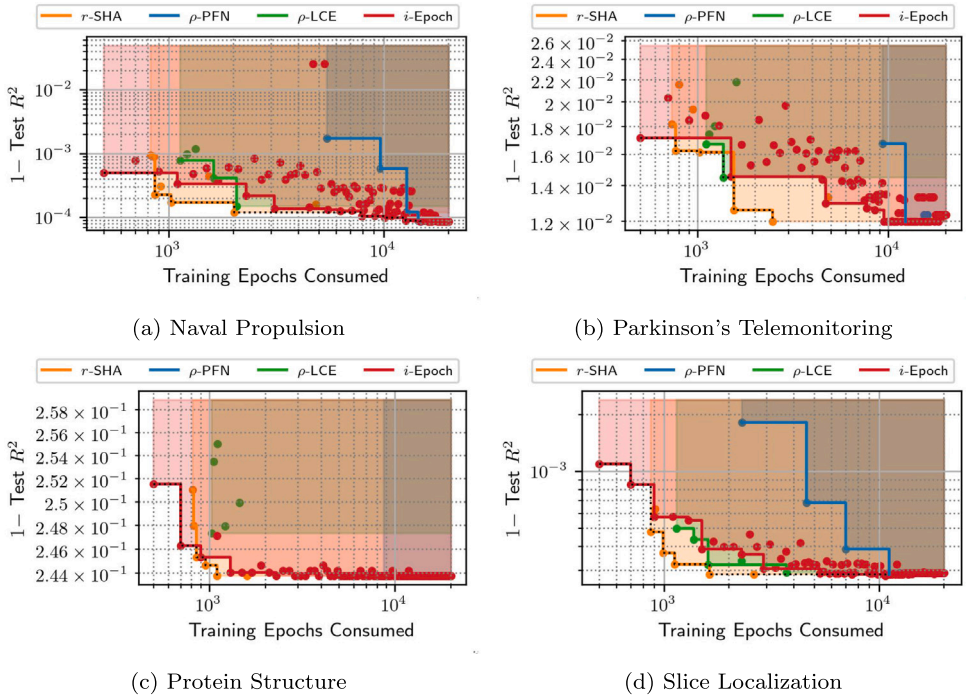
Fig. 3. Comparing the any-time performance of various early discarding techniques during a random search (mean and one standard error over 10 repetitions) of 200 iterations (on 13 classification tasks).

aggressiveness is more significant (10%-steps in the case of ρ compared to single epochs in the case of i-Epoch).

The first observation confirms our expectation that learning curve extrapolation-based techniques offer little diversity of trade-offs. ρ-LCE, no matter how aggressiveness is configured, tends to use about 10x less training epoch than 100-Epoch while sometimes slightly underperforming in attained predictive performance. And, again, PFN on most datasets offers almost no reductions regardless of the configuration of ρ.

5.3. RQ3 — *Which methods offer diverse trade-offs?*

To quantity the observation that i-Epoch offers a more diverse set of trade-offs we compute the relative hypervolume spanned by each method in Figs. 4 and 5. To evaluate the hypervolume we set as reference point $y_{\text{ref}} := (\max \mu_L + \sigma_L^{\text{err}}, \max \mu_B + \sigma_B^{\text{err}})$ (i.e., element-wise upper-bound of observations) for all methods. Then we apply a $\log_{10}(.)$ transformations on both y_L and y_I values (including the reference point). This transformation serves to spread the volume contributed by



(a) Naval Propulsion

(b) Parkinson's Telemonitoring

(c) Protein Structure

(d) Slice Localization

Fig. 4. Multi-objective profiles built from spanning various levels of aggressiveness of early discarding methods (on 4 regression tasks). The estimated Pareto-Front of each method is shown in a plane line. The black dotted line corresponds to the estimated Pareto-Front including the methods altogether. It can be seen that the i-Epoch strategy spans more trade-offs (larger area) than other methods while never being significantly dominated.

small and large values equally. Otherwise, differences in hypervolume would become unnoticeable as soon as improvements in y_L or y_I become orders of magnitude smaller than the largest reference point values. Finally, we compute the hypervolume of all methods which we divide the hypervolume of the Pareto-Front considering all observations (in dotted black line). This relative hypervolume then quantifies how much each method contributes to the available set of trade-offs that we observed. The closer is the value to 1 the more complete the method. The resulting scores are presented in Table 4.

As it can be observed i-Epoch achieves the highest scores on all but one task giving it an average rank of 1.125. The second best-ranked method is ρ-SHA followed by ρ-LCE. The ρ-PFN method consistently finishes last ranked on all tested tasks. Lastly, we also notice that relative hypervolume scores of i-Epoch are often close to 1 which confirms that this method spans most of the observed trade-offs and it is never significantly outperformed in either objective.

5.4. RQ4 — Is 1-Epoch so good and if so, how?

Last but not least, throughout our presented results we can notice the unreasonable effectiveness of 1-Epoch. Despite sometimes being noisier in its performance profiles such as in Figs. 2(c), 5(f) and, 5(l), it always achieved better any-time performance than other early discarding methods. This is demonstrated by the fact that its performance curve does not cross with the performance curves of other methods. However, the difference in final predictive performance y_L can sometimes be statistically significant such as in Figs. 3(c), 3(g), 3(i) and, 3(k) which confirms the trade-off between the two objectives.

How is it possible this approach can perform so well? To better understand this, we analyze the learning curves of our experiments. In Figs. 6 and 7 we display 500 randomly sampled learning curves from our pre-computed sets, we then color the curves by their ranking at 100 epochs (the maximum number of training epochs). Low ranks, colored in light blue, correspond to the best models, while high ranks, colored in red and then yellow, correspond to the worst models. We plot the performance of the constant predictor as a dashed lime line and also plot its rank.

In these plots, it can be observed that for all benchmarks there exists among the best models some that are also the best early in the training process. This observation explains the performance of 1-Epoch. Then, in a few cases, we can observe a significant proportion of models perform worse than the constant predictor. It is about 33% of models in Fig. 6(a) and about 80% of models in 7(g) to 7(i). Finally, it seems that learning curve oscillations are correlated with the final predictive performance. The best models present much less oscillations than the worst models, which justifies high aggressiveness in the early discarding method.

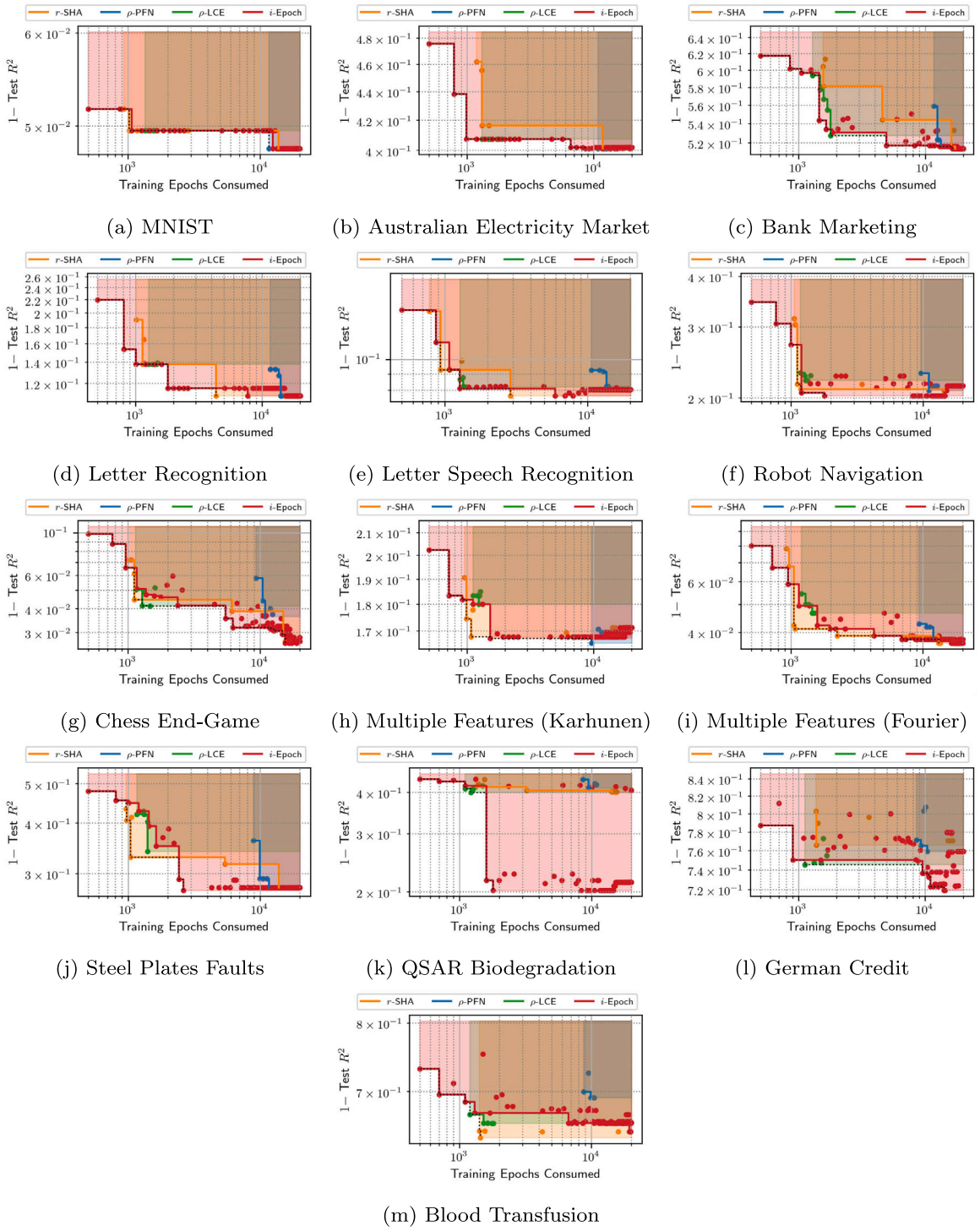
6. Conclusion

In this paper, we conducted a comprehensive analysis of early discarding techniques for hyperparameter optimization of fully connected deep neural networks. Our study rigorously compared an array of advanced techniques and unveiled intriguing findings: (1) the unreasonable effectiveness of the 1-Epoch strategy, a straightforward yet previously overlooked baseline method, and (2) the Pareto-dominance of the i-Epoch strategy despite its simplicity.

We attribute the success of this strategy to effectively differentiating between high and low-potential models in the early stages of training. Notably, models with promising prospects exhibit minimal performance oscillations, a pattern consistently observed in widely used benchmarks. These insights not only underscore the importance of incorporating the i-Epoch strategy in future benchmark analyses but also highlight the potential necessity of considering the multi-objective problem hidden behind early discarding strategies. An early discarding method would bring significant value only if it complements or dominates the i-Epoch Pareto-Front. Current early discarding approaches only add minimal or no utility in this sense.

Besides its good performance, we believe that 1-Epoch's simplicity is valuable in itself. Besides being easy to implement, before execution, it is easy to predict the number of training epochs consumed by i-Epoch for any i when it is not possible for either ρ-LCE or r-SHA. This makes i-Epoch practically attractive.

To be noted, our work is limited to using "epoch" as iteration units for early discarding. While this is convenient and appealing to conduct



(a) MNIST (b) Australian Electricity Market (c) Bank Marketing

(d) Letter Recognition (e) Letter Speech Recognition (f) Robot Navigation

(g) Chess End-Game (h) Multiple Features (Karhunen) (i) Multiple Features (Fourier)

(j) Steel Plates Faults (k) QSAR Biodegradation (l) German Credit

(m) Blood Transfusion

Fig. 5. Multi-objective profiles built from spanning various levels of aggressiveness of early discarding methods (13 classification tasks). The estimated Pareto-Front of each method is shown in a plane line. The black dotted line corresponds to the estimated Pareto-Front including the methods altogether. It can be seen that the *i*-Epoch strategy spans more trade-offs (larger area) than other methods while never being significantly dominated.

studies independent of hardware implementation considerations, practical application settings may require considering wall time or other options as units for early discarding. In particular, since different configurations may have different batch sizes, some configurations could be much faster to train than others. However, comparing wall-clock time is extremely hardware and software implementation dependent. Maybe considering the size of the deep neural network as a third objective of Eq. (1) could be an improvement.

We have tried a limited range for the aggressiveness parameters of ρ-LCE and r-SHA. Their Pareto-Front could be larger and more dominant for a wider range of parameters considered. However, values of $\rho < 0.5$ seem relatively strange for ρ-LCE because in that case it will be very pessimistic about extrapolated performance and discard models as soon as there is a small probability of under-performing. r-SHA could be more aggressive but it should be noted that our largest reduction factor of 64 corresponds to continuing training only if the model is in

Table 4
Relative hypervolumes of each early discarding technique with respect to the hypervolume including all the techniques. Bold and green is best, followed by yellow, orange and red. These scores assess the diversity of trade-offs, in consumed training epochs and predictive performance, offered by each technique among all observed outcomes. The higher the score the more complete (in terms of possible trade-offs) is the early discarding technique. In our experiments, the i-Epoch technique offers the best set of trade-offs and achieves a trade-off close to 1 indicating optimality amongst all methods.

Dataset	r-SHA	ρ-PFN	ρ-LCE	i-Epoch
Slice Localization	0.930	0.401	0.823	**0.932**
Protein Structure	0.916	0.241	0.652	**0.989**
Naval Propulsion	0.881	0.280	0.742	**0.951**
Parkinson's Telemonitoring	**0.930**	0.201	0.667	0.880
MNIST	0.858	0.176	0.743	**0.994**
Australian Electricity Market	0.768	0.205	0.829	**1.000**
Bank Marketing	0.609	0.184	0.847	**0.989**
Letter Recognition	0.851	0.169	0.672	**0.988**
Letter Speech Recognition	0.915	0.175	0.810	**0.974**
Robot Navigation	0.882	0.218	0.789	**0.992**
Chess End-Game	0.866	0.233	0.827	**0.965**
Multiple Features (Karhunen)	0.901	0.231	0.606	**0.955**
Multiple Features (Fourier)	0.936	0.239	0.697	**0.951**
Steel Plates Faults	0.806	0.257	0.644	**0.923**
QSAR Biodegradation	0.141	0.037	0.176	**0.993**
German Credit	0.585	0.198	0.800	**0.970**
Blood Transfusion	0.811	0.167	0.753	**0.856**
Average Rank	2.029	4.000	2.846	**1.125**

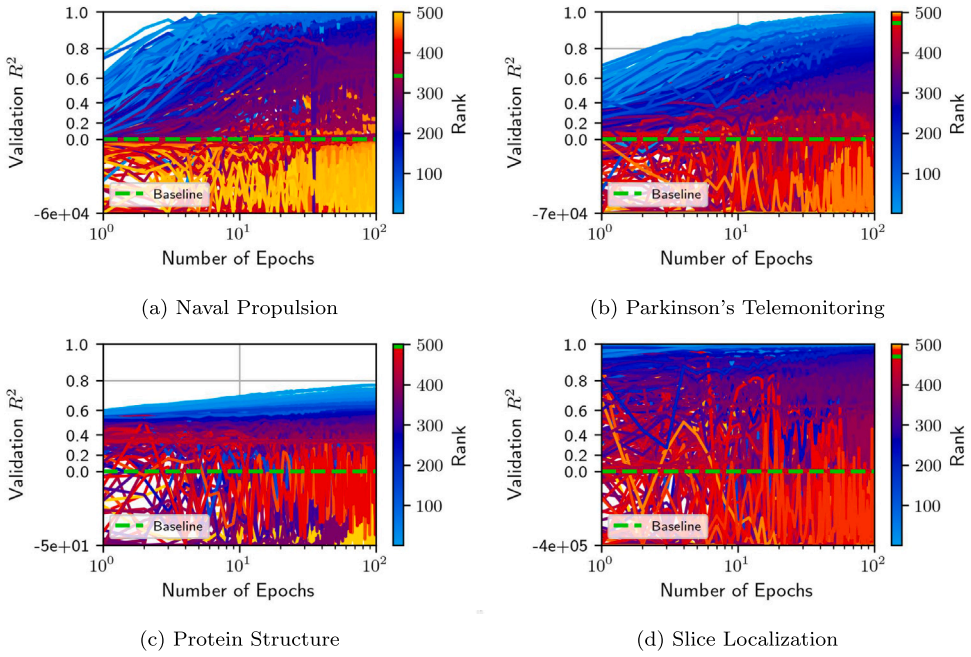
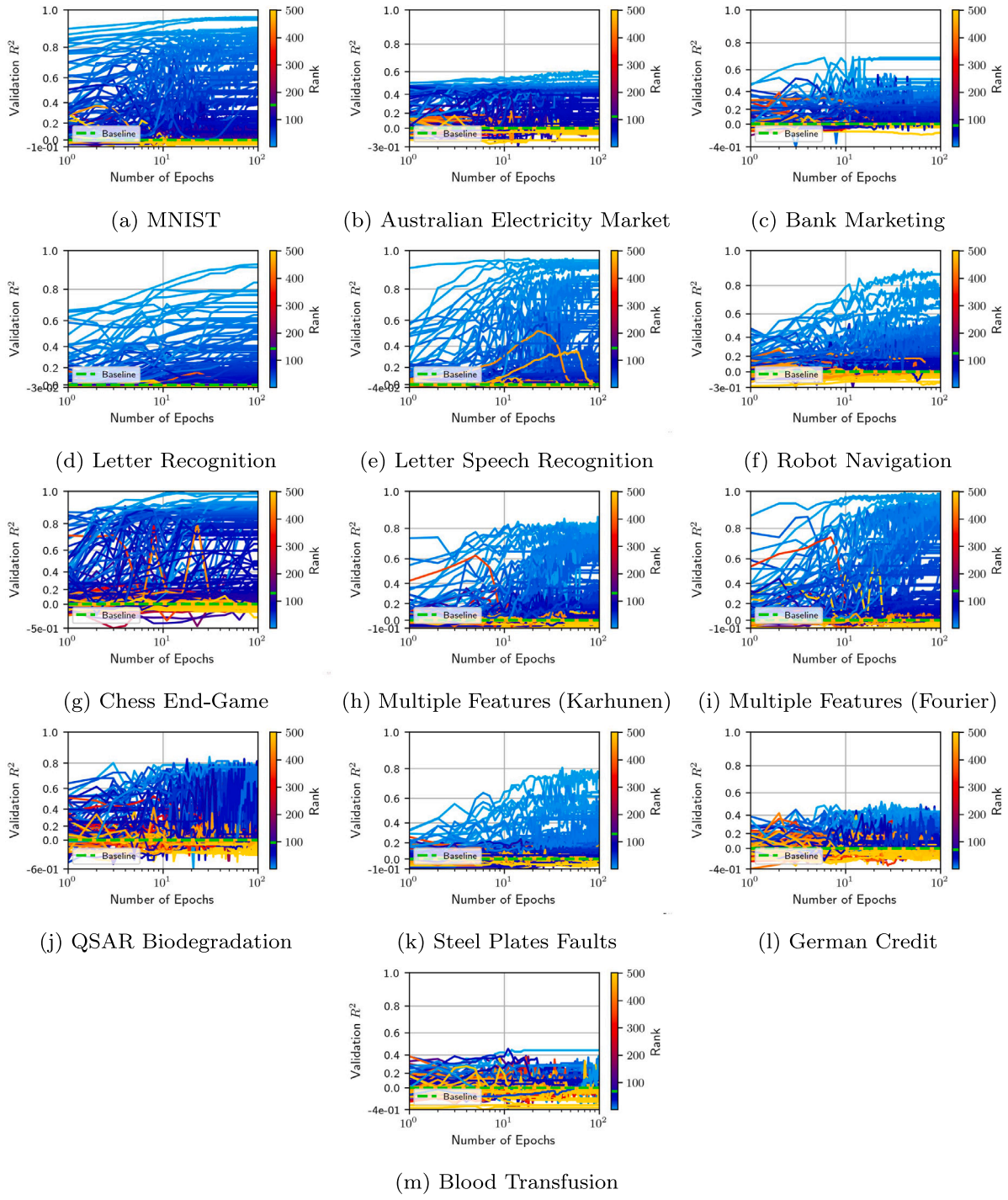


Fig. 6. Visualizing the final ranking for good (light blue) and bad (yellow) models for 500 randomly sampled learning curves (on 4 regression tasks). The constant predictor performance (at 0) is shown as a green dashed line. Models can be selected from the first epoch as there appear to be dominant models early on in the training epochs.

the current Top-1.5% meaning comparing to the single best model after 100 Hyperparameter suggestions and Top-3 of 200. Also, this value is significantly larger than the suggested default value of 4 in the original paper [8].

Also, we studied the early discarding methods in combination with a random search. In other words, HPO is often combined with techniques that suggest candidates through more sophisticated methods, such as Bayesian optimization or portfolio [4,25–27]. However, for such approaches we cannot quantify the computational cost as easily through the number of epochs, as the Bayesian optimization may not be a neural net. Besides that, the comparison becomes more complicated, because the different components (configuration proposer, early discarding technique, etc.) may interact in unexpected ways. Therefore, such a comparison is out of the scope.

To come back to the question of the earlier work: is one epoch all you need? We think the answer remains to be seen, in particular, we think that the 1-epoch approach can be even pushed further. During the first epoch, more information is available for making decisions. For example, the loss per batch could be collected. This again forms a curve of performances versus the number of batches processed, which seems conceptually similar to a learning curve. This curve could be extrapolated as well. This will allow us to make potentially better decisions after 1 epoch or even training could be stopped before finishing one epoch. The latter could be especially promising for large language models, for which one epoch of training can already consume hours of training time.



(a) MNIST (b) Australian Electricity Market (c) Bank Marketing

(d) Letter Recognition (e) Letter Speech Recognition (f) Robot Navigation

(g) Chess End-Game (h) Multiple Features (Karhunen) (i) Multiple Features (Fourier)

(j) QSAR Biodegradation (k) Steel Plates Faults (l) German Credit

(m) Blood Transfusion

Fig. 7. Visualizing the final ranking for good (light blue) and bad (yellow) models for 500 randomly sampled learning curves (on 13 classification tasks). The constant predictor performance (at 0) is shown as a green dashed line. Models can be selected from the first epoch as there appear to be dominant models early on in the training epochs.

CRediT authorship contribution statement

Romain Egele: Writing – review & editing, Writing – original draft, Visualization, Supervision, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Felix Mohr:** Writing – review & editing, Writing – original draft, Software, Methodology, Investigation. **Tom Viering:** Writing – review & editing, Writing – original draft, Software, Methodology, Investigation. **Prasanna Balaprakash:** Supervision, Methodology, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The code to generate custom data is publicly shared. Other sources of data can be downloaded through the publicly shared code.

Acknowledgments

We would like to thank Prof. Isabelle Guyon for participating with us in discussions that improved the quality of this work.

This material is based upon work supported by the U.S. Department of Energy (DOE), Office of Science, Office of Advanced Scientific Computing Research, under Contract DE-AC02-06CH11357. This research used resources from the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility. This material is based upon work supported by ANR Chair of Artificial Intelligence HUMANIA ANR-19-CHIA-0022 and TAILOR EU Horizon 2020 grant 952215. Felix Mohr participated through the project ING-312-2023 from Universidad de La Sabana, Campus del Puente del Común, Km. 7, Autopista Norte de Bogotá. Chía, Cundinamarca, Colombia.

References

[1] T. Yu, H. Zhu, Hyper-parameter optimization: A review of algorithms and applications, 2020, arXiv preprint arXiv:2003.05689.
[2] T. Viering, M. Loog, The shape of learning curves: a review, IEEE Trans. Pattern Anal. Mach. Intell. (2022).
[3] F. Mohr, J.N. van Rijn, Learning curves for decision making in supervised machine learning–A survey, 2022, arXiv preprint arXiv:2201.12150.
[4] S. Falkner, A. Klein, F. Hutter, BOHB: Robust and efficient hyperparameter optimization at scale, in: International Conference on Machine Learning, PMLR, 2018, pp. 1437–1446.
[5] R. Egele, I. Guyon, Y. Sun, P. Balaprakash, Is one epoch all you need for multi-fidelity hyperparameter optimization? 2023, arXiv preprint arXiv:2307.15422.
[6] S. Falkner, A. Klein, F. Hutter, PASHA: Efficient HPO and NAS with progressive resource allocation, in: The Eleventh International Conference on Learning Representations, 2023, URL https://openreview.net/forum?id=syfgJE6nFRW.
[7] S. Adriaensen, H. Rakotoarison, S. Müller, F. Hutter, Efficient bayesian learning curve extrapolation using prior-data fitted networks, 2023, arXiv preprint arXiv:2310.20447.
[8] L. Li, K. Jamieson, A. Rostamizadeh, E. Gonina, J. Ben-Tzur, M. Hardt, B. Recht, A. Talwalkar, A system for massively parallel hyperparameter tuning, Proc. Mach. Learn. Syst. 2 (2020) 230–246.
[9] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, A. Talwalkar, Hyperband: A novel bandit-based approach to hyperparameter optimization, J. Mach. Learn. Res. 18 (1) (2017) 6765–6816.
[10] T. Domhan, J.T. Springenberg, F. Hutter, Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves, in: Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.
[11] B. Baker, O. Gupta, R. Raskar, N. Naik, Accelerating neural architecture search using performance prediction, 2017, arXiv preprint arXiv:1705.10823.
[12] A. Klein, S. Falkner, J.T. Springenberg, F. Hutter, Learning curve prediction with Bayesian neural networks, in: International Conference on Learning Representations, 2017.
[13] J. Wu, S. Toscano-Palmerin, P.I. Frazier, A.G. Wilson, Practical multi-fidelity Bayesian optimization for hyperparameter tuning, in: Uncertainty in Artificial Intelligence, PMLR, 2020, pp. 788–798.
[14] N. Hollmann, S. Müller, K. Eggensperger, F. Hutter, Tabpfn: A transformer that solves small tabular classification problems in a second, 2022, arXiv preprint arXiv:2207.01848.
[15] T. Ruhkopf, A. Mohan, D. Deng, A. Tornede, F. Hutter, M. Lindauer, MASIF: Meta-learned algorithm selection using implicit fidelity information, Trans. Mach. Learn. Res. (2022).
[16] F. Mohr, J.N. van Rijn, Fast and informative model selection using learning curve cross-validation, IEEE Trans. Pattern Anal. Mach. Intell. (2023).
[17] A. Klein, S. Falkner, S. Bartels, P. Hennig, F. Hutter, Fast bayesian optimization of machine learning hyperparameters on large datasets, in: Artificial Intelligence and Statistics, PMLR, 2017, pp. 528–536.
[18] Y. Wu, M. Ren, R. Liao, R. Grosse, Understanding short-horizon bias in stochastic meta-optimization, in: International Conference on Learning Representations, 2018, URL https://openreview.net/forum?id=H1MczcgR-.
[19] F. Mohr, T.J. Viering, M. Loog, J.N. van Rijn, LCDB 1.0: An extensive learning curves database for classification tasks, in: Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2022, Grenoble, France, September 19–23, 2022, Proceedings, Part V, Springer, 2023, pp. 3–19.
[20] K. Eggensperger, P. Müller, N. Mallik, M. Feurer, R. Sass, A. Klein, N. Awad, M. Lindauer, F. Hutter, Hpobench: A collection of reproducible multi-fidelity benchmark problems for HPO, 2021, arXiv preprint arXiv:2109.06716.
[21] A. Klein, F. Hutter, Tabular benchmarks for joint architecture and hyperparameter optimization, 2019, arXiv preprint arXiv:1905.04970.
[22] L. Zimmer, M. Lindauer, F. Hutter, Auto-pytorch: Multi-fidelity metalearning for efficient and robust autodl, IEEE Trans. Pattern Anal. Mach. Intell. 43 (9) (2021) 3079–3090.
[23] A. Bansal, D. Stoll, M. Janowski, A. Zela, F. Hutter, JAHS-bench-201: A foundation for research on joint architecture and hyperparameter search, in: Thirty-Sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track, 2022.
[24] F. Pfisterer, L. Schneider, J. Moosbauer, M. Binder, B. Bischl, Yahpo gym-an efficient multi-objective multi-fidelity benchmark for hyperparameter optimization, in: International Conference on Automated Machine Learning, PMLR, 2022, pp. 3/1–39.
[25] K. Jamieson, A. Talwalkar, Non-stochastic best arm identification and hyperparameter optimization, in: Artificial Intelligence and Statistics, PMLR, 2016, pp. 240–248.
[26] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, A. Talwalkar, Hyperband: A novel bandit-based approach to hyperparameter optimization, arXiv:1603.06560[cs,stat], URL http://arxiv.org/abs/1603.06560.
[27] N. Awad, N. Mallik, F. Hutter, DEHB: Evolutionary hyberband for scalable, robust and efficient hyperparameter optimization, in: Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, International Joint Conferences on Artificial Intelligence Organization, pp. 2147–2153, http://dx.doi.org/10.24963/ijcai.2021/296, URL https://www.ijcai.org/proceedings/2021/296.
[28] F. Mohr, J.N. van Rijn, Fast and informative model selection using learning curve cross-validation, arXiv:2111.13914[cs], URL http://arxiv.org/abs/2111.13914.
[29] S. Müller, N. Hollmann, S.P. Arango, J. Grabocka, F. Hutter, Transformers can do bayesian inference, 2021, arXiv preprint arXiv:2112.10510.
[30] K. Eggensperger, P. Müller, N. Mallik, M. Feurer, R. Sass, A. Klein, N. Awad, M. Lindauer, F. Hutter, HPOBench: A collection of reproducible multi-fidelity benchmark problems for HPO, arXiv:2109.06716[cs], URL http://arxiv.org/abs/2109.06716.
[31] R. El-Yaniv, Y. Geifman, Y. Wiener, The prediction advantage: A universally meaningful performance measure for classification and regression, 2017, arXiv preprint arXiv:1705.08499.
[32] C. Audet, J. Bigeon, D. Cartier, S. Le Digabel, L. Salomon, Performance indicators in multiobjective optimization, European J. Oper. Res. 292 (2) (2021) 397–422.