On the Difficulty of DNN Hyperparameter Optimization Using Learning Curve Prediction

Daeyoung Choi, Hyunghun Cho, and Wonjong Rhee, *Fellow, IEEE*
Department of Transdisciplinary Studies
Seoul National University
Seoul, South Korea
{choid, webofthink, wrhee}@snu.ac.kr

***Abstract*—With the recent success of deep learning on a variety of applications, efficiently tuning hyperparameters of Deep Neural Networks (DNNs) with less effort has become a timely and practical topic. As an algorithmic solution, automatic hyperparameter optimization methods like Bayesian optimization have gained popularity for achieving human-comparable or even human-surpassing performance. To further speed up hyperparameter optimization, learning curves of DNNs can be predicted and used to early terminate the training phase of the chosen hyperparameter setting when the expected training performance is not satisfactory. While the previous studies show promising results, it is still unclear if an effective general rule can be derived for a broad spectrum of DNN hyperparameter optimization problems. In this work, we consider hyperparameter optimization of MNIST and CIFAR-10, and for each task, we analyze the characteristics of the 20,000 learning curves that correspond to the 20,000 different hyperparameter configurations. By investigating a large number of learning curves for a given task, we find that the characteristics of learning curve shapes can drastically change depending on the choice and range of hyperparameters. Therefore, utilizing learning curves for speed improvement is not a simple task and can be dependent on many factors. Based on the observations and analyses on the 20,000 learning curves, we design two early termination rules, ETR-1 and ETR-2, and show that the rules can be beneficial in the best case but can be harmful as well. Our observations and experimental results highlight that hyperparameter optimization of DNNs using learning curve prediction is challenging. In particular, the results of recent studies that are based on at most thousands of learning curves of a limited number of tasks should be carefully interpreted depending on the task, DNN model, hyperparameter choice, and hyperparameter range.**

***Index Terms*—deep neural networks, hyperparameter optimization, learning curve prediction, early termination**

I. Introduction

For traditional machine learning, feature engineering is usually necessary for achieving high performance. However, feature engineering is typically labor-intensive and can be performed well only by experts. In contrast, Deep Neural Networks (DNNs) can automatically capture task-relevant information in the data and significantly reduce the effort needed for feature engineering. As a result, deep learning has achieved promising results for many applications in recent years. Deep learning makes less use of feature engineering for improving task performance but requires careful Hyperparameter Optimization (HPO) instead. This is because DNNs usually have many more hyperparameters than traditional machine learning algorithms, making the task more sophisticated to approach with feature engineering alone and because its performance can be highly dependent on the choice of hyperparameters.

HPO of DNNs can be considered as a problem of finding an optimal hyperparameter configuration $\mathbf{x}^*$ of DNN performance which can be regarded as a black-box function f. That is, we can define the problem of choosing an optimal set of hyperparameters as $\mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$, where DNN hyperparameter configuration $\mathbf{x}$ is in hyperparameter space $\mathcal{X} \subset \mathbb{R}^d$ for real-valued hyperparameters (extension is obvious for categorical hyperparameters). In the real world, what practitioners typically do is to repeat the steps of choosing a $\mathbf{x}$, train the network, and evaluate $f(\mathbf{x})$. That is, based on the result of the previous evaluation, selection and evaluation process are repeated within a given amount of time budget or until a predetermined objective value is obtained. However, manually carrying out this process is time- and effort-consuming, and doing it well requires in-depth comprehension of deep learning which may be more difficult than learning feature engineering techniques.

To overcome the above difficulty, practitioners often use simple grid or random search as the default automated HPO method [1]. As a more sophisticated solution, Bayesian Optimization (BO), which is a sequential model-based optimization method, can be used. BO has been successfully applied to many traditional machine learning and deep learning tasks. Spearmint [2] using Gaussian Process (GP), Sequential Model-Based Algorithm Configuration (SMAC) [3] using random forests (RF), and Tree-structured Parzen Estimator (TPE) [4] using non-parametric density estimation are widely used in practice.

Automated HPO algorithms, however, can become inefficient if DNN models with unpromising hyperparameter configurations are trained until the end of the pre-fixed number of training epochs without early termination. It would be better to stop training DNNs if we convincingly identify that the current hyperparameter configuration is not promising. In this way, the computation resource can be allocated to more promising settings. Therefore, accurate performance prediction of DNNs at early training stage can be considered to be essential for an efficient HPO. This issue is critical for DNNs because training cost of DNNs is usually much higher than that of traditional machine learning algorithms in terms of time and

978-1-5386-5457-6/18/$31.00 ©2018 IEEE

computational resource.

Lately, a few studies proposed learning curve prediction (also called performance curve prediction) models and showed that results are promising when the models are applied to HPO and Neural Architecture Search[1] (NAS) of DNNs. The models predict the final objective value at an early point before training ends. Learning curves can be modeled with the partial curve of the current evaluation alone or along with the curves of the previously completed ones. Hyperparameters of DNNs can be used as well because each hyperparameter can contribute to forming a distinct curve shape. Probabilistic [10] [11] and regression models [5] of learning curve prediction were proposed. Automated early stopping rules were provided in [12] as well. However, their analysis of learning curves is rather limited, and the effectiveness of learning curve prediction needs to be further investigated.

In this paper, we pre-evaluate 20,000 DNN hyperparameter configurations for each of MNIST and CIFAR-10 datasets and analyze the learning curves to gain insights for designing learning curve prediction models (Section III). We propose two rules that early-terminate the training of DNNs and investigate the performance trade-offs (Section IV). We discuss the practical difficulties of using learning curve prediction in Section V.

II. Related Work

Domhan et al. proposed a probabilistic learning curve prediction model to terminate low-performance configurations early [10]. By speeding up DNN HPO using the prediction model for early-termination, they identified hyperparameter configurations that outperform human-tuned configurations. More recently, Baker et al. proposed regression models to predict the performance of DNNs at an early stage of training [5]. They used three types of input features for the regression model: time-series validation accuracies including their first and second-order differences, architecture parameters, and other hyperparameters. As a Bayesian approach, Klein et al. used Bayesian neural networks for learning curve prediction [11].

In the studies above, public benchmark datasets such as MNIST, CIFAR-10/100, and Penn Tree Bank (PTB) were used. Hyperparameter ranges were determined mainly by reflecting the way practitioners manually tune hyperparameters. For example, the number of units in a layer is typically set to a power of 2. Categorical hyperparameters such as activation functions and regularizer types are rarely used. In such previous studies, the use of knowledge on each dataset and DNN architecture could be an important part of obtaining good learning curve prediction performance. Therefore, more experiments, where such knowledge is seldom available, are required to confirm if learning curve prediction can be used in practice, especially when a new type of dataset and task needs to be addressed.

[1]There is no formal definition of NAS (called meta-modeling in [5]). NAS is used to find new architectures so more focuses on architecture parameters such as the number of layers and skip connections that are known to be difficult to be modeled by BO methods. Model-free approaches such as reinforcement learning [6] [7] [8] and genetic algorithm [9] are often used for NAS.

TABLE I: List of hyperparameters and their range settings for MNIST and CIFAR-10.

Hyperparameter	Type	Range setting	
		MNIST	***CIFAR-10***
Number of filters in C1	integer	1∼350	8∼32
Number of filters in C2	integer	1∼350	32∼64
Number of filters in C3	integer	-	64∼128
Number of filters in C4	integer	-	64∼128
Number of units in Fc	integer	1∼1024	10∼1000
Convolution filter size	integer	2∼10	2∼3
Pooling size in C1	integer	2∼3	2*
Pooling size in C2	integer	2∼3	2*
Learning rate (log scale)	float	$10^{-4} \sim 10^{-0.5}$	$10^{-4} \sim 10^{-0.5}$
L2 weight	float	0.0∼1.0	0.0∼1.0
Dropout rate	float	0.0∼0.9	0.5*
Activation function	cat.[a]	ReLU*	ReLU, ELU[b], tanh
Regularizer	cat.[a]	Dropout*	None, Dropout, BN[c]

[a]categorical, [b]Exponential Linear Units, [c]Batch Normalization
*fixed, not a hyperparameter

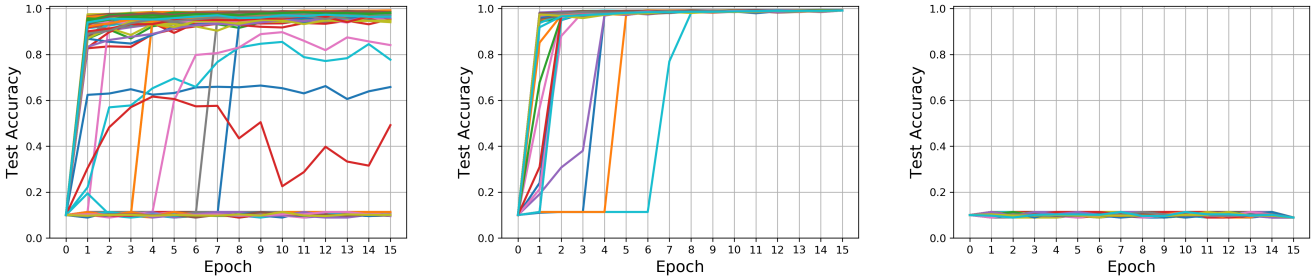
Early termination has already been adopted in real-world services. For instance, Google Vizier provides two automated early termination rules [12]. Its Performance Curve Stopping Rule uses Bayesian nonparametric regression, and Median Stopping Rule compares the performance of the current trial with the median performance of all completed trials. The Median Stopping Rule is model-free, so can be robust to the diverse shapes of learning curves. Our learning curve prediction is made with simple heuristic rules like this rule.

III. Learning Curves

A. Pre-evaluation of DNNs

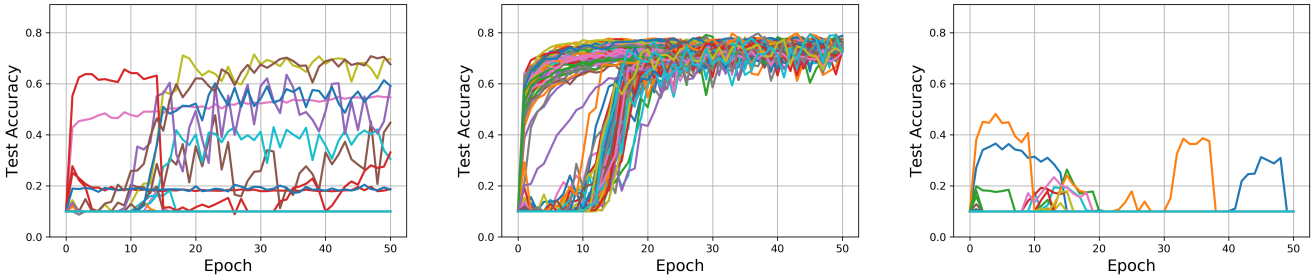
Typically, it has been assumed that common patterns exist for learning curves and that learning curves of unseen hyperparameter configurations can be modeled using a small number of sample curves. Therefore, the more learning curves we investigate, the more chances we have to observe if there are indeed common characteristics of learning curves. Recent studies [5] [11], however, evaluated only at most a few thousand configurations because DNN training is expensive. In this work, we pre-evaluate 20,000 hyperparameter settings. The number is more significant than what was investigated in previous studies.

Besides the purpose of investigating learning curves, there is another reason why pre-evaluating more configurations can be beneficial. By doing this, multiple HPO experiments can be performed without repeating DNN training for a given hyperparameter configuration. This is achieved by building a look-up table based on the 20,000 pre-evaluated configurations. The training and validation performances for a given number of epochs are read from the pre-built lookup table, and the required time for DNN training is also recorded when generating the pre-evaluation data and later read from the pre-built look-up table. In this way, HPO experiments can be repeated many times without running new DNN training, and only the optimization part of choosing the next configuration needs to be freshly calculated. Because the time required



(a) Random 100 (b) Top 100 (c) Bottom 100

Fig. 1: Random 100, the top 100, and the bottom 100 learning curves of MNIST



(a) Random 100 (b) Top 100 (c) Bottom 100

Fig. 2: Random 100, the top 100, and the bottom 100 learning curves of CIFAR-10

for optimization is typically much less than that of DNN training, the overall effect is a significantly reduced experiment time for evaluating HPO performance. In our configurations, the amount of time required for DNN training takes tens to hundreds of times more than that required for optimization for both of MNIST and CIFAR-10 depending on hyperparameter configurations. Therefore, the pre-evaluation strategy is useful to save time when performing extensive experiments with a variety of settings as we do in this work.

We generated hyperparameter settings using Sobol sequences that are low-discrepancy and quasi-random. Sobol sequences more evenly cover the high dimensional hyperparameter space than pseudorandom sequences, and it can be considered as a more representative set of the whole hyperparameter ranges. Total 20,000 configurations of nine-dimensional Sobol sequences were first generated and mapped to 20,000 configurations of nine hyperparameters. We then pre-evaluated the configurations, and validation accuracies and training times were recorded at each epoch to build a look-up table.

B. DNN Architectures and Hyperparameters

Two CNN architectures are used in this work. For MNIST, we use a small LeNet-5 style network [13] that consists of two convolutional layers with successive max-pooling layers, one fully-connected layer, and a softmax layer (C1-P1-C2-P2-Fc-Softmax)[2]. For CIFAR-10, we add two convolutional layers and one max-pooling layer on the network (C1-C2-P1-C3-P2-C4-P3-Fc-Softmax). Rectified Linear Unit (ReLU) and Adam are used as the activation function and the optimizer, respectively. Dropout and batch normalization are also included as regularizers, and they are used in the standard way.

Each CNN architecture has nine hyperparameters composed of architecture parameters, regularization related parameters, and learning rate as shown in Table I. Knowledge on the datasets was minimally used when determining the range of each hyperparameter. In other words, we did not apply only well-performing and nearby ranges of hyperparameters such that our study can have a more general interpretation. We instead chose wider ranges as much as possible because it makes more sense in practice when HPO is applied to a new dataset, and the high-performance range is unknown.

C. Analysis of Learning Curves

We analyze the pre-evaluated 20,000 learning curves to find patterns that can be used for designing learning curve prediction models. Sampled learning curves for random 100, top 100, and bottom 100 in terms of classification accuracy are shown in Fig. 1 and 2 for MNIST and CIFAR-10, respectively. One can easily observe that there is no strong and consistent pattern in random 100 plots, which is different from the results of the previous studies [5] [11]. In our study, the set and range of hyperparameters are different from those of previous studies where we have included many more combinations of hyperparameters. Therefore the random 100 plots indicate that it may be difficult to model learning curves according to simple assumptions based on a small number of sample curves.

[2]C: convolutional layer, P: max-pooling layer, Fc: fully-connected layer, Softmax: softmax layer, Numbers after layer names indicate layer indices.

The top 100 and the bottom 100 performing curves, however, show less diversity in learning curve patterns. Most of the top 100 curves in Fig. 1 (b) converge within one or two epochs, and their curve shapes are similar to each other even when some curves begin to rise relatively later after the third epoch. Two very distinct shapes are observed in Fig. 2 (b), and the two groups correspond to the use of different regularizers. Learning curves of DNNs without any regularizers and with dropout regularizer rise very rapidly once training starts, but curves of DNNs with batch normalization begin to climb after around tenth epoch. As for the bottom 100, it can be clearly seen that the curves are almost identical to one another. DNNs with these configurations do not seem to be trainable meaning that their performance is no better than a random guess that is around 0.1 for a ten-label classification task.

From the top 100 and the bottom 100 learning curves for the two datasets, it can be observed that most of the curves converge before reaching the half of the preset total training epoch. Therefore, it is reasonable to stop training DNNs if performance at a pre-determined early epoch is worse than the target performance. However, at the same time, a few DNNs with well-performing hyperparameter configurations can be undesirably stopped if training is terminated too early. This is because DNNs with a particular combination of hyperparameters is trained slowly during the early training phase, but eventually, they achieve competitive performance. We conjecture that this can happen more often when learning rate and regularization-related parameters are used as hyperparameters.

Based on the above observations, we can expect that early termination rules may save the time needed for HPO. However, performance improvement cannot be guaranteed when applying the rules, because it depends on the termination condition and time. We suggest two early termination rules and investigate their effectiveness in the following section.

IV. HPO using Early Termination Rules

A. Early Termination Rules

In this section, two early termination rules are proposed and analyzed. Learning curve prediction models can be implemented using a probabilistic or regression model using a small number of sample learning curves, but we instead develop heuristic rules based on our observations on learning curves of MNIST and CIFAR-10. This is because probabilistic and regression approaches can be convincing only when reasonably many evaluated curves are used to avoid overfitting and the cost for such evaluations before running HPO might not be acceptable.

As described in Section III, we recorded validation accuracy $y_{\mathbf{x}}^t$ with a hyperparameter configuration $\mathbf{x}$ at every epoch t while pre-evaluating DNNs, where $1 \leq t \leq T$, $t \in \mathbb{Z}^+$, and T is a preset total training epoch. We only use these validation accuracies for our rules other than DNN hyperparameters, and our rules stop training of a configuration at epoch τ if $y_{\mathbf{x}}^\tau$ meets termination criteria. Termination epoch τ and criteria are designed heuristically based on the example learning curves.

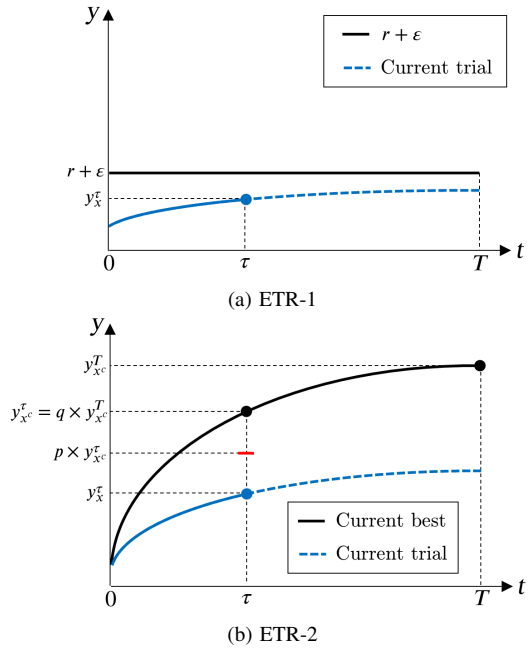
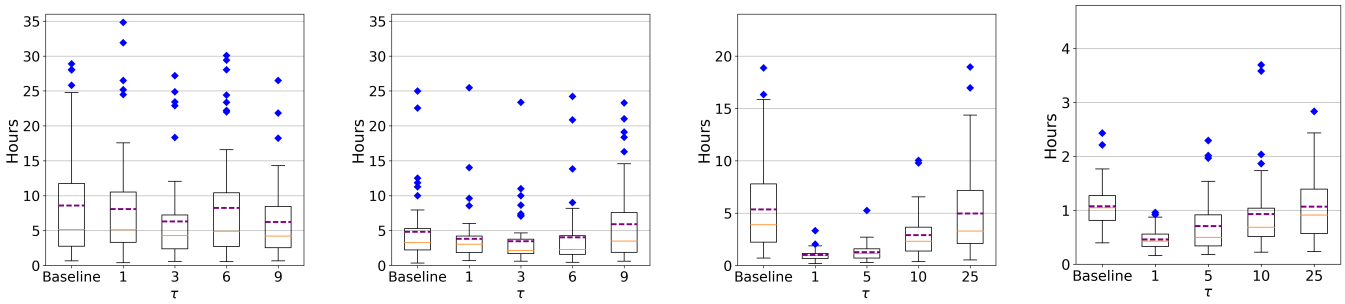


Fig. 3: Illustrations of ETR-1 and ETR-2. The blue learning curves meet early-termination criteria, so stop DNN training at τ.

Details on the termination criteria and how to select τ are elaborated below. Fig. 3 illustrates two early termination rules.

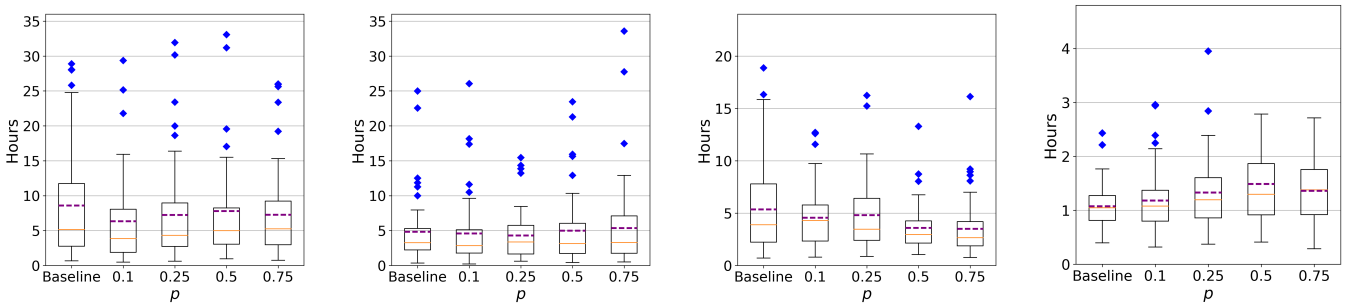
• **Early Termination Rule 1 (ETR-1)**. This rule terminates training at epoch τ if $y_{\mathbf{x}}^\tau < r + \epsilon$, where r is the performance of the random guess strategy, and ϵ is a small constant to reflect stochasticity of DNN training. The parameter r is set to 0.1 in our studies because our task has ten labels, and ϵ is fixed to 0.05 in this work. Our motivation for this rule is that DNNs with an inappropriate combination of hyperparameters are difficult to be trained. It can be observed that some DNN configurations are not trainable meaning that the performance at the final epoch $y_{\mathbf{x}}^T$ is no better than r as shown in Fig. 1 (c) and 2 (c).

• **Early Termination Rule 2 (ETR-2)**. This rule terminates training at epoch τ if $y_{\mathbf{x}}^\tau < p \times y_{\mathbf{x^c}}^\tau$, where $\mathbf{x^c}$ is the best hyperparameter configuration found so far during HPO, and τ is the maximum epoch satisfying $y_{\mathbf{x^c}}^\tau \leq q \times y_{\mathbf{x^c}}^T$. In other words, τ is set to the epoch where $(q \times 100)\%$ of the current best $y_{\mathbf{x^c}}^T$ is achieved, and therefore is updated whenever a new best hyperparameter configuration is observed. p is a tuning parameter to determine how much gap between $y_{\mathbf{x}}^\tau$ and $y_{\mathbf{x^c}}^\tau$ is allowed. This rule is motivated by the fact that BO sequentially and iteratively searches a better hyperparameter configuration than the current best one. It is, therefore, reasonable to terminate DNN training of a candidate hyperparameter configuration when the configuration does not seem to have a high probability of performing better than the current best found so far. Note that early termination epoch τ is a tuning parameter for ETR-1, but is adapted as the current best configuration changes for ETR-2. q is fixed to 0.75 in our experiments.



(a) MNIST (Top 0.25%) (b) MNIST (Top 0.5%) (c) CIFAR-10 (Top 0.05%) (d) CIFAR-10 (Top 0.5%)

Fig. 4: Distribution of time to achieve a goal accuracy using ETR-1. The percentiles of goal accuracies are in the parenthesis.



(a) MNIST (Top 0.25%) (b) MNIST (Top 0.5%) (c) CIFAR-10 (Top 0.05%) (d) CIFAR-10 (Top 0.5%)

Fig. 5: Distribution of time to achieve a goal accuracy using ETR-2. The percentiles of goal accuracies are in the parenthesis.

Note that our two rules are model-free, which means that they do not depend on any probabilistic or regression model. Furthermore, the information of all the completed trials and DNN hyperparameters are not used, making the method less sensitive to the diversity of learning curve characteristics.

B. HPO Algorithm and Acquisition Function

Our experiments were performed using SMAC based on RF [14] as the HPO algorithm. RF is an ensemble algorithm of multiple decision trees. RF regression for BO is performed as follows. Each tree is built using bootstrapped samples of a few evaluated hyperparameter configurations seen until the current iteration and predicts a single objective value which is accuracy performance of DNNs in our experiments. Mean and variance of the objective value of each candidate hyperparameter setting is then computed using the RF. The variance gives uncertainty, and the acquisition function uses the mean and variance to evaluate candidate configurations and to choose the next candidate to be assessed. We use SMAC implemented in *Spearmint* package [2], and 50 trees are used for RF regression.

We chose EI (Expected Improvement) as the acquisition function to maximize expected improvement over the current best performance. EI reflects the magnitude of improvement against the current best as well as the probability of improvement. We would refer readers to Snoek et al. [2] for rigorous and mathematical definitions of different acquisition functions.

C. Experimental Results

We performed 50 experiments using SMAC with EI as the HPO algorithm. For each dataset, experiments with two different accuracy goals were performed. Goal accuracies were set to 0.992 (top 0.25%) and 0.9915 (top 0.5%) for MNIST, and set to 0.7635 (top 0.05%) and 0.7229 (top 0.5%) for CIFAR-10. Then the HPO performance was evaluated by checking the time needed to find an acceptable solution. This experiment was repeated 50 times for each accuracy goal of each dataset.

Fig. 4 and 5 show distributions of time to achieve the accuracy goal. It can be seen that both ETR-1 and ETR-2 can improve or hurt performance for both of MNIST and CIFAR-10. We also observed that there is a trade-off between when to terminate (τ) and HPO performance in Fig. 4. In the case of $\tau = 9$ in Fig. 4 (b), early termination degrades performance considerably. Fig. 4 (c), however, shows a drastic performance improvement when $\tau = 1$. It can also be seen in Fig. 5 that a trade-off exists between how much gap is allowed to the current best (p) and the expected time to achieve a goal performance. Overall, trade-offs were easily found indicating that developing a general rule can be difficult.

We can observe that the performances of the two rules vary depending on the datasets and the target performance. For example, ETR-1 works better for CIFAR-10 than MNIST where the characteristics of learning curves are different and the training time of CIFAR-10 is longer than that of MNIST as shown in Fig. 4 (a) and (c). ETR-2 contributes to improving performance for CIFAR-10 when the goal accuracy is set to

the top 0.05 percentile but does not when the goal accuracy is set to the top 0.5 percentile as shown in Fig. 5 (c) and (d). This result may be due to the possibility that ETR-2 terminates some of the promising hyperparameter configurations undesirably.

V. Discussion and Conclusion

With our observations on learning curves and experimental results, we conclude that the HPO of DNNs using learning curve prediction is difficult for the following reasons.

First, it is difficult to build a learning curve prediction model that improves HPO performance in general. This is because the shapes of learning curves can drastically vary depending on the kinds and ranges of hyperparameters as shown in Fig. 1 and 2, and the HPO performance of DNNs changes as datasets and target accuracies vary as observed in Section IV. This statement matches the implication of the no free lunch theorem [15] that a specific learning curve prediction model can be designed to perform well on a particular task but not for all the tasks. Tuning a learning curve model only for a particular HPO task, however, might be meaningless in practice because HPO is most useful when applied to a completely new task.

Second, additional tuning parameters are commonly required for the learning curve prediction model itself, and HPO performance can be sensitive to tuning parameters. In the case of our rules, early termination criteria and time should be appropriately chosen to improve HPO performance as shown in Fig. 4 and 5. Selecting input predictors for learning curve prediction can be challenging as well.

Finally, learning curve prediction models may not be as useful as desired even when knowledge on the learning curves of a specific task is entirely used. This is related to how a BO algorithm behaves - typically it tries to choose a high-performance or high-potential setting instead of those subject to early termination. For instance, ETR-1 was designed based on the observation that many non-trainable configurations exist. However, the performance improvement for MNIST can be insignificant because SMAC with EI rarely chooses hyperparameter configurations that meet the termination criteria of ETR-1 (i.e., non-trainable).

To summarize, our observations and results highlight the practical difficulty of DNN learning curve prediction for improving HPO. To overcome these difficulties, one might need to obtain general knowledge on DNN configurations and their performance or to develop adaptive solutions that can slowly but confidently take advantage of the aspects that are very likely to be true for an HPO task under consideration. Designing an adaptive early termination rule having no tuning parameters of its own is a possible direction for future work.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2017R1E1A1A03070560) and supported by SK telecom Co., Ltd.

References

[1] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.

[2] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in neural information processing systems*, 2012, pp. 2951–2959.

[3] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *International Conference on Learning and Intelligent Optimization*. Springer, 2011, pp. 507–523.

[4] J. Bergstra, D. Yamins, and D. D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures." *ICML (1)*, vol. 28, pp. 115–123, 2013.

[5] B. Baker, O. Gupta, R. Raskar, and N. Naik, "Accelerating neural architecture search using performance prediction," 2018.

[6] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," *arXiv preprint arXiv:1611.02167*, 2016.

[7] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

[8] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," *arXiv preprint arXiv:1802.03268*, 2018.

[9] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, Q. Le, and A. Kurakin, "Large-scale evolution of image classifiers," *arXiv preprint arXiv:1703.01041*, 2017.

[10] T. Domhan, J. T. Springenberg, and F. Hutter, "Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves." in *IJCAI*, vol. 15, 2015, pp. 3460–8.

[11] A. Klein, S. Falkner, J. T. Springenberg, and F. Hutter, "Learning curve prediction with bayesian neural networks," 2016.

[12] D. Golovin, B. Solnik, S. Moitra, G. Kochanski, J. Karro, and D. Sculley, "Google vizier: A service for black-box optimization," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 1487–1495.

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

[14] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[15] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE transactions on evolutionary computation*, vol. 1, no. 1, pp. 67–82, 1997.