Improving Multi-fidelity Optimization with a Recurring Learning Rate for Hyperparameter Tuning

HyunJae Lee  Gihyeon Lee  Junhwan Kim  Sungjun Cho  Dohyun Kim  Donggeun Yoo
Lunit Inc.
{hjlee,gihyeon.lee,kimjh12,scho,донny8,dgyoo}@lunit.io

Abstract

Despite the evolution of Convolutional Neural Networks (CNNs), their performance is surprisingly dependent on the choice of hyperparameters. However, it remains challenging to efficiently explore large hyperparameter search space due to the long training times of modern CNNs. Multi-fidelity optimization enables the exploration of more hyperparameter configurations given budget by early termination of unpromising configurations. However, it often results in selecting a sub-optimal configuration as training with the high-performing configuration typically converges slowly in an early phase. In this paper, we propose Multi-fidelity Optimization with a Recurring Learning rate (MORL) which incorporates CNNs’ optimization process into multi-fidelity optimization. MORL alleviates the problem of slow-starter and achieves a more precise low-fidelity approximation. Our comprehensive experiments on general image classification, transfer learning, and semi-supervised learning demonstrate the effectiveness of MORL over other multi-fidelity optimization methods such as Successive Halving Algorithm (SHA) and Hyperband. Furthermore, it achieves significant performance improvements over hand-tuned hyperparameter configuration within a practical budget.

1. Introduction

Convolutional Neural Networks (CNNs) have recently achieved a huge success in a wide range of computer vision tasks [21, 25, 27, 29]. While the performance of CNNs is greatly affected by the choice of hyperparameters [7], the optimal combination of hyperparameters is hard to know a priori. Therefore, model developers often explore the hyperparameter configurations manually, which requires a huge labor cost yet it is sub-optimal with respect to performance [5]. Recent approaches in hyperparameter optimization (HPO) [4, 22] try to automate this painful tuning process by efficiently exploring multi-dimensional search space. In practice, it is known that the automatic tuning process has contributed significantly to improving the winning rate during the development of AlphaGo [6].

As modern CNNs get more sophisticated and complex, the search space of hyperparameters becomes larger and the training time becomes longer, which makes it difficult to explore diverse configurations. In order to speed up the HPO process and save computational expense, multi-fidelity optimization early stops unpromising configurations and adaptively allocates more resources to promising configurations. The early termination of configuration is determined by a low-fidelity approximation based on partial training results which is computationally much cheaper than full training.

Successive Halving Algorithm (SHA) [16] and Hyperband [22] are the two most popular multi-fidelity optimization methods. These methods explore orders-of-magnitude more configurations by early stopping low-performing configurations. However, we experimentally discovered that the high-performing configurations typically do not perform well in the early phase when training CNNs. Figure 1 illustrates the validation curves for the grid search of learning rate with ResNet-56 on CIFAR-100 dataset. It demonstrates that the best configuration does not outperform other configurations until the late phase of the training process. This slow-starting peculiarity of promising configurations limits the performance of multi-fidelity optimization when training CNNs due to the early termination of promising configurations.

A CNN is learned with a predefined learning rate schedule in general. From the beginning to the end of the learning rate schedule influences the final performance of the CNN. However, previous methods discard low-performing configurations at an early/mid-point of the learning rate schedule and consequently result in a sub-optimal configuration. Inspired by this problem, we propose Multi-fidelity Optimization with a Recurring Learning rate (MORL) which condenses the learning rate schedule to fit into every round of promotion. Even at an early phase of training, it enables precise evaluation of a configuration as the model comes from the entire schedule. To the best of our knowledge, this is the first work that integrates the learning rate schedule into multi-
fidelity optimization.

Our experiments on a wide range of general computer vision tasks including image classification [12], transfer learning [38] and semi-supervised learning [2] verify the efficacy of the proposed method. Throughout the experiments, MORL consistently outperforms other multi-fidelity optimization methods such as SHA and Hyperband. Furthermore, it outperforms highly hand-tuned configurations significantly within a reasonable budget. Also, we show that our method is orthogonal to the existing Bayesian optimization method and applicable to a variety of learning rate schedules. The experimental results demonstrate that MORL is a practical method available for a wide range of scenarios.

2. Related Work

Hyperparameter Optimization A CNN includes many hyperparameters and it is very crucial to find a good hyperparameter configuration to achieve successful performance [5]. While the search space for hyperparameter is often very high-dimensional and complex, it is hard to apply classical optimization methods that incorporate gradient descent, convexity or smoothness [9]. Therefore, model developers often go through a manual tuning process which requires a huge labor cost [5]. Hyperparameter optimization (HPO) aims to automate this painful process and search good hyperparameter configurations efficiently.

Bayesian optimization [4, 6] optimizes black-box function by adaptively suggesting hyperparameter configurations given observations. In order to estimate the target function, it first fits a probabilistic surrogate model with observations of input configuration and its corresponding performance. Then it selects the next configuration that maximizes acquisition function. For instance, expected improvement [17] attempts to trade off exploration against exploitation. By iteratively fitting a surrogate model and evaluating probable configuration, Bayesian optimization outperforms brute-force methods such as random search [4, 14]. However, Bayesian optimization inherently cannot be easily parallelized due to the nature of sequentially fitting a probabilistic model with previous observations [23]. Furthermore, it is known that Bayesian optimization does not adapt well to high-dimensional search spaces where it shows similar performance to random search [36].

More related to our method, multi-fidelity optimization methods enable to evaluate order-of-magnitude more configurations by exploiting cheaper proxy tasks, e.g. training model only for few iterations, using partial datasets or down-sized images [4, 22, 18]. As training a single configuration could take from days to weeks due to the increasing model complexity and dataset size [37], it becomes more crucial to speed up the HPO process by harnessing the power of multi-fidelity optimization. Successive Halving Algorithm (SHA) [16] uniformly allocates a small initial budget among randomly selected configurations and early stops the worst half configurations. It then doubles the budget and repeats the same process until it reaches maximum resource. In addition to its simplicity, SHA shows comparable performance to other state-of-the-art HPO methods such as Vizier [11], FABOLAS [18] and PBT [15] in a wide range of tasks [23]. Hyperband further extends SHA by automating the choice of initial budget via running different variants of SHA with respect to initial resource. BOHB [8] combines Hyperband with Bayesian optimization in order to benefit from both adaptive resource allocation and configuration sampling. Our approach enables more precise early stage ranking among configurations by incorporating the learning rate schedule into multi-fidelity optimization.

Recurring learning rate schedule Learning rate is one of the most important factors when tuning the performance of CNNs [32]. A typical learning rate schedule used by a wide range of modern CNNs is a step learning rate schedule where an initial learning rate is decayed by an adequate factor at given milestone epochs [20, 12, 31, 13, 34]. In an attempt to eliminate the need for tuning the initial learning rate and its schedule, Cyclical Learning Rates (CLR) [32] monotonically increases then decreases the learning rate within reasonable bounds and repeats this process cyclically. Stochastic gradient descent with warm restart (SGDR) [26] periodically warms restarts SGD process where the learning rate is reinitialized in each cycle and scheduled to decrease following cosine annealing schedule. SGDR accelerates the training process of CNN and it achieved state-of-the-art performance by an ensemble of models obtained at the end of each cycle. Our method differs from these methods in that...
we aim to improve multi-fidelity optimization by leveraging the learning rate schedule.

3. Multi-fidelity Optimization with Recurring Learning Rate

In this section, we introduce Multi-fidelity Optimization with a Recurring Learning rate (MORL) algorithm which extends the Successive Halving Algorithm (SHA) to improve low fidelity approximations by taking the optimization process of CNNs into consideration. We first provide a brief introduction to SHA, motivate the need for a recurring learning rate then introduce MORL algorithm in detail.

3.1. Successive Halving Algorithm

Given a reduction factor \( \eta \) and maximum resource \( r \), SHA trains the network with an initial resource allocated to each configuration, evaluates the performance of all configurations, and promotes top \( 1/\eta \) configurations to the next round. It then increases the resource allocation for each configuration by a factor of \( \eta \) and repeats the process until the resource allocation per each configuration reaches \( r \). By allocating relatively small initial resources and early stopping, SHA enables the evaluation of order-of-magnitude more configurations given a fixed budget. However, when low-fidelity approximation does not reflect the final performance, SHA ends up terminating configuration which will have high performance at the end.

Figure 1 depicts the validation curves of grid search on learning rate with stochastic gradient descent (SGD) and step learning rate schedule which is typically employed when training modern CNNs [20, 12, 31, 13, 34]. The top-performing configuration rather shows inferior performance in the early phase but begins to exhibit high performance in the late phase. In this scenario, utilizing SHA results in the early termination of the top-performing configuration. While assigning a large initial budget could mitigate this problem, it leads to evaluating only a small number of configurations which limits the benefit of utilizing multi-fidelity optimization.

3.2. Effect of Recurring Learning Rate

SGD updates the parameter of the network as \( \theta = \theta - \eta g \) given a learning rate \( \eta \), model parameter \( \theta \) and gradient \( g \). Since convergence towards the final performance generally occurs when the magnitude of the parameter update is small [39], we assumed that it would be more reasonable to evaluate low-fidelity approximation when the learning rate is small. As a typical learning rate schedule starts with a relatively high learning rate and ends with a relatively low learning rate, we condense the original learning rate schedule to fit each round of promotion so that each round ends with a small learning rate. We refer to this scheme as a recurring learning rate where the learning rate schedule is condensed and restarted every round of promotion. Figure 2 follows the same procedure as Figure 1 using a recurring learning rate every 30 epochs with a cosine annealing schedule, and reports the rank of the best configuration at the end of each cycle. When compared with Figure 1, the slow-starting property of the best configuration is much alleviated while the final performance remains. The top-performing configuration begins to outperform other configurations at the late phase of each cycle where it has a relatively small learning rate which confirms our assumption.

3.3. MORL Algorithm

Among various multi-fidelity optimization methods, we opt for SHA due to its simplicity and theoretically principled justification. While maintaining the essence of SHA, MORL enhances the ability to early differentiate promising configurations with a recurring learning rate. The overall procedure of MORL is summarized in Algorithm 1. MORL first suggests a set of hyperparameter configuration \( H \) with suggest_hyperparameters subroutine given a number of configurations \( n \), then it initializes the network with create_network subroutine for each hyperparameter \( h \). There exists a total of \( \lfloor \log_\eta (r) \rfloor - s_{\min} + 1 \) rounds of promotion where minimum resource of \( \eta^{s_{\min}} \) is allocated to each configuration on the first round and increases by a factor of \( \eta \) every round until it reaches maximum resource \( r \).

Each get_start_epoch and get_end_epoch subroutine computes the beginning and last epoch given exponent \( s \) in each round, respectively. The start epoch \( c_{\text{start}} \) is set to \( \eta^{s-1} + 1 \), except for the first round where it is set to 1. On the other hand, the end epoch \( c_{\text{end}} \) is set to \( \eta^s \) except for the last round which is set to \( r \). In each round, every hyperparameter \( h \in H \) is evaluated after training with allo-
While hyperparameter tuning is a critical aspect of machine learning, it is often challenging to determine the optimal values for various parameters. One approach to address this issue is through multi-objective reinforcement learning (MORL), which allows for the optimization of multiple objectives simultaneously. In this section, we will discuss the effectiveness of MORL in hyperparameter optimization.

### 3.1. MORL Algorithm

We define an algorithm that utilizes a combination of MORL and a cosine annealing schedule to optimize hyperparameters. The algorithm is as follows:

**Algorithm 1: MORL Algorithm.**

**Input:** number of configurations $n$; maximum resource $r$; initial learning rate $l$; reduction factor $\eta$ (default: 3); minimum exponent $s_{\min}$ (default: 2); $H = \text{suggest\_hyperparameters}(n)$

for $h \in H$ do
  $G_h = \text{create\_network}(h)$
  for $s \in \{s_{\min}, s_{\min} + 1, \ldots, \log_\eta(r)\}$ do
    $e_{\text{start}} = \text{get\_start\_epoch}(s)$
    $e_{\text{end}} = \text{get\_end\_epoch}(s, r)$
    for $e \in \{e_{\text{start}}, e_{\text{start}} + 1, \ldots, e_{\text{end}}\}$ do
      $l_e = \frac{l}{2}(1 + \cos \pi e/(e_{\text{end}} - e_{\text{start}}))$
      $G_h = \text{train\_an\_epoch}(G_h, l_e, h)$
    end
    $P_h = \text{compute\_performance}(G_h, h)$
  end
  $H = \text{top}_k(H, P, \eta)$ // promote top $1/\eta$ configurations to next round
end

In this algorithm, we first create a network $G_h$ for each hyperparameter configuration $h$. Then, for each configuration, we compute the learning rate $l_e$ for each epoch using a cosine annealing schedule. The network is trained for each epoch, and the performance is computed. Finally, the top $1/\eta$ configurations are promoted to the next round.

### 3.2. Experiments

To verify the effectiveness of the proposed MORL algorithm, we conducted experiments on a variety of computer vision tasks, including object classification, transfer learning, and semi-supervised learning. The experiments demonstrated that MORL is quite robust to the choice of hyperparameters and generally works well with various learning rate schedules. This is illustrated in Figure 3.

![Learning Rate Schedule](image)

Figure 3. Illustration of learning rate schedules utilized in object classification task with maximum resource $r$ of 164 epochs. The initial learning rate $l$ is 0.1. For the baseline (orange curve), we follow the schedule used in the original implementation. For MORL, we notate it independently for simplicity of the algorithm. The figure shows two learning rate schedules: one for MORL and the other for the baseline.

**4. Experiments**

In this section, we conduct a comprehensive evaluation on a wide range of computer vision tasks including object classification, transfer learning, and semi-supervised learning to verify the effectiveness of MORL for hyperparameter optimization. We focus on comparison with SHA [16] and Hyperband [22] which are the de-facto standard multi-fidelity optimization methods implemented in widely adopted HPO frameworks [1, 24].

Our search space consists of learning rate $l$, weight decay $\omega$, momentum $1 - m$ and batch size $b$ where $l, \omega$ are sampled from log $[10^{-6}, 10]$, $m$ from log $[10^{-6}, 1]$, and $b$...
from [16, 256]. While there exist various tasks that include task-specific hyperparameters, we believe our search space would serve as a good starting point that could be applied to a wide range of tasks. For all experiments, one unit of resource corresponds to one epoch and the maximum resource $r$ is set to the training epochs specified in the original implementation of each task. For each HPO experiment, we allocate a budget of $64r$ which corresponds to 64 different runs of an experiment when early-stopping is not applied.

### 4.1. Object Classification

We first evaluate MORL on object classification with publicly available implementation\(^1\) of classification networks and training schemes for a fair comparison with the human baseline. For the baseline, we use the hyperparameter configuration suggested in the original implementation. We adopt three different datasets: CIFAR-10/100 datasets [19] comprising 50,000 training and 10,000 test images of 10/100 object classes and Tiny ImageNet dataset [30] containing 100,000 training and 10,000 validation images of 200 object classes. While the original images of Tiny ImageNet consist of $64 \times 64$ pixels, they are downsized to $32 \times 32$ pixels for the consistency with CIFAR dataset training process.

We follow the standard practice for data augmentation [12] where each image is zero-padded with 4 pixels then randomly cropped to the original size and evaluation is performed on the original images. We utilize SGD optimizer with an initial learning rate of 0.1, a weight decay of 5e-4, a momentum of 0.9 and a batch size of 128 on a single GPU for the baseline configuration. Following the original implementation, the networks are trained for 164 epochs and a step learning rate schedule is applied where the initial learning rate is divided by 10 at 81 and 122 epochs. The illustration of the learning rate schedule for the baseline and MORL is shown in Figure 3.

**Comparison of multi-fidelity optimization methods.**

We first evaluate MORL compared to other multi-fidelity optimization methods with varying initial resources. Figure 4 illustrates the performance of multi-fidelity optimization methods along with the baseline and random search on CIFAR-100 with VGG-11 [31] and AlexNet [20] architecture. Each SHA with $s_{\min} = 0$ and $s_{\min} = 2$ represents the most and intermediate aggressive method of SHA, Hyperband performs grid search over possible values of $s_{\min}$, and random search performs full training without early-stopping. Throughout the optimization process except for the very beginning, MORL exhibits considerably higher performance compared to other methods. In the very early stage, SHA($s_{\min} = 0$) and Hyperband achieve moderate performance by exploring more configurations and early-stopping hopeless configurations which usually exhibit near random-guessing accuracy. However, they slowly improve the performance whereas MORL improves at a rapid pace. For the rest of the experiments, we utilize SHA($s_{\min} = 2$) for the SHA method which exhibits better characteristics than SHA($s_{\min} = 0$).

**Scalability to different datasets.** We demonstrate the effectiveness of MORL with respect to various datasets of CIFAR-10/100 and Tiny ImageNet. Table 1 compares the top-1 accuracy with the average and the 95% confidence interval over 5 repetitions. MORL outperforms all other methods in all of the datasets. Whereas other multi-fidelity optimization methods often fail to surpass the hand-tuned baseline given a limited budget and even exhibit lower performance than random search, MORL consistently boosts the performance of the baseline by a significant margin. It

---

\(^1\)https://github.com/bearpaw/pytorch-classification
Table 1. Top-1 accuracy (%) on CIFAR10/100 and Tiny-ImageNet with VGG-11. The reported results are the average and the 95% confidence interval over 5 repetitions. MORL outperforms hand-tuned baseline and other competing methods across varying datasets, and obtains relatively narrow confidence interval.

<table>
<thead>
<tr>
<th></th>
<th>CIFAR-10</th>
<th>CIFAR-100</th>
<th>Tiny-ImgNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>70.99±0.26</td>
<td>48.21±0.44</td>
<td>91.46±0.52</td>
</tr>
<tr>
<td>Random</td>
<td>69.79±1.84</td>
<td>47.63±2.66</td>
<td>52.64</td>
</tr>
<tr>
<td>SHA</td>
<td>70.27±1.47</td>
<td>48.34±1.95</td>
<td>66.86</td>
</tr>
<tr>
<td>Hyperband</td>
<td>70.11±1.09</td>
<td>47.99±1.75</td>
<td>94.46</td>
</tr>
<tr>
<td>MORL</td>
<td>72.74±0.42</td>
<td>50.96±0.33</td>
<td>75.22</td>
</tr>
</tbody>
</table>

Table 2. Top-1 accuracy (%) on CIFAR-100 with different network architectures. MORL consistently improves the performance of baseline with a meaningful margin in all tested networks.

<table>
<thead>
<tr>
<th></th>
<th>AlexNet</th>
<th>ResNet-20</th>
<th>ResNet-56</th>
<th>VGG-16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>44.08</td>
<td>68.67</td>
<td>71.54</td>
<td>73.58</td>
</tr>
<tr>
<td>Random</td>
<td>44.63</td>
<td>66.86</td>
<td>69.75</td>
<td>73.84</td>
</tr>
<tr>
<td>SHA</td>
<td>45.67</td>
<td>67.15</td>
<td>70.41</td>
<td>72.48</td>
</tr>
<tr>
<td>Hyperband</td>
<td>45.69</td>
<td>67.31</td>
<td>71.03</td>
<td>72.66</td>
</tr>
<tr>
<td>MORL</td>
<td>46.92</td>
<td>69.51</td>
<td>72.87</td>
<td>76.01</td>
</tr>
</tbody>
</table>

is worth noting that the model acquired by MORL does not introduce any additional weights or computational overhead. The performance gain merely comes from tuning hyperparameters of CNN’s optimization process.

Scalability to various CNN architectures. We further verify the scalability of MORL with respect to diverse CNN architectures including AlexNet, ResNet-20, ResNet-56 and VGG-16 on the CIFAR-100 dataset. As shown in Table 2, MORL consistently outperforms other methods and improves the performance of the baseline in all experiments. Our results demonstrate the efficacy of MORL with respect to varying CNN architectures. While SHA and Hyperband only succeed to improve baseline performance on AlexNet, MORL brings meaningful performance improvement even for the relatively large model such as VGG-16. It further implies the importance of HPO that the performance gain obtained by tuning hyperparameter is not simply substituted by increasing the capacity of CNN architecture.

4.2. Transfer Learning

A wide range of computer vision tasks employs a pre-trained model that is trained on a large-scale dataset such as ImageNet [30]. By utilizing meaningful features learned from a large amount of data, transfer learning has successfully boosted the performance of CNNs in various tasks [35]. In this scenario, it is reasonable to suspect that the problem of slow-starter is probably mitigated as the network converges fast from the pre-trained weights.

We follow the same training policy as our object classification experiment except for the initial learning rate of the baseline which is reduced by a factor of 10, following common practice for transfer learning. We opt for the ImageNet pre-trained VGG-16 network, which is widely adopted in various computer vision tasks [10, 25, 29] and train the network on CIFAR-10/100 and Tiny ImageNet dataset. We report the top-1 validation accuracy in Table 3. As expected, SHA and Hyperband show improved performance with respect to the baseline, whereas they often failed to outperform when trained from scratch as shown in Table 1 and 2. Nevertheless, MORL consistently outperforms other methods by a meaningful margin which suggests that the slow-starting tendency is mitigated but remained in the transfer learning setting as well.

4.3. Semi-Supervised Learning

We finally demonstrate the effectiveness of MORL in semi-supervised learning (SSL) task which jointly learns from a small number of labeled samples and a large number of unlabeled samples. Since considerable amounts of unlabeled images are utilized with the supervision from scarce labeled images, it is crucial to find a good hyperparameter configuration. We make use of pseudo-label based SSL algorithm [2] which outperforms state-of-the-art consistency regularization methods. For a fair comparison, we adopt their official implementation [2] utilizing its default training seed in order to compare different methods under consistent settings and facilitate further research.

The hand-tuned baseline consists of two-stage training. It first trains the network only with the labeled data as a warm-up phase, then finetunes the network with both labeled and unlabeled data. While the warm-up stage stabilizes the training process by initializing unlabeled data with more reliable prediction, it incurs model developers to explore a much larger search space since the hyperparameter for each stage needs to be tuned independently. Furthermore, it requires additional resources to train a model for the first stage. However, if HPO provides better configurations which enable stable training of the second stage, we can omit the

2https://github.com/EricArazo/PseudoLabeling
Table 4. Top-1 accuracy (%) of semi-supervised learning on CIFAR-100 with respect to a varying number of labeled data. While MORL does not include a warm-up stage, it significantly boosts the performance of the baseline which consists of two-stage training.

<table>
<thead>
<tr>
<th></th>
<th>500 labels</th>
<th>1000 labels</th>
<th>2000 labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>29.73</td>
<td>45.71</td>
<td>55.56</td>
</tr>
<tr>
<td>Random</td>
<td>31.19</td>
<td>46.65</td>
<td>54.08</td>
</tr>
<tr>
<td>SHA</td>
<td>32.77</td>
<td>45.21</td>
<td>54.65</td>
</tr>
<tr>
<td>Hyperband</td>
<td>32.56</td>
<td>45.51</td>
<td>54.99</td>
</tr>
<tr>
<td>MORL</td>
<td><strong>35.22</strong></td>
<td><strong>48.52</strong></td>
<td><strong>57.18</strong></td>
</tr>
</tbody>
</table>

Table 5. The best configuration obtained with MORL algorithm in semi-supervised learning experiments. The optimized WD value decreases as it uses more labeled samples. (LR: learning rate, WD: weight decay, MMT: momentum, BS: batch size)

<table>
<thead>
<tr>
<th></th>
<th>LR</th>
<th>WD</th>
<th>MMT</th>
<th>BS</th>
</tr>
</thead>
<tbody>
<tr>
<td>500 labels</td>
<td>0.0242</td>
<td>0.0043</td>
<td>0.5652</td>
<td>92</td>
</tr>
<tr>
<td>1000 labels</td>
<td>0.0311</td>
<td>0.0032</td>
<td>0.6834</td>
<td>87</td>
</tr>
<tr>
<td>2000 labels</td>
<td>0.0238</td>
<td>0.0027</td>
<td>0.3926</td>
<td>84</td>
</tr>
</tbody>
</table>

warm-up stage. Therefore, we exclude the warm-up stage and train the network from scratch for multi-fidelity optimization methods. However, for the baseline, we follow the original two-stage training as the author suggested.

We utilize 13-CNN architecture [3] which is mainly explored in the paper and follow its data processing and optimization setting. Following the original implementation, the network is trained using SGD optimizer with Dropout [33] and Mix-up [40] regularization for 400 epochs where the learning rate is decayed by a factor of 10 at 250 and 350 epoch for the baseline. Table 4 shows the top-1 accuracy of CIFAR-100 dataset with respect to a varying number of labeled samples. Surprisingly, MORL achieves significant performance improvement over the baseline throughout the experiments even without the warm-up stage. It further demonstrates the efficacy of MORL in various settings where it works well with a varying number of labeled samples and strong regularization methods such as Mix-up and Dropout.

We further report the top-performing configuration for varying number of labels in Table 5, where each setting shows a distinct configuration. While it remains challenging to perform precise analysis due to the nature of hyperparameters, we observe that higher weight decay is applied when there exists a small number of labeled images. It conforms to common intuition that more regularization is desired when a small number of data is given as the network tends to overfit. Our results suggest that it is important to tailor hyperparameters in each experiment setting, a process that can be effectively automated by MORL.

5. Ablation Study and Analysis

In this section, we perform an ablation study and analytical experiments to gain an insight into the algorithm design choice and potential extensions of MORL. We follow the same procedure of object classification experiment on CIFAR-100 in section 4.1 for the empirical study.

5.1. Ablation on Minimum Exponent

We conduct an ablation on minimum exponent $s_{\text{min}}$ which adjusts the trade-off between the total number of configurations and minimum resource given a fixed budget. Since a minimum resource of $\eta^{s_{\text{min}}}$ is allocated to each configuration, small $s_{\text{min}}$ enables to explore more configurations by aggressive early-stopping whereas large $s_{\text{min}}$ enables more precise low-fidelity approximation by allocating more resources to each configuration.

As shown in Figure 5, while $s_{\text{min}} = 0$ achieves moderate performance in the early phase by exploring order-of-magnitude more configurations, it shows a slow rate of improvement due to imprecise low-fidelity approximation. On the other hand, it exhibits comparably low performance throughout the optimization process when $s_{\text{min}}$ is large because only a small number of configurations can be explored. This result indicates the importance of allocating adequate initial resource and exploring various configurations. While $s_{\text{min}} = 2$ works generally well throughout the various tasks, datasets and architectures, one might combine MORL with Hyperband which automates the choice of $s_{\text{min}}$. We found that Hyperband works well with MORL, showing slightly worse performance than $s_{\text{min}} = 2$ (72.8 vs 72.6).

5.2. Integration with Bayesian Optimization

Bayesian optimization adaptively suggests configuration to evaluate given observations of hyperparameter configuration and its corresponding performance. By incorporating the prior observations into fitting a probabilistic model, it enables to explore more probable candidates compared to
random search [4, 6]. However, vanilla Bayesian optimization typically requires a lot of resources to achieve satisfying performance as it needs to train a full model to get one observation. Previous work [8] showed that Bayesian optimization can be successfully combined with multi-fidelity optimization to benefit from the advantage of each method.

Among various Bayesian optimization algorithms, we choose Tree-Structured Parzen Estimator (TPE) [4] which models density function based on good and bad observations. It is widely adopted because it is relatively fast and robust to high dimension. We borrow a TPE implementation of the popular HPO framework [1] with the multivariate option which considers the dependencies among hyperparameters. As shown in Figure 6, TPE exhibits significantly better performance compared to random search as it suggests more plausible configurations. However, it shows limited performance as it only evaluates a small number of configurations. When integrating MORL with TPE, it consistently shows the best performance throughout the HPO process. It implies that the advantage of MORL is orthogonal to Bayesian optimization methods and they complement each other by exploring more configurations and adaptively suggesting configurations. We summarize top-performing configurations in Table 6. The top-5 high-performing configurations are diversely distributed, not similar to each other, demonstrating the importance of exploration in HPO.

### 5.3. Various Learning Rate Schedule

We finally demonstrate the scalability of MORL with respect to various learning rate schedules. Specifically, we further examine MORL with step, cyclical and linear learning rate which are widely used and adopted in popular deep learning framework [28]. Step learning rate decays the learning rate by a given factor once the epoch reaches the specified milestone, cyclical learning rate [32] monotonically increases then decreases learning rate within a cycle and linear learning rate linearly decreases learning rate given cycle. Each learning rate schedule is condensed to fit each round of promotion and restarted every round. For the step learning rate, we follow the ratio of decaying epoch in section 4.1. Table 7 shows the results of various learning rate schedules combined with MORL. Throughout the all tested learning rate schedules, MORL boosts the performance of the baseline by a considerable margin. Furthermore, the performance gap among varying schedules is within the allowable range. These results show that the advantage of MORL is not restricted to a certain learning rate schedule but can be extended to various schedules.

### 6. Conclusion

In this work, we present Multi-fidelity Optimization with a Recurring Learning rate (MORL) which enables precise low-fidelity approximation among hyperparameter configurations. By incorporating the learning rate schedule into the multi-fidelity optimization process, MORL achieves to search outstanding configuration within a practical budget. Our extensive experiments on a wide range of settings demonstrate the effectiveness of MORL in hyperparameter optimization. While previous works often fail to improve hand-tuned hyperparameters, MORL successfully outperforms human experts by a significant margin. Furthermore, we verify that the proposed method is orthogonal to Bayesian optimization and applicable to various learning rate schedules. We hope that our work contributes to automating hyperparameter tuning process and pushing the boundary of CNN’s performance in various applications.
References

[3] Ben Athiwaratkun, Marc Finzi, Pavel Izmailov, and Andrew Gordon Wilson. There are many consistent explanations of unlabeled data: Why you should average. ICLR, 2019. 7
[10] Ravi Garg, Vijay Kumar Bg, Gustavo Carneiro, and Ian Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In ECCV, 2016. 6


