Accelerating Hyperparameter Optimization of Deep Neural Network via Progressive Multi-Fidelity Evaluation

Guanghui Zhu(✉) and Ruancheng Zhu

National Key Laboratory for Novel Software Technology,
Nanjing University, Nanjing 210023, China
{guanghui.zhu,zrc}@smail.nju.edu.cn

Abstract. Deep neural networks usually require careful tuning of hyperparameters to show their best performance. However, with the size of state-of-the-art neural networks growing larger, the evaluation cost of the traditional Bayesian optimization has become unacceptable in most cases. Moreover, most practical problems usually require good hyperparameter configurations within a limited time budget. To speed up the hyperparameter optimization, the successive halving technique is used to stop poorly-performed configurations as early as possible. In this paper, we propose a novel hyperparameter optimization method FastHO, which combines the progressive multi-fidelity technique with successive halving under a multi-armed bandit framework. Furthermore, we employ Bayesian optimization to guide the selection of initial configurations and an efficient data subsampling based method to warm start the surrogate model of Bayesian optimization. Extensive empirical evaluation on a broad range of neural networks and datasets shows that FastHO is not only effective to speed up hyperparameter optimization but also can achieve better anytime performance and final performance than the state-of-the-art hyperparameter optimization methods.

Keywords: Hyperparameter optimization · Deep neural network · Multi-fidelity optimization

1 Introduction

In recent years, deep learning has achieved great success on a variety of machine learning problems such as computer vision and natural language processing. However, deep neural networks (DNNs) are with too many hyperparameters and the learning performance depends seriously on the careful tuning of them [8]. The correct setting of hyperparameters for DNNs often needs a tedious endeavor, and typically requires considerable expert knowledge and experience. As a result, both researchers and practitioners desire to set hyperparameters automatically without any human intervention.

Unlike traditional machine learning models, hyperparameter optimization for DNNs is more challenging. Since the architecture of DNNs is getting more

H. W. Lauw et al. (Eds.): PAKDD 2020, LNAI 12084, pp. 752–763, 2020.
https://doi.org/10.1007/978-3-030-47426-3_58

and more complex, training large DNNs is more computationally expensive. For example, state-of-the-art neural networks often require days or even weeks to train. Thus, the well-known Bayesian optimization [1,2,10,18,20] methods that view the performance as a black-box function suffer low computational efficiency due to the expensive evaluation of hyperparameters. On the other hand, most practical problems usually require a good hyperparameter configuration within a limited time budget. Besides the strong final performance given a larger budget, practical hyperparameter optimization methods should also achieve strong anytime performance in the case of a small budget.

To speed up the hyperparameter optimization of DNNs, Hyperband [15] uses the successive halving (SH) technique [11] to stop poorly-performed configurations as early as possible and dynamically allocate more resources (i.e., the number of iterations) to well-performed configurations. Another popular method is the multi-fidelity optimization [9,12,13,19]. Generally, the fidelity indicates the sampling ratio of the full dataset. Multi-fidelity optimization uses many cheap low-fidelity evaluations instead of expensive high-fidelity evaluations to extrapolate the performance of hyperparameter configurations on the full dataset.

In this paper, we propose a novel hyperparameter optimization method FastHO to further accelerate the hyperparameter optimization of DNNs, while achieving good anytime performance and final performance. FastHO combines the progressive multi-fidelity technique with successive halving under a multi-armed bandit framework. Each hyperparameter configuration is viewed as an arm. At first, we aggressively evaluate each arm with fewer resources (i.e., small iteration budget and low fidelity). The poorly-performed arms are discarded and more resources are dynamically allocated to the promising configurations. The process is repeated until the maximum iteration budget and the highest fidelity are reached. Furthermore, we employ Bayesian optimization to guide the selection of initial configurations. Additionally, an efficient warmup method based on data subsampling is proposed to initialize the surrogate model of Bayesian optimization. Extensive empirical evaluation on different neural networks and datasets shows that FastHO outperforms the existing hyperparameter optimization methods. FastHO is not only effective to speed up hyperparameter optimization but also can achieve robust and better final performance.

2 Related Work

Given a machine learning algorithm A having hyperparameters $\lambda_1; \ldots; \lambda_n$ with respective domains $\delta_1; \ldots; \delta_n$, we define its hyperparameter space as $\delta = \delta_1 \times \ldots \times \delta_n$. For each hyperparameter setting λ, we use A_λ to denote the learning algorithm A using this configuration. We further use $l(\lambda) = L(A_\lambda; D_{train}; D_{valid})$ to denote the validation loss (e.g., error rate) that A_λ achieves on data D_{valid} when trained on D_{train}. The hyperparameter optimization problem is then to find λ minimizing $l(\lambda)$.

Bayesian optimization (BO) is the most popular hyperparameter optimization method [2,18,20]. BO models the conditional probability $p(y|\lambda)$ of a configuration's performance on an evaluation metric y, given a set of hyperparameters λ.

The commonly-used probabilistic model in BO is Gaussian process (GP), but GP does not typically scale well to high dimensions and exhibits cubic complexity in the number of data points. Another model-based Bayesian optimization method is SMAC [10], which uses random forest as the surrogate model. SMAC can perform well in high-dimensional categorical spaces. TPE [1] is a non-standard Bayesian optimization algorithm based on tree-structured Parzen density estimators. Due to the nature of kernel density estimators, TPE easily supports mixed continuous and discrete spaces. The above three BO methods are well-established and successful, but they are inefficient for the hyperparameter optimization of DNNs due to the huge evaluation cost.

Unlike the model-based Bayesian optimization, the bandit-based strategy Hyperband [15] formulates hyperparameter optimization as a pure-exploration problem by addressing how to allocate resources among randomly-sampled hyperparameter configurations. Besides, it uses successive halving to early stop the poorly-performed configurations. Compared to Bayesian optimization, Hyperband shows strong anytime performance, but it may lead to poor final performance because the initial hyperparameter configurations are selected randomly. BOHB [4] takes advantage of both Bayesian optimization and Hyperband and thus achieves the state-of-the-art anytime performance and final performance.

Another efficient method for the tuning of hyperparameters is multi-fidelity optimization, which uses cheap approximations to the function of interest to speed up the overall optimization process. Generally, the fidelity can be represented by the sampling ratio of the full dataset. Multi-fidelity Bayesian optimization that ranges from a finite number of approximations to continuous approximations has been well studied [12,13,19]. Furthermore, a general multi-fidelity framework based on the black-box optimization methods was proposed [9].

However, due to the huge training cost of DNNs, the existing hyperparameter optimization methods are inefficient and time-consuming for DNNs. Even for the state-of-the-art method BOHB, it still requires 33 GPU days for optimizing the hyperparameters of a medium-sized residual network [4]. In this paper, we combine the successive halving technique with multi-fidelity optimization to accelerate the hyperparameter optimization of DNNs.

3 Method

In this section, we propose a novel early-stopping mechanism to speed up the hyperparameter optimization of DNNs by taking the number of iterations and multi-fidelity into account at the same time. We first analyze the low-fidelity evaluation bias of DNNs, which motivates to progressively increase the fidelity. Then, we introduce the IF-SH (Iteration-and-Fidelity Based Successive Halving) method in detail. Moreover, we propose an efficient warmup technique for Bayesian optimization to further improve the performance, especially the anytime performance of hyperparameter optimization.

3.1 Low-Fidelity Evaluation Bias

Multi-fidelity optimization uses many cheap low-fidelity approximations instead of the expensive high-fidelity evaluations to speed up the overall optimization process. The lower the fidelity is, the cheaper the evaluation will be. However, it is intuitive that the evaluation on a part of the dataset is badly biased because it provides less accurate information about the target function. The experimental results of multi-fidelity optimization on LightGBM [14] show that the hyperparameter configurations chosen by low-fidelity evaluations usually perform poorly on the test dataset. We have tried to apply BOHB to find the best hyperparameter configuration of a convolutional neural network LeNet (with two convolutional layers, a full-connection layer, and a softmax layer) on the MNIST and CIFAR-10 datasets. We evaluated the hyperparameter configurations chosen on the data subset (i.e., 10%) and the whole dataset, and then compared the test error rate. The results are shown in Table 1.

It turns out as expected that the configuration chosen by high-fidelity evaluations is superior to those selected by low-fidelity evaluations. Additionally, we note that the main difference between the hyperparameter configurations chosen by the high-fidelity and low-fidelity evaluations is the regularization hyperparameters such as weight decay and dropout rate. It makes sense because the neural networks trained on the data subset usually require more regularization to deal with overfitting.

Table 1. Test error rate (%) of LeNet on MNIST and CIFAR-10, using hyperparameter configurations chosen by BOHB with different fidelity evaluations. CIFAR-10+ means CIFAR-10 with standard data augmentation. Results are the average over 5 runs.

Data	MNIST	CIFAR-10	CIFAR-10+
The whole dataset	0.6 ± 0.05	20.68 ± 0.68	16.32 ± 0.54
10% data subset	0.76 ± 0.13	23.81 ± 1.17	17.94 ± 0.78

As shown in Table 1, the evaluation performance on the data subset is biased in different cases. Therefore, it is necessary to develop a new method to balance the low-fidelity and high-fidelity evaluations. To address this issue, we propose a progressive multi-fidelity evaluation technique. We further combine this technique with the existing successive halving optimization. The configurations are first evaluated with a small number of epochs and low fidelity. After filtering the poorly-performed configurations as early as possible, we dynamically increase the number of epochs and fidelity simultaneously for the remaining configurations. The process is repeated until the maximum number of epochs and the maximum fidelity (i.e., the full dataset) are used. We call this procedure IF-SH (Iteration-and-Fidelity Based Successive Halving).

3.2 Progressive Multi-Fidelity Evaluation

Multi-armed bandit based methods such as Hyperband and BOHB view each hyperparameter configuration as an arm and dynamically allocate resources to different arms. The existing successive halving method can ensure that the poorly-performed configurations are discarded as early as possible and those promising hyperparameter configurations will get more resources overtime automatically. For the hyperparameter optimization of DNNs, the resource means the number of iterations (i.e., epochs). Since using evaluation on fewer iterations as criteria in HyperBand is feasible, it is worth a try that we use evaluation on both fewer iterations and lower fidelity to judge a hyperparameter configuration, which will remarkably reduce the overall time cost.

Thus, we propose a progressive multi-fidelity evaluation method IF-SH and dynamically allocate multiple resources including iteration budget and fidelity. Specifically, IF-SH usually begins with a small iteration budget on data subsets instead of the whole dataset. Then, IF-SH ranks the configurations by the validation performance and select the top η^{-1} to continue running with an iteration budget η times larger and a fidelity θ times larger. This process is repeated until it runs with the largest iteration budget on the whole dataset.

Another problem of the multi-armed bandit based hyperparameter optimization is how to set the number of initial configurations n. Similar to Hyperband, we consider several possible values of n to balance exploration and exploitation. Associated with each value of n is a minimum iteration budget b_{min} that is allocated to all configurations. A larger value of n corresponds to a smaller b_{min} and hence means more aggressive early stopping.

Algorithm 1. Iteration-and-Fidelity Based Successive Halving

Input: Iteration budget $b_{\min}$ and $b_{\max}, \eta, \theta$

$s_{\max} = \log_{\eta} \frac{b_{\max}}{b_{\min}}$
for s in $\{s_{\max}; s_{\max} - 1; \ldots; 0\}$ **do**
  $n = \frac{s_{\max}+1}{s+1} * \eta^s$
  $T =$ get hyperparameter configurations(n) using Bayesian optimization
  $b_{\min} = b_{\max} * \eta^{-s}$
  $f_{\min} = \theta^{-s}$
  //begin the SH inner loop
  for i in $\{0; \ldots; s\}$ **do**
    $n_i = n * \eta^{-i}$
    $b_i = b_{\min} * \eta^i$
    $f_i = f_{\min} * \theta^i$
    $D_{\text{sub}} =$ sample f_i data from the training dataset D_{train}
    $L = \{$ run on D_{sub} then return validation loss(t, b_i): t in $T\}$
    $T = top_k(T, L, n_i/\eta)$
  end for
end for
return *Configuration with the smallest intermediate loss seen so far*

Algorithm 1 shows the process of IF-SH, which requires the following inputs: (1) $[b_{min}, b_{max}]$ that determines the iteration budget space (2) η, an input that controls the proportion of configurations discarded and the number of iterations in each round of SH. Also, η determines the minimum number of iterations of the next round (3) θ, an input that controls the size of fidelity in each round of SH. θ also determines the minimum fidelity of the next round.

This algorithm balances between very aggressive evaluations with many configurations on the minimum resource, and very conservative runs that are directly evaluated on the maximum resource. Table 2 displays the resources allocated within each round of SH in IF-SH. The size of data subsampling is controlled by θ. In practice, the difference caused by various θ settings is not so notable. We will discuss it in Sect. 4.1.

Table 2. The values of n_i, b_i and f_i in IF-SH corresponding to various values of s, when $b_{\min} = 1, b_{\max} = 27, \eta = 3, \theta = 3$

i	$s=3$			$s=2$			$s=1$			$s=0$		
	n_i	b_i	f_i	n_i	b_i	f_i	n_i	b_i	f_i	n_i	b_i	f_i
0	27	1	1/27	9	3	1/9	6	9	1/3	4	27	1
1	9	3	1/9	3	9	1/3	2	27	1			
2	3	9	1/3	1	27	1						
3	1	27	1									

3.3 Surrogate Model Warmup

Hyperparameter configurations within each round of SH is selected by Bayesian optimization (Line 4 in Algorithm 1). It is well known that all model-based optimization methods including Bayesian optimization need initial observations to build the surrogate model. The most commonly-used startup is to choose random hyperparameter configurations, which is not efficient and robust. When the randomly-sampled configurations perform poorly, the surrogate model will be slow to work, causing a negative influence on anytime performance. A lot of previous work [5,16,21] focuses on meta-learning to handle this issue, but they require historical data or pre-trained models.

Obviously, the data subset contains part information of the whole dataset. As discussed in Sect. 3.1, the difference between configurations chosen by the data subset and the whole dataset is not too remarkable. Thus, using the sampling data to warm start the surrogate model is a feasible way. The configurations chosen by low-fidelity evaluations probably exceed the randomly-sampled configurations, although their final performance may not be so satisfying. To improve the efficiency of the warmup phase, we just run one round of SH. We first sample data from the training dataset with the sampling percent r. Then, we run SH on the sampling data D_r and select the top-k configurations to warm up the

surrogate model. In Sect. 4.1, we will evaluate the effect of r, subsampling percent of the warmup phase. In fact, by selecting more promising hyperparameters rather than random selection, the warmup phase is helpful for improving the final performance of hyperparameter optimization.

4 Experiments

In this section, we evaluated the empirical performance of our proposed method FastHO on different neural networks including CNN, Fully-Connected neural network, ResNet18 [7], and ResNet with Shake-Shake [6] and Cutout [3] regularization. The datasets include MNIST, CIFAR-10, and CIFAR-100. We compared the anytime performance and final performance of FastHO with TPE [1], Hyperband [15], and BOHB [4]. BOHB is the state-of-the-art hyperparameter optimization method that combines TPE and Hyperband (HB). We set $\eta = 3$, $\theta = 3$, $r = 0.1$ as default and explore how to set suitable θ and r. If not stated otherwise, for all methods we report the average error rate on the test dataset.

4.1 Convolutional Neural Network

We first evaluated FastHO on a CNN with two convolutional layers, a full-connection layer, and a softmax output layer. We optimized the hyperparameters including learning rate, momentum, weight decay, dropout rate, batch size, the number of full-connection units, kernel size, and weight initialization mode. For this network, we set $b_{min} = 2$ and $b_{max} = 60$ for successive halving. The budget indicates the number of epochs. The IF-SH process contains 4 rounds of successive halving, resulting in 240 epochs in total.

CIFAR-10: The CIFAR-10 dataset contains 50000 training and 10000 test RGB images with 32×32 pixels. The standard data augmentation techniques (i.e., random crop and horizontal flip) are used. To perform hyperparameter optimization, we split off 10000 training images as a validation set. Figure 1 shows the average test error of FastHO with and without the warmup phase. We also compared FastHO with TPE, Hyperband, and BOHB. The total iteration budget is 16 resource units and each resource unit represents 240 epochs. As a result, the complete FastHO process includes 16 runs of IF-SH.

From Fig. 1, we can see that the traditional Bayesian optimization method TPE does not work well and has the worst anytime performance. HB improves the efficiency of hyperparameter optimization with successive halving and thus achieves better anytime performance than TPE. However, its convergence to the global optimal value is limited by its reliance on randomly-drawn configurations. Thus, the final performance of HB is not very strong. BOHB performs well with limited resources and at the same time can achieve better final performance.

In contrast, FastHO outperforms BOHB on anytime performance by combining the progressive multi-fidelity optimization with SH. Furthermore, the

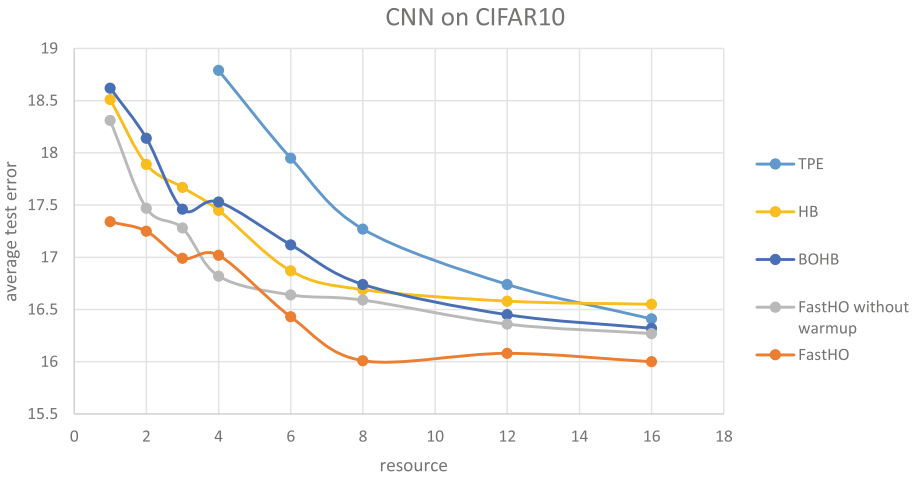

Fig. 1. Average test error of the best-observed configuration of CNN on CIFAR-10. One resource unit represents 240 epochs.

warmup technique can improve both anytime performance and final performance, while its time cost is negligible compared to the following hyperparameter optimization phase. More importantly, it can help FastHO to reach the best performance with much fewer resources. For instance, FastHO gets the best test error rate within 8 units of resources (8*240 epochs in total), about half of the resources consumed by other methods. Additionally, for the wall clock time, BOHB takes 31 h for hyperparameter optimization within 16 resource units. In contrast, FastHO takes only 19 h which is 63% faster than BOHB.

Evaluation of the θ and r Setting. We also evaluated two key parameters of FastHO: the proportion of θ that controls the size of fidelity in each round of successive halving, and the subsampling percent r in the warmup phase.

Intuitively, setting θ to a smaller value leads to a larger fidelity. If θ is set to 1, we will evaluate all configurations on the entire dataset. However, this disobeys our purpose to accelerate the evaluation procedure. In contrast, it should not be set to a very large value, because we aim to differentiate between configurations even when they are evaluated on the smallest data subset. Therefore, we set θ to be 2, 3 (default) and 4, and then compared their performance in Fig. 2(a).

As shown in Fig. 2(a), the difference caused by various θ settings is not so notable. The reason is that the three values are all appropriate ones that guarantee the discrimination of configurations on the smallest data subset. Besides, even if the evaluation bias exists on the low fidelity, the final evaluation is performed on the full dataset, which weakens this bias to some extent. Thus, FastHO is insensitive to the θ setting.

Next, we discuss the setting of r in the warmup phase. If r is set to a larger value, the data subset contains more information, leading to good hyperparameter configurations to warm start the Bayesian surrogate model. Nevertheless,

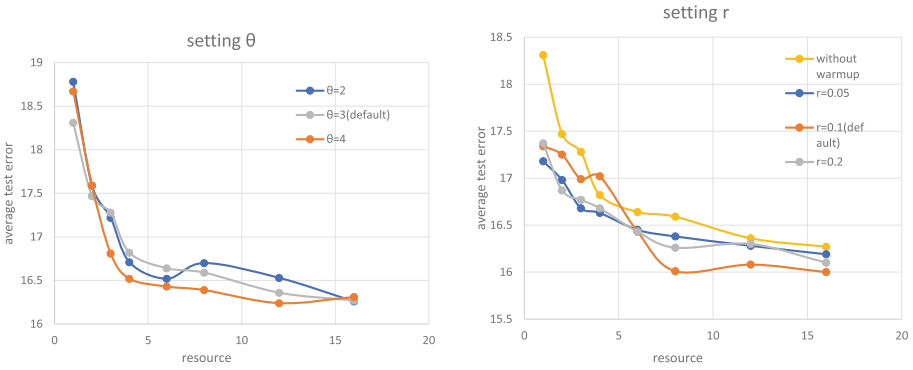

Fig. 2. Average test error of the best-observed configuration of CNN on CIFAR-10 with different θ and r settings. One resource unit represents 240 epochs.

the time cost of the warmup procedure will become larger. We chose 0.05, 0.1, and 0.2 for r, and then compared their performance in Fig. 2(b). Note that the warmup technique can improve the performance of FastHO no matter which value to choose. Meanwhile, none of these values is remarkably superior to other ones. The performance difference is acceptable due to the randomness in the data subsampling and hyperparameter optimization phases. Thus, we can infer that r makes little difference.

4.2 Fully-Connected Neural Network

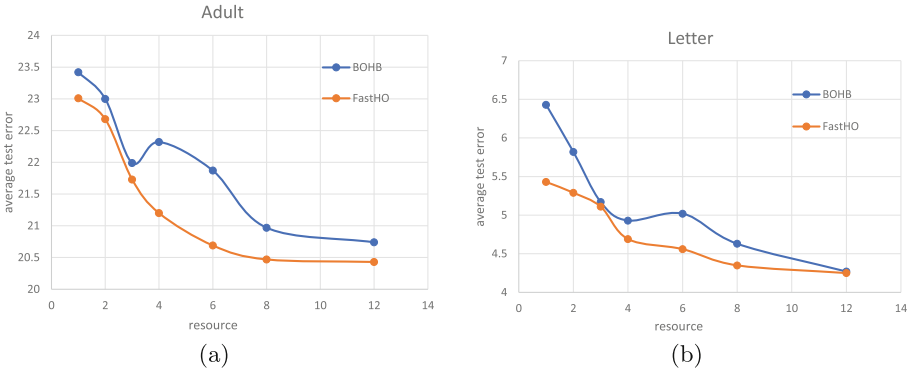

Fig. 3. Average test error of the best-observed configuration of FC network on Adult and Letter. One resource unit represents 90 epochs.

We optimized 6 hyperparameters that control the training procedure (learning rate, batch size, dropout rate, and weight decay). We also optimized the

architecture hyperparameters (number of layers, number of units per layer, weight initialization mode, and activation function) of a full-connected neural network. We selected two datasets: Adult and Letter, and set $b_{min} = 3, b_{max} = 30$. Since BOHB outperforms TPE and HB in most cases, we compared FastHO only with BOHB in the following experiments. In both Fig. 3(a) and Fig. 3(b), FastHO outperforms BOHB with not only the better anytime performance but also the better final performance. Moreover, the warmup technique is helpful for converging to optimum faster.

4.3 Large Convolutional Neural Network: ResNet

Next, we optimized the hyperparameters of large and widely-used neural networks including ResNet18 [7] and ResNet with Shake-Shake [6] and Cutout [3] regularization on the CIFAR-10 and CIFAR-100 datasets. We used standard data augmentation techniques (i.e., random crop and horizontal flip) and Nesterov momentum SGD optimizer with a cosine learning decay [17].

ResNet18: We tuned 4 hyperparameters including learning rate, momentum, weight decay, and batch size on the CIFAR-10 and CIFAR-100 datasets. CIFAR-100 shares similar input images to CIFAR-10. The only difference is that CIFAR-100 has 100 classes. We split the training dataset into 80% training data and 20% validation data. We set $b_{min} = 7$ and $b_{max} = 200$. As shown in Fig. 4, the performance improvement of FastHO is more significant, which indicates that FastHO is more effective for the hyperparameter optimization of larger neural networks. Moreover, FastHO is 67% faster than BOHB in terms of the total evaluation time cost.

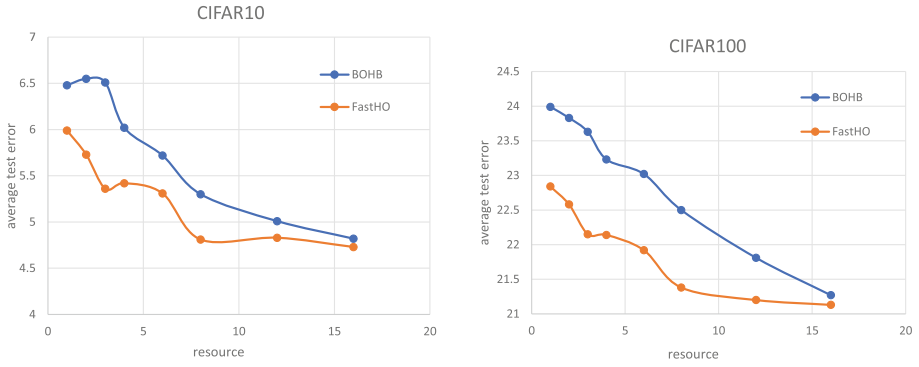

Fig. 4. Average test error of the best-observed configuration of ResNet18 on CIFAR-10 and CIFAR-100. One resource unit represents 800 epochs.

ResNet with Shake-Shake and Cutout Regularization: Next, we used the ResNet with Shake-Shake and Cutout regularization. For this network, we

set $b_{min} = 22$ and $b_{max} = 600$ and optimized learning rate, momentum, weight decay, and batch size. In this case, we just trained and evaluated the network with the best configuration after the complete hyperparameter optimization process (i.e., 16 runs of IF-SH). We ran the complete process 3 times and get a test error of $2.81\% \pm 0.07\%$, which is slightly larger than that reported in [4] ($2.78\% \pm 0.09\%$). However, regardless of the differences in training details or the search space setting, FastHO requires only 19 GPU days, while BOHB needs 33 GPU days. Moreover, as shown in Fig. 4, FastHO outperforms BOHB with much better anytime performance.

5 Conclusion and Future Work

In this paper, we presented a novel method to accelerate the hyperparameter optimization of DNNs by combining the progressive multi-fidelity technique with successive halving under a multi-armed bandit framework. Also, we proposed an efficient warmup method for the surrogate model of Bayesian optimization. Extensive empirical evaluation on a broad range of neural networks and datasets shows that FastHO is not only effective to speed up hyperparameter optimization but also can achieve better anytime performance and final performance than other state-of-the-art methods.

Future work includes taking feature subsampling into account to further accelerate hyperparameter optimization.

Acknowledgments. We thank Rong Gu, Chunfeng Yuan, and Yihua Huang for helpful advice. This work was supported by the National Natural Science Foundation of China (U1811461, 61702254), National Key R&D Program of China (2019YFC1711000), Jiangsu Province Science and Technology Program (BE2017155), National Natural Science Foundation of Jiangsu Province (BK20170651), and Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

1. Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: Proceedings of the 24th International Conference on Neural Information Processing Systems, pp. 2546–2554 (2011)
2. Bergstra, J., Yamins, D., Cox, D.D.: Making a science of model search: hyperparameter optimization in hundreds of dimensions for vision architectures. In: Proceedings of the 30th International Conference on Machine Learning, pp. 115–123 (2013)
3. Devries, T., Taylor, G.W.: Improved regularization of convolutional neural networks with cutout. CoRR abs/1708.04552 (2017)
4. Falkner, S., Klein, A., Hutter, F.: BOHB: robust and efficient hyperparameter optimization at scale. In: Proceedings of the 35th International Conference on Machine Learning, pp. 1436–1445 (2018)
5. Feurer, M., Springenberg, J.T., Hutter, F.: Initializing bayesian hyperparameter optimization via meta-learning. In: Proceedings of the 29th AAAI Conference on Artificial Intelligence, pp. 1128–1135 (2015)

6. Gastaldi, X.: Shake-shake regularization. CoRR abs/1705.07485 (2017)
7. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 630–645. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_38
8. Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., Meger, D.: Deep reinforcement learning that matters. In: Proceedings of the 32rd AAAI Conference on Artificial Intelligence, pp. 3207–3214 (2018)
9. Hu, Y., Yu, Y., Tu, W., Yang, Q., Chen, Y., Dai, W.: Multi-fidelity automatic hyper-parameter tuning via transfer series expansion. In: Proceedings of the 33rd AAAI Conference on Artificial Intelligence, pp. 3846–3853 (2019)
10. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: Coello, C.A.C. (ed.) LION 2011. LNCS, vol. 6683, pp. 507–523. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25566-3_40
11. Jamieson, K.G., Talwalkar, A.: Non-stochastic best arm identification and hyperparameter optimization. In: Proceedings of the 19th International Conference on Artificial Intelligence and Statistic, pp. 240–248 (2016)
12. Kandasamy, K., Dasarathy, G., Oliva, J.B., Schneider, J.G., Póczos, B.: Gaussian process bandit optimisation with multi-fidelity evaluations. In: Proceedings of the 30th International Conference on Neural Information Processing Systems, pp. 1000–1008 (2016)
13. Kandasamy, K., Dasarathy, G., Schneider, J.G., Póczos, B.: Multi-fidelity bayesian optimisation with continuous approximations. In: Proceedings of the 34th International Conference on Machine Learning, pp. 1799–1808 (2017)
14. Ke, G., et al.: LightGBM: a highly efficient gradient boosting decision tree. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 3149–3157 (2017)
15. Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., Talwalkar, A.: Hyperband: bandit-based configuration evaluation for hyperparameter optimization. J. Mach. Learn. Res. **18**(1), 6765–6816 (2017)
16. Lindauer, M., Hutter, F.: Warmstarting of model-based algorithm configuration. In: Proceedings of the 32rd AAAI Conference on Artificial Intelligence, pp. 1355–1362 (2018)
17. Loshchilov, I., Hutter, F.: SGDR: stochastic gradient descent with warm restarts. In: Proceedings of the 5th International Conference on Learning Representations (2017)
18. Mendoza, H., Klein, A., Feurer, M., Springenberg, J.T., Hutter, F.: Towards automatically-tuned neural networks. In: Proceedings of the Workshop on Automatic Machine Learning, pp. 58–65 (2016)
19. Sen, R., Kandasamy, K., Shakkottai, S.: Multi-fidelity black-box optimization with hierarchical partitions. In: Proceedings of the 35th International Conference on Machine Learning, pp. 4545–4554 (2018)
20. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. In: Proceedings of the 25th International Conference on Neural Information Processing Systems, vol. 2, pp. 2951–2959 (2012)
21. Springenberg, J.T., Klein, A., Falkner, S., Hutter, F.: Bayesian optimization with robust bayesian neural networks. In: Proceedings of the 30th International Conference on Neural Information Processing Systems, pp. 4141–4149 (2016)