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

***Guanghui Zhu*, Ruancheng Zhu***

*National Key Laboratory for Novel Software Technology*

*Nanjing University, China*

# Motivation

- **Deep neural networks (DNNs)**
  - ✓ Good Prediction Performance in many applications such as CV, NLP
  - ✗ Performance of DNNs depends on the hyperparameter configuration
  - ✗ Too many hyperparameters required to be carefully-tuned
  - ✗ Hyperparameter tuning of DNNs requires considerable expert knowledge and experience
- **Solution**
  - **Automatic hyperparameter optimization for DNNs without any human intervention**
  - However, unlike traditional ML models, hyperparameter optimization for DNNs is challenging

# Challenges

- Training large DNNs is computationally expensive
  - Days or even weeks to train
  - Powerful computational facilities (e.g. GPU)
- Well-known Bayesian optimization (BO) methods are inefficient
  - Require many evaluations to initialize the BO model
  - Each evaluation of hyperparameter configuration is very expensive
- Practical applications require good hyperparameter configurations with a limited time budget
  - Strong anytime performance in the case of a small budget

# Related Work

- **Hyperband**
  - ✓ successive halving (SH) technique to early stop poorly-performed configurations
  - ✓ dynamically allocate more iterations to well-performed configurations
  - × initial hyperparameter configurations are selected randomly, leading to poor final performance
- **BOHB**
  - ✓ combine Bayesian optimization with Hyperband
  - ✓ achieve the state-of-the-art anytime performance and final performance
  - × inefficient and time-consuming for DNNs (33 GPU days for optimizing hyperparameters of a medium-sized residual network)
- **Multi-fidelity optimization**
  - ✓ many cheap low-fidelity evaluations instead of expensive high-fidelity evaluations to infer the performance of hyperparameter configurations
  - ✓ the fidelity indicates the sampling ratio of the full dataset
  - × low-fidelity evaluation may exist bias

# Contributions

- **A novel hyperparameter optimization method *FastHO* to accelerate hyperparameter optimization of DNNs, while achieving good anytime performance and final performance**
  - Combines the progressive multi-fidelity technique with successive halving under a multi-armed bandit framework
  - 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
  - Employ Bayesian optimization to guide the selection of initial configurations
  - Initialize the surrogate model of Bayesian optimization using an efficient warmup method based on data sub-sampling
  - Extensive evaluation on different neural networks and datasets shows that FastHO outperforms the existing hyperparameter optimization methods

# Low-Fidelity Evaluation Bias

- The lower the fidelity is, the cheaper the evaluation will be. However, the evaluation on a part of the dataset may be badly biased because it provides less accurate information about the target function
- We have tried to apply BOHB to find the best hyperparameter configuration of a convolutional neural network LeNet on the MNIST and CIFAR-10 datasets
- We set the fidelity to be 0.1 (i.e., 10% of the full dataset)

# Low-Fidelity Evaluation Bias

**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 | $0.6 \pm 0.05$ | $20.68 \pm 0.68$ | $16.32 \pm 0.54$ |
| 10% data subset | $0.76 \pm 0.13$ | $23.81 \pm 1.17$ | $17.94 \pm 0.78$ |

- the configuration chosen by high-fidelity evaluations is superior to those selected by low-fidelity evaluations. The evaluation performance on the data subset is biased in different cases
- the main difference between the hyperparameter configurations is the regularization hyperparameters such as weight decay and dropout rate
- the neural networks trained on the data subset usually require more regularization to deal with overfitting

# Progressive Multi-Fidelity Evaluation

- To address this issue, we propose a progressive multi-fidelity evaluation technique. Moreover, we combine this technique with the existing successive halving optimization
- Specifically, 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)

# Progressive Multi-Fidelity Evaluation

- $[b_{min}, b_{max}]$ determines the iteration budget space
- IF-SH begins with a small iteration budget on data subsets instead of the whole dataset
- Then, it ranks the configurations by the validation performance and select the top $\frac{1}{\eta}$ to continue running with an iteration budget $\eta$ times larger and a fidelity $\theta$ times larger
- $\eta$ controls the proportion of configurations discarded and the number of iterations in each round of SH
- $\theta$ controls the size of fidelity in each round of SH

**Algorithm 1** Iteration-and-Fidelity Based Successive Halving

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

```
s_max = log_η (b_max / b_min)
for s in {s_max; s_max − 1; ... ; 0} do
    n = (s_max + 1)/(s + 1) ∗ η^s
    T = get hyperparameter configurations(n) using Bayesian optimization
    b_min = b_max ∗ η^(−s)
    f_min = θ^(−s)
    //begin the SH inner loop
    for i in {0; ... ; s} do
        n_i = n ∗ η^(−i)
        b_i = b_min ∗ η^i
        f_i = f_min ∗ θ^i
        D_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/η)
    end for
end for
return Configuration with the smallest intermediate loss seen so far
```

# Progressive Multi-Fidelity Evaluation

PAKDD2020 SINGAPORE

NJU-PASA Lab
Parallel Algorithm, System & Application for Big Data

- Another problem is how to set the number of initial configurations $n$ in each round of SH. We consider several possible values of $n$ to balance exploration and exploitation
- Associated with each value of $n$ is a minimum iteration budget. A larger value of $n$ corresponds to a smaller $b_{min}$, meaning more aggressive early stopping
- Table 2 displays the resources allocated within each round of SH in IF-SH. IF-SH 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.** 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 | | | | | | | | | |

# Surrogate Model Warmup

- Hyperparameter configurations within each round of SH is selected by Bayesian optimization, which needs initial observations to build the surrogate model
- When the randomly-sampled configurations perform poorly, the surrogate model will be slow to work, causing a negative influence on anytime performance
- To address the issue, we use the sampling data to warm start the surrogate model. Specifically, 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 top-$k$ configurations to warm up the surrogate model
- In fact, by selecting more promising hyperparameters rather than random selection, the warmup phase is helpful for improving the final performance of hyperparameter optimization

# Surrogate Model Warmup

- Hyperparameter configurations within each round of SH is selected by Bayesian optimization, which needs initial observations to build the surrogate model
- When the randomly-sampled configurations perform poorly, the surrogate model will be slow to work, causing a negative influence on anytime performance
- To address the issue, we use the sampling data to warm start the surrogate model. Specifically, 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 top-$k$ configurations to warm up the surrogate model
- In fact, by selecting more promising hyperparameters rather than random selection, the warmup phase is helpful for improving the final performance of hyperparameter optimization

# Experiments

- We evaluated the empirical performance of our proposed method FastHO on different neural networks including CNN, Fully-Connected neural network, ResNet18
- The datasets include MNIST, CIFAR-10, and CIFAR-100
- We compared the anytime performance and final performance of FastHO with TPE, Hyperband, and BOHB
- $\eta = 3$, $\theta = 3$, $r = 0.1$ as default and explore how to set suitable $\theta$ and $r$
- If not stated otherwise, for all methods we report the average error rate on the test dataset

# 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 8 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 (60*4) epochs in total

# Convolutional Neural Network

- The traditional Bayesian optimization method TPE has the worst anytime performance
- Hyperband can achieve better anytime performance with SH. However, the final performance of HB is not very strong
- BOHB performs well with limited resources and can achieve better final performance
- FastHO outperforms BOHB on anytime performance by combining the progressive multi-fidelity optimization with SH. Moreover, the warmup technique can improve both anytime performance and final performance

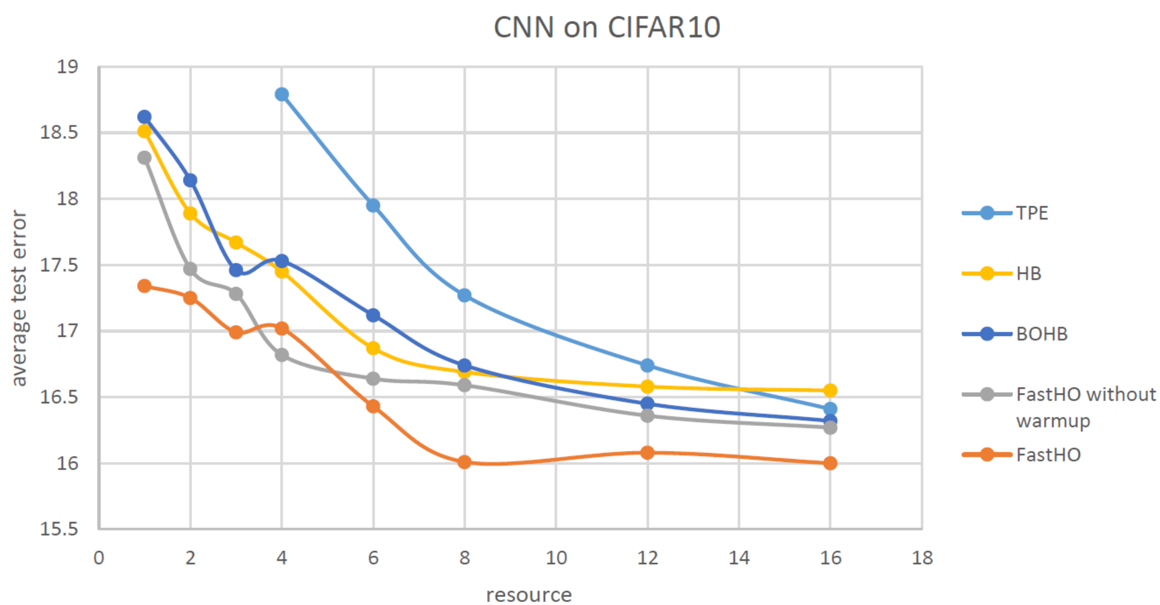


**Fig. 1.** average test error of the best-observed configuration of CNN on CIFAR-10. One resource unit represents 240 epochs.

# Convolutional Neural Network

- More importantly, FastHO can reach the best performance with much fewer resources (8 *240 epochs in total), about half of the resources consumed by other methods
- Additionally, for the wall clock time, BOHB takes 31 hours for hyperparameter optimization within 16 resource units. In contrast, FastHO takes only 19 hours, which is 63% faster than BOHB

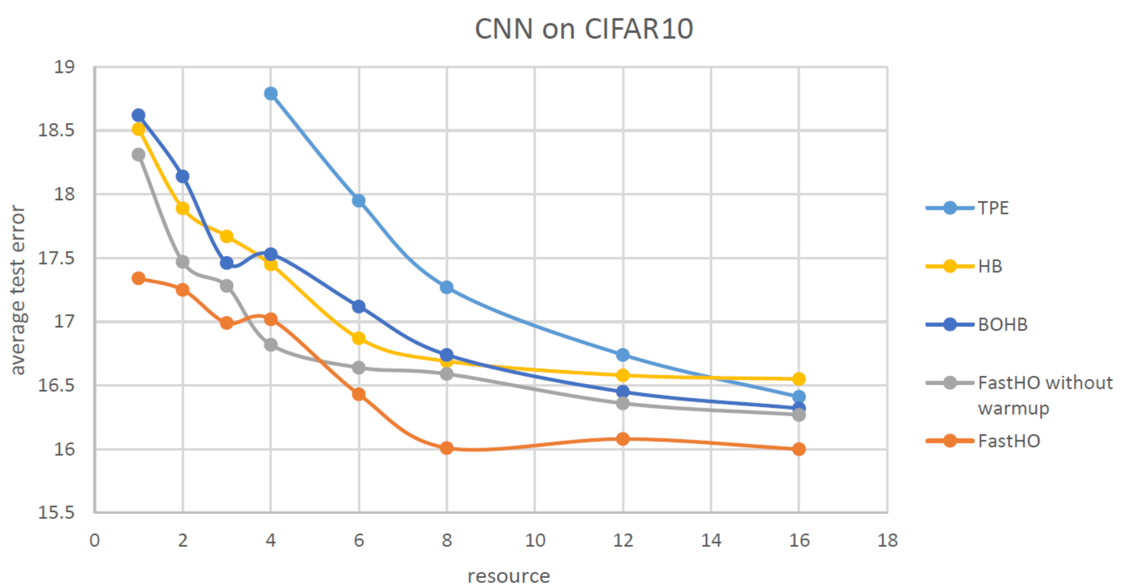


**Fig. 1.** average test error of the best-observed configuration of CNN on CIFAR-10. One resource unit represents 240 epochs.

# Evaluation of $\theta$ and $r$ setting

- We also evaluated two key parameters of FastHO: $\theta$ that controls the size of fidelity in SH, and the subsampling percent $r$ in the warmup phase
- The difference caused by various $\theta$ settings is not so notable and FastHO is insensitive to the $\theta$ setting
- The warmup technique can improve the performance no matter which value to choose. Meanwhile, none of these values is remarkably superior to other ones

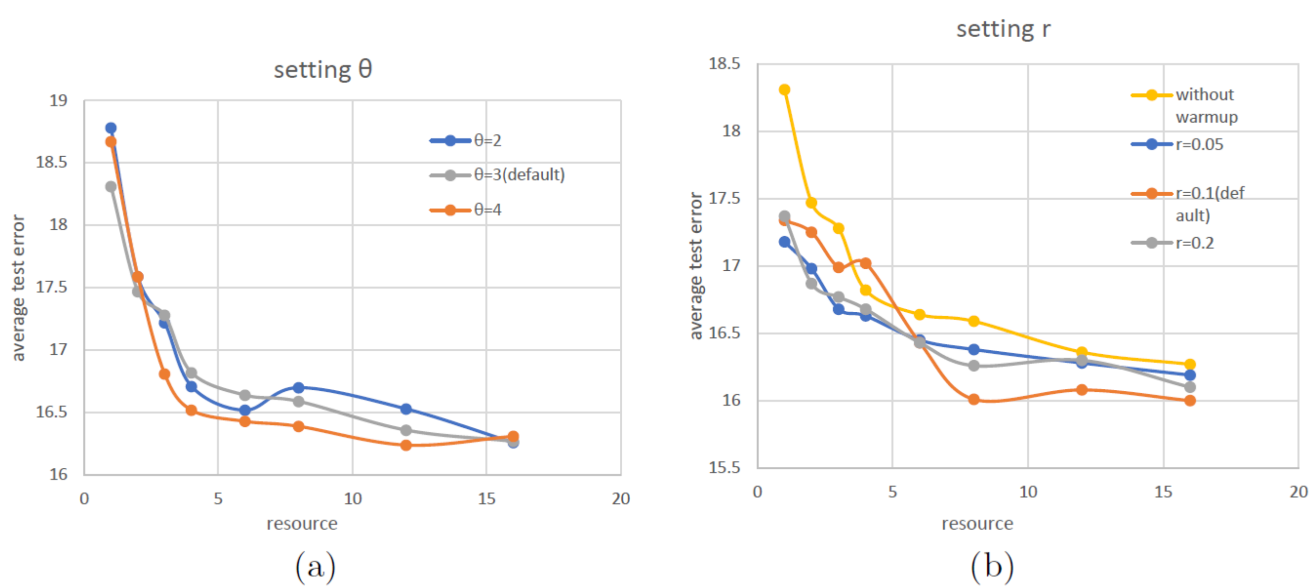


**Fig. 2.** average test error of the best-observed configuration of CNN on CIFAR-10 with different $\theta$ and $r$ settings. One resource unit represents 240 epochs.

# Fully-Connected Neural Network

- We optimized 6 hyperparameters that control the training procedure and 4 architecture hyperparameters of a fully-connected neural network
- We selected two datasets: Adult and Letter, and set $b_{min} = 3$, $b_{max} = 30$
- FastHO outperforms BOHB with not only the better anytime performance but also the better final performance

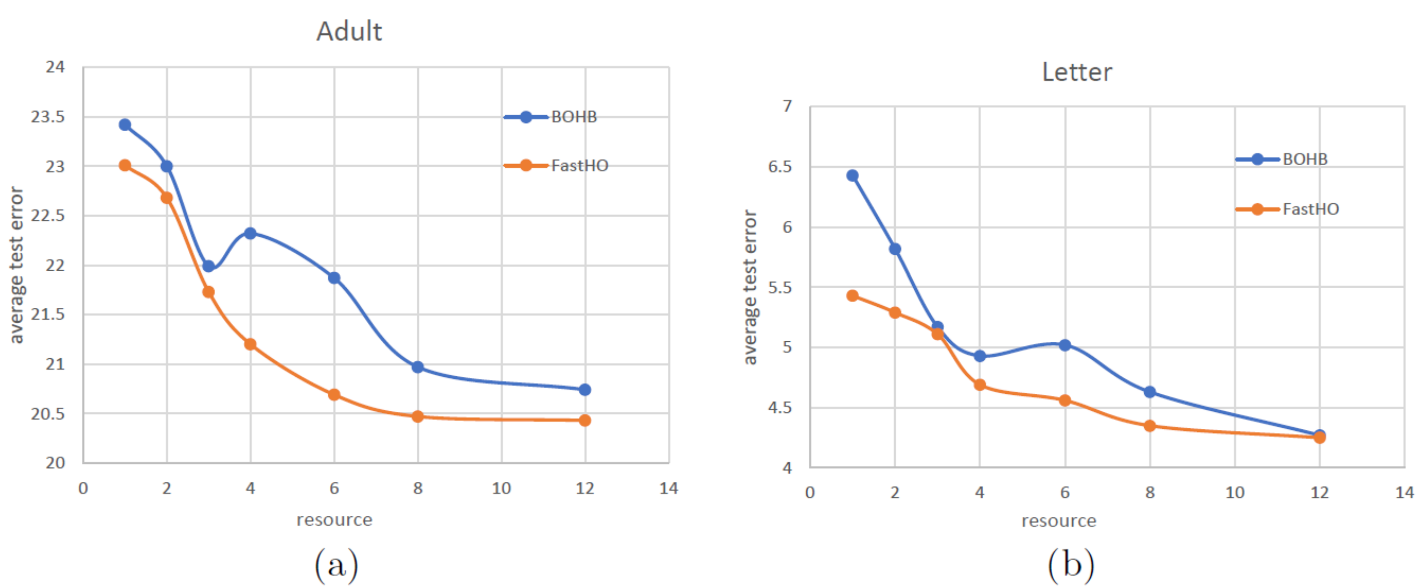


**Fig. 3.** average test error of the best-observed configuration of FC network on Adult and Letter. One resource unit represents 90 epochs.

# Large Convolutional Neural Network ResNet

- We tuned 4 hyperparameters including learning rate, momentum, weight decay, and batch size on the CIFAR-10 and CIFAR-100 datasets
- The performance improvement of FastHO is more significant, which indicates that FastHO is more effective for hyperparameter optimization of larger neural networks
- Moreover, FastHO is 67% faster than BOHB in terms of the total evaluation time cost

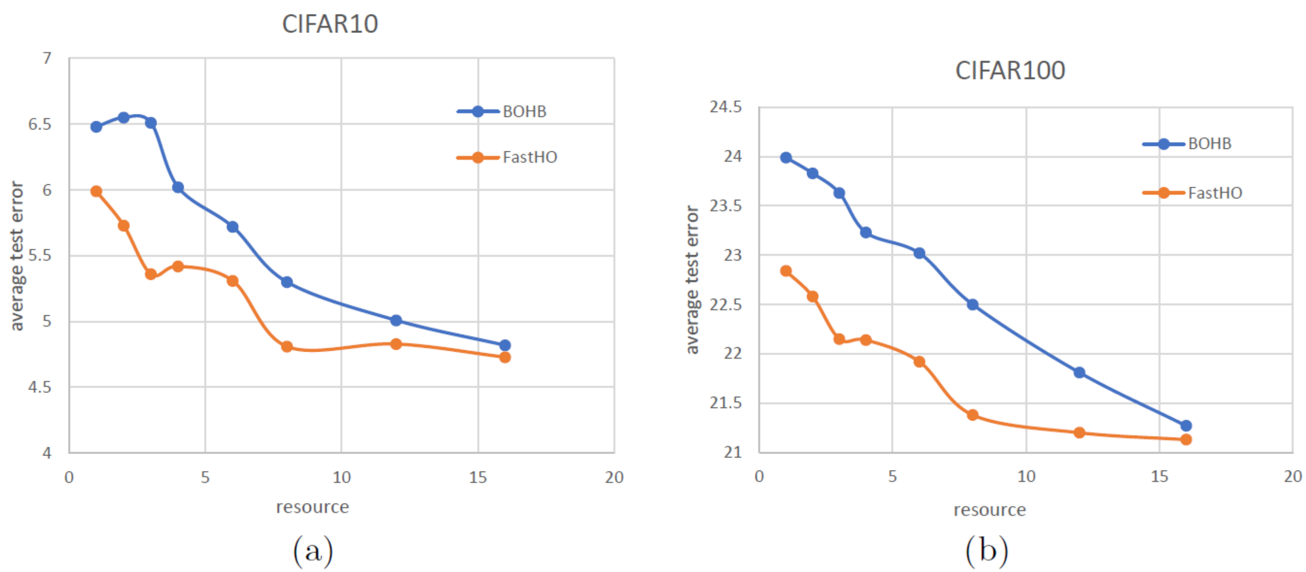


**Fig. 4.** average test error of the best-observed configuration of ResNet18 on CIFAR-10 and CIFAR-100. One resource unit represents 800 epochs.

# Conclusion and Future Work

PAKDD2020 SINGAPORE

NJU-PASA Lab

Parallel Algorithm, System & Application for Big Data

- 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
- We proposed an efficient warmup method for the surrogate model of Bayesian optimization
- Experimental results show 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 how to select suitable fidelities to avoid bias. We also plan to take feature subsampling into account to further accelerate hyperparameter optimization

# Thanks

Q&A