AME: Attention and Memory Enhancement in Hyper-Parameter Optimization

Nuo Xu[a,b], Jianlong Chang[c], Xing Nie[a,b], Chunlei Huo *[a,b], Shiming Xiang[a,b] and Chunhong Pan[a,b]

[a]NLPR, Institute of Automation, Chinese Academy of Sciences, Beijing, China
[b]School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China
[c]Huawei Cloud & AI, Beijing, China

nuo.xu@nlpr.ia.ac.cn, jianlong.chang@huawei.com, niexing2019@ia.ac.cn, {clhuo, smxiang, chpan}@nlpr.ia.ac.cn

Abstract

Training Deep Neural Networks (DNNs) is inherently subject to sensitive hyper-parameters and untimely feedbacks of performance evaluation. To solve these two difficulties, an efficient parallel hyper-parameter optimization model is proposed under the framework of Deep Reinforcement Learning (DRL). Technically, we develop Attention and Memory Enhancement (AME), that includes multi-head attention and memory mechanism to enhance the ability to capture both the short-term and long-term relationships between different hyper-parameter configurations, yielding an attentive sampling mechanism for searching high-performance configurations embedded into a huge search space. During the optimization of transformer-structured configuration searcher, a conceptually intuitive yet powerful strategy is applied to solve the problem of insufficient number of samples due to the untimely feedback. Experiments on three visual tasks, including image classification, object detection, semantic segmentation, demonstrate the effectiveness of AME.

1. Introduction

Hyper-Parameter Optimization (HPO) [53] is a crucial subfield in Automatic Machine Learning (AutoML), which is formulated as a bi-level optimization problem. Recently, the rise of deep learning has promoted giant development of machine learning and computer vision, but it also places higher requirements on computing resources. The optimization of large-scale neural networks often takes days or even weeks and a large number of GPUs to train, thus manual tuning of hyper-parameters has gradually become expensive. At the same time, the networks are highly sensitive to the choice of hyper-parameters. Improper hyper-parameters directly lead to the failure of training, *e.g.*, gradient explosion. In addition, when training modern neural networks, a large number of hyper-parameters are required to be set, including architecture hyper-parameters (*e.g.*, network depth and types), optimization hyper-parameters (*e.g.*, learning rate, batch size), and regularization hyper-parameters (*e.g.*, weight decay), which result in a huge search space. Moreover, the challenge of HPO varies greatly in various subfields of machine learning. In the field of computer vision, hyper-parameters in detection and segmentation are more sensitive than classification. Therefore, a practical modern HPO algorithm must be able to easily handle the selection of several to dozens of hyper-parameters for different tasks within an acceptable time.

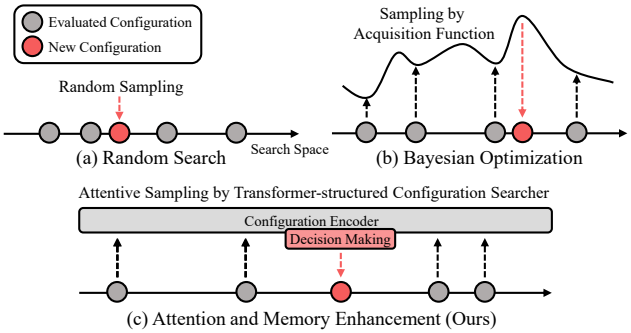

Figure 1. Comparison of different configuration searchers. (a) Random Search. The selection of the new configuration is independent of other evaluated configurations. (b) Bayesian Optimization. Under the given distribution assumption, the new configuration is finally obtained by acquisition function which models the relationship between evaluated configurations. (c) Attention and Memory Enhancement (AME). The relationship is captured by attentive sampling without distribution assumptions, and is employed for the prediction of all types of new configurations.

*Corresponding author

Mainstream HPO algorithms consist of two parts, trial scheduler and configuration searcher. Among them, the scheduler is responsible for the allocation of computing resources. Specifically, it is capable of judging when to start a new trial, and whether to suspend or terminate the trials according to the corresponding performance and running time. The searcher is in charge of the proposals of new hyper-parameter configuration. For instance, the simplest case is that the new configuration is in a position to be obtained by random search [3, 23, 25, 31, 32] (*see* Fig. 1(a)), but the trials are independent, *i.e.*, the relationship between each other is not considered, so it is extremely unstable in the huge search space. Using Bayesian Optimization (*see* Fig. 1(b)) to build an acquisition function by the evaluated configurations is another type of searcher [2, 12, 20, 48, 49]. Although this type of searcher selects the new hyper-parameter to be evaluated in an informed manner, it is limited by strong assumptions, *e.g.*, assuming that the distribution obeys the Gaussian Process. Also, the modeling is so complicated that is more suitable for the optimization with a low-dimensional search space. In addition, the searcher based on Evolutionary Algorithms [21, 22, 37, 43] is sometimes time-consuming and is not able to cope with hyper-parameters that can not be inherited, mutated or hybridized, such as network depth and types.

In order to maximize the potential of machine learning models, and to select appropriate hyper-parameters more efficiently and stably in the search space, this paper proposes a new configuration searcher based on Deep Reinforcement Learning (DRL) [41,46] and Transformer [42,50] (*see* Fig. 1(c)). Our searcher enhances the ability to capture the relationship between different configurations through multi-head attention and memory mechanism. Combined our searcher with the parallel trial scheduler ASHA [32], Attention and Memory Enhancement (AME) is proposed. AME actively encourages searcher to generate high-performance configurations, and penalizes configurations that reduce performance. The main contributions are as follows:

- A transformer-structured configuration searcher is designed under reinforcement learning. Based on the novel searcher, an efficient parallel HPO model AME is proposed, which is sufficient to optimize all types of hyper-parameters, without distribution assumptions.
- AME is capable of learning both the short-term and long-term relationships to achieve attentive configuration sampling, and effectively locating high-performance configurations in a huge search space.
- Bootstrap is applied to solve the insufficient number of samples caused by the difficulty of sample acquisition. This makes the online training of searcher efficient.
- Experiments demonstrate the efficiency of AME on three vision tasks, including image classification, object detection, and semantic segmentation.

2. Related Work

2.1. Hyper-parameter Optimization

There are two mainstream HPO methods, Multi-fidelity Optimization and Black-box Optimization, optimized for trial scheduler and configuration searcher respectively.

Multi-fidelity Optimization. This is an optimization technique aimed at reducing evaluation costs for scheduler, by obtaining a large number of cheap low-fidelity evaluations and a small number of expensive high-fidelity evaluations. It mainly includes two methods: Bandit-based Algorithm (*e.g.*, SHA [23, 25], Hyperband [31], ASHA [32], BOHB [12], BOSS [18]) and Early-stopping Algorithm (*e.g.*, median stopping [14], modeling learning curve [9,26]). The former is a trade-off between exploration and exploitation, while the latter terminates the trials with poor performance in time. In addition, TSE [17] learns the low-fidelity correction predictor efficiently by linearly combining a set of base predictors. AABO [39] adopts BOSS in detection for adaptively searching for the optimal setting of anchor boxes.

Black-box Optimization. This is a way to learn the mapping $f : x \rightarrow y$ between input x and output y, ignoring the internal mechanisms. The easiest way is Grid Search [51], *i.e.*, exhaustive search, which is only suitable for a small search space. Overall, there are four methods suitable for modern neural networks: Model-free Methods (*e.g.*, Grid Search, Random Search [3, 23, 25, 31, 32], OATM [57]), Bayesian Optimization (*e.g.*, BO-GP [48, 49], SMAC [20], BO-TPE [2], BOHB [12], distBO [30], AHGP [35], Dragonfly [24]), Evolutionary Algorithms (*e.g.*, PBT [22], PB2 [43], GA [21], PSO [37]) and Gradient-based Methods [1, 38, 40, 44, 47]. The advantages and disadvantages of the first three have been discussed in Sec. 1. Recently, Gradient-based Methods have shown better efficiency. However, they only enable the differentiable hyper-parameters to update, such as weight decay, and can not directly update non-differentiable ones, such as batch size.

2.2. Deep Reinforcement Learning

DRL emphasizes how the agent acts based on the environment to obtain the maximum cumulative reward. As a pioneering work, PPO [46] surpasses other existing algorithms in performance. Recently, GTrXL [42] employs the transformer-structured agent to capture long-term and short-term memory for the first time. Other DRL-based AutoML methods generally learn the configuration in their respective tasks, *e.g.*, network architecture [61], compression rate [16], and augmentation strategies [7], directly from input images. Learning from the original images with sparse knowledge is expected to result in inefficiency. Our AME learns the relationship between evaluated configurations to select new ones. In addition, AME overcomes the problem of insufficient samples with the help of bootstrap.

2.3. Vision Tasks

In this paper, three vision tasks are set up for experiments: image classification, object detection, semantic segmentation. Currently popular networks for image classification mainly include ResNet [15], ResNeXt [52], ResNeSt [55], and so on. These networks serve the backbones of other visual tasks. The goal of object detection is to find all the objects of interest in the image and determine their location and size. Mainstream detectors include FasterRCNN [45], CascadeRCNN [4], RetinaNet [34], RepPoints [54], FoveaBox [27], FSAF [60], ATSS [56]. Semantic segmentation is pixel-level classification, which understands images from pixels. Popular segmentors include PSPNet [58], PSANet [59], CCNet [19], DANet [13], DeepLabv3+ [6], FCN [36], GCNet [5]. In both detection and segmentation, the choice of head is crucial. Different heads have varying sensitivity to hyper-parameters, *e.g.*, learning rate, so it is challenging to comprehensively consider the full configurations to optimize.

3. Attention & Memory Enhancement in HPO

3.1. Hyper-Parameter Optimization

Before introducing our approach, we first briefly review HPO. In HPO, both parameters ω and hyper-parameters h need to be optimized, with the non-differentiable nested relationship between them. Therefore, HPO is formulated as a bi-level optimization problem:

$$\begin{aligned} &\min_{h} \mathcal{L}(h, \omega^*, \mathbb{D}_{val}) \\ &\text{s.t. } f_h(\omega^*) \leftarrow f_h(\omega^0, \mathbb{D}_{train}),\ h \in \mathcal{H}. \end{aligned} \tag{1}$$

where $\mathcal{L}$ is the objective function, ω^0 and ω^* are the initial and final parameters of this network, $\mathbb{D}_{train}$ and $\mathbb{D}_{val}$ are the training and validation dataset, h is one hyper-parameter configuration sampled from the search space $\mathcal{H}$, and f_h is the neural network with hyper-parameter set to h. The simplest Grid Search is to train networks with the complete set of hyper-parameters to converge, and then select the optimal. Due to the low efficiency, how to accelerate the search process becomes the most significant issue in HPO. The mainstream research direction is to improve the two components of HPO, trial scheduler and configuration searcher.

Trial Scheduler. Trial scheduler is in charge of the allocation of resources (*see* Fig. 2(a,b)). It is capable of judging whether to terminate or start trials according to the corresponding performance. For example, ASHA [32] divides the training process of each trial into multiple rungs t, which is adopted by our AME:

$$\{\omega^t | t = 0, 1, \cdots, \lfloor \log_\eta (R/r) \rfloor\} \tag{2}$$

where R and r are the maximum and minimum budget (*e.g.*, epoch or iteration) for one trial, and η is the reduction factor. As long as the evaluation indicator of one trial in a certain rung t is greater than the dynamically updated threshold, this trial is promoted to the next rung for training. Besides, trials with poor performance are required to be terminated in time. Therefore, the total number of trials starts from n and gradually declines in the ratio of $1/\eta$ in each rung.

Configuration Searcher. Configuration searcher chooses better hyper-parameter configurations for network training (*see* Fig. 2(a,c)) by building a sampling function $g(\cdot)$:

$$\begin{aligned} h_{new} = g(\{\hat{h}_i | i = 1, 2, \cdots, k\}), \\ \hat{h}_i \in \mathcal{H} \times \mathbb{R}^1,\ h_{new} \in \mathcal{H}. \end{aligned} \tag{3}$$

where h_{new} is the new hyper-parameter configuration to be evaluated, $\hat{h}_i$ refers to the configuration with evaluation indicator, and k is the number of input configurations. The suggestion of new configurations is a notoriously hard problem, mainly owing to the following two difficulties:

- The search space $\mathcal{H}$ is huge, and relationships between the configurations $\{\hat{h}_i\}$ are difficult to be captured.
- The evaluation is time-consuming, *i.e.*, the evaluated configurations $\{\hat{h}_i\}$ are difficult to be obtained.

Although random search is fast, it is unstable because it ignores these difficulties. Bayesian-optimization-based algorithms require prior strong distribution assumptions. Evolutionary-algorithm-based and gradient-based methods are only able to deal with specific hyper-parameters.

Reinforced HPO. The recommendations for selecting hyper-parameters are able to be modeled under the framework of reinforcement learning, since DRL is naturally suitable for decision-making without explicit annotations. Searcher is equivalent to the agent, whose task is to learn a series of state-to-action mappings (*see* Eq. (3)) based on the reward. State ($\mathcal{S}$) refers to the combination of evaluated configurations $\{\hat{h}_i\}$, action ($\mathcal{A}$) refers to the new configuration h_{new} that the agent picks from search space, and reward ($\mathcal{R}$) is the evaluation of action. However, Reinforced HPO is still subject to the two difficulties that HPO has. Our Attention and Memory Enhancement (AME) captures the relationships with the help of attentive sampling, and applies bootstrap to solve the problem of lack of samples.

3.2. AME via Attentive Sampling

Fully Connected network (FC) with weak learning ability is not able to generalize the relationship between configurations well. Therefore, multi-head attention and memory mechanism are introduced to enhance training through attentive sampling. Intuitively, there is uniform continuity in the mapping between the search space and the corresponding evaluation indicators, *i.e.*, the configurations around the configuration with high performance tend to be high-performance. For example, PBT [22] applies *small disturbances* generated by random noise to find a better configuration. The application of multi-head attention predicts

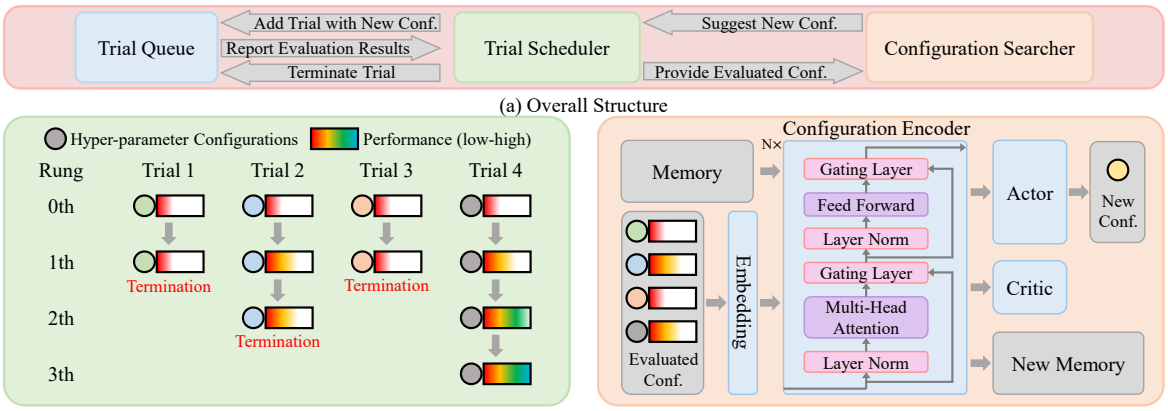

Figure 2. Pipeline of our AME. (a) Overall Structure. Trial Scheduler is responsible for the start, termination and suspension of trials, and collects the evaluation results. Configuration Searcher is in charge of proposing new configurations (conf.). (b) Trial Scheduler. This is an example of vanilla ASHA. A total of n trials are required to be run on limited hardware devices in turn. Every trial is evaluated in each rung, with the low-performance ones terminated. In the sequential mode, the selection of candidates in each rung does not start until the training and evaluation of all trials are completed; in the parallel mode, they are carried out simultaneously. (c) Configuration Searcher. Gated Transformer-XL (GTrXL) is employed to model the relationship between different configurations for attentive sampling. Multi-head attention and memory mechanism capture the short-term and long-term relationships between evaluated configurations, respectively. After each new configuration is evaluated, configuration searcher will be trained, *i.e.*, it is a process of online training, which is parallel to the training of trials. With continuous training, searcher gives more and more reliable suggestions.

the new configuration by weighting the evaluated configuration. The weighting of configurations is able to generate new configurations by assigning higher weights to high-performance configurations, and explore the search space through other configurations at the same time. The introduction of the memory mechanism allows current prediction to influence subsequent ones, so that getting the suggestion of new configuration not limited by current input.

Network Structure. In order to capture the relationships between evaluated configurations and better choose new configurations, Gated Transformer-XL (GTrXL) [42] is chosen as the searcher. GTrXL stabilizes training with a reordering of the layer normalization coupled with the addition of a gating mechanism. Under this new architecture (Fig. 2(c)), the searcher is in a position to simultaneously capture both the long-term and short-term relationships Y_l by memory mechanism and multi-head attention to achieve attentive sampling $h_{new} = g_{ag}(\{\hat{h}_i\})$ (Eq. (3)):

$$
\begin{aligned}
&X_0 = Concat(\{Embedding(\hat{h}_i)|i = 1, 2, \cdots, k\}),\\
&X_l^c = LN(X_l), X_l^{cm} = Concat(X_l^c, X_l^m),\\
&Y_l = MHAttention(X_l^c W^Q, X_l^{cm} W^K, X_l^{cm} W^V),\\
&X_{l+1} = GRUGating(Y_l, X_l),\ l = 0, 1, \cdots, N-1,\\
&h_{new} = Actor(X_N),\ A = Critic(X_N).
\end{aligned} \tag{4}
$$

where N refers to the total number of multi-head attention blocks, W^Q, W^K, W^V are learnable matrixes, X_l^m is the memory matrix, and A is the advantage value administered to assist the training of actor. Transformer-XL [8] introduces the memory mechanism to give configuration encoder the ability to capture long-term dependence, which is similar to the hidden state in RNN. The initial value of the memory matrix X_l^m is zero, and it is updated in the form of $X_l^m = X_{l+1}$. The adoption of gating layer is for stabilizing the training of DRL [42]. Actor-Critic architecture is performed for decision making.

Feature Extraction. The features of discrete hyper-parameters are extracted as one-hot vectors, while continuous hyper-parameters need to be discretized first, and then represented by one-hot vectors. The features extracted from different kinds of hyper-parameters are concatenated together to form hyper-parameter configurations h. In addition, indicators to measure the performance of models (*e.g.*, Accuracy, mIoU, mAP. Normalized to [0, 1]) need to be added to the features as the last dimension. Each time a fixed number of evaluated configurations $\hat{h}$ are input for decision making. After input to the Embedding layer, they are converted from discrete vectors to continuous ones.

3.3. Optimization by Bootstrap

Deep Reinforcement Learning requires a large number of training samples to drive. Since the training of each trial is time-consuming and the feedback of evaluation results is not timely, there is a lack of samples for training, which is the reason why DRL is rarely adopted in HPO. It is inefficient to learn one-to-one mapping between images and configurations by imitating NAS [61], which is limited by the efficiency of sampling. Learning the many-to-one mapping from modeling the relationship between different configurations to propose suggestions is another more efficient way

(*see* Eq. (3)). The combination of multiple configurations makes the application of bootstrap natural and reasonable.

Bootstrap and Random Strategy. Bootstrap is a uniform sampling with replacement from the given dataset, which generates enough configurations for training from a small number of evaluated configurations. Since training the agent is inseparable from a lot of trials-and-errors, bootstrap increases the number of attempts and better overcomes the high variance problem in DRL [46]. On the other hand, during the training, generating a new configuration by agent does not return the validation in time because the new configuration may not have been evaluated. Random strategy is used to solve this problem. As the name implies, the action is not given by the agent network, but is selected from the evaluated configurations randomly, just like any configuration in the state. The number of samples is increased by bootstrap and random strategy on another level.

Reward Function. The design of reward is related to the evaluation results of each trial. Evaluation indicators of the same trial in different rungs are unequal, so it may not be appropriate to directly employ indicators as reward. The difference between evaluation indicators is performed to construct reward function:

$$\mathcal{R} = clip(P_{\mathcal{A}} - \max\{P_s | s \in \mathcal{S}\}, -M, M). \quad (5)$$

where $P_{\mathcal{A}}$ is the evaluation indicator (Normalized to [0, 100]) to the new configuration in action, $\max\{P_s|s \in \mathcal{S}\}$ is the maximum evaluation indicator to the configurations in state, and M is a constant threshold to prevent reward from being too large or too small. Note that the configurations disseminated to combine ($\mathcal{S}$, $\mathcal{A}$) are sampled from the evaluated configurations set in the same rung. Reward function encourages the agent to actively generate new configurations that exceed all input configurations in performance, and inhibits generation of low-performance ones.

Online Training. Analogous to Bayesian Optimization, Reinforced HPO can also be modeled as a process of online training. Once trial scheduler gets the evaluation results, configuration searcher updates its parameters based on the new evaluated configurations. It is worth mentioning that the suggestions of the new configuration are able to be carried out at the same time as the training of agent. In order to strike a balance between exploration and exploitation, the choice of configurations is random during inference and training. The way of taking the top-k to predict leads to insufficient exploration. During training, the loss function of the agent $\mathcal{L}_{ag}$ adopts the form of PPO [46]:

$$\mathcal{L}_{ag} = \frac{1}{N}\sum_{i=1}^{N}\left[\min(r_i A_i, clip(r_i, 1-\varepsilon, 1+\varepsilon)A_i)\right]. \quad (6)$$

where $r_i = \frac{\pi(\mathcal{A}_i|\mathcal{S}_i)}{\pi_{old}(\mathcal{A}_i|\mathcal{S}_i)}$ and $A_i = A(\mathcal{S}_i, \mathcal{A}_i)$, ε is a constant, and A_i is the advantage value, calculated by the critic. PPO hopes to ensure the monotonic improvement of strategy π after the network parameters are updated with a few batches, while keeping the difference in the probability distribution of the old and new strategies within a certain range.

Algorithm 1 Attention and Memory Enhancement (AME)

Input: Configuration Search Space $\mathcal{H}$, Evaluated Configuration Set $\mathcal{H}_E$, Unevaluated Configuration Set $\mathcal{H}_U$, Configuration with Evaluation Indicator $\hat{h}$.
Output: New Configuration h

function CONFSEARCHER($\cdot$) *// Reward $\mathcal{R}$, Action $\mathcal{A}$, State $\mathcal{S}$, The number of input conf. k, A constant ρ.*
  if Need a new configuration **then**
    if $|\mathcal{H}_E| \leq \rho k$ ($\rho \geq 1$) **then**
      Randomly sample h from $\mathcal{H}$. Add h to $\mathcal{H}_U$.
    else
      Randomly sample $\hat{h_1}, \hat{h_2}, \cdots, \hat{h_k}$ from $\mathcal{H}_E$.
      $h = g_{ag}(\{\hat{h_1}, h_2, \cdots, \hat{h_k}\})$. Add h to $\mathcal{H}_U$.
    end if
  else if Get an evaluated conf. & $|\mathcal{H}_E| > \rho k$ **then**
    Randomly sample $\hat{h_0}, \hat{h_1}, \cdots, \hat{h_k}$ from $\mathcal{H}_E$.
    Calculate $\mathcal{R}$ with h_0 as $\mathcal{A}$, $\hat{h_1}, \hat{h_2}, \cdots, \hat{h_k}$ as $\mathcal{S}$.
    Training $Agent$ with loss $\mathcal{L}_{ag}$.
  end if
end function

3.4. Implementation Details

AME is an asynchronous algorithm that combines Deep Reinforcement Learning with ASHA (*see* Alg. 1). For each trial, trial scheduler sends it to different trial queues according to its current stage. If the hardware device is idle, trial scheduler will select a configuration from the unevaluated configuration set $\mathcal{H}_U$ to run. When the trial has been evaluated, trial scheduler decides whether to terminate it according to evaluation result. There is no need to wait for all trials to end training and evaluation in the current rung to start training in the next rung, *i.e.*, training and evaluation in different rungs are parallel. As for configuration searcher, generating a new configuration is the inference of agent, and the acquisition of each evaluated configuration promotes online model training. If the number of evaluated samples is insufficient, a random strategy will be adopted to generate a new configuration instead of model selection. During the training, bootstrap and random strategy are applied to solve the problem of insufficient training samples. Training samples are not limited to those in the first rung, as long as new configurations are still needed. Since reward function (*see* Eq. (5)) is based on the difference of two indicators, samples obtained in different rungs are enabled to be performed to train the same agent. Note that configurations in each combination must be sampled from the same rung.

Vision Task	Head	Learning Rate	Backbone	Batch Size	Optimizer	Weight Decay
Classification	-	{0.001, [0.005:0.005:0.1]}	ResNet18, 34, 50	[8:4:64]	SGD, Adam, Adamax, Adagrad, Adadelta	{1e-5, [0:5e-5:0.001]}
Detection	CascadeRCNN [4], FoveaBox [27], RetinaNet [34], FasterRCNN [45], RepPoints [54], ATSS [56], FSAF [60]	[0.001:0.001:0.025]	ResNet50 [15], ResNeXt50 [52], ResNeSt50 [55]	[4:2:12]		
Segmentation	GCNet [5], DeepLabv3+ [6], DANet [13], CCNet [19], FCN [36], PSPNet [58], PSANet [59]	[0.001:0.001:0.020]				

Table 1. Search Space. The continuous hyper-parameters are discretized into the form of [start:step:end]. In the $100,000$-level search space, finding the optimal hyper-parameter configuration under limited resources is an extremely difficult problem.

Method \ Task	CIFAR-10 (Cls.)			CIFAR-100 (Cls.)			Stanford Cars (Cls.)			VOC (Det.)			VOC (Seg.)		
	Search (Acc)	Time (day)	Retrain (Acc)	Search (Acc)	Time (day)	Retrain (Acc)	Search (Acc)	Time (day)	Retrain (Acc)	Search (mAP)	Time (day)	Retrain (mAP)	Search (mIoU)	Time (day)	Retrain (mIoU)
PBT [22]	93.5 ± 0.6	1.8	94.3	75.1 ± 0.4	1.8	75.6	87.3 ± 0.5	2.9	88.1	79.1 ± 0.5	4.4	79.6	74.7 ± 0.4	3.1	75.3
PB2 [45]	93.8 ± 0.4	2.1	94.3	75.6 ± 0.3	2.0	76.1	87.4 ± 0.4	3.2	87.8	79.4 ± 0.3	4.7	79.9	74.9 ± 0.3	3.2	75.2
BayesOpt [43]	94.1 ± 0.5	1.0	94.8	75.5 ± 0.3	1.1	75.8	87.0 ± 0.4	**1.6**	87.6	79.3 ± 0.4	2.5	79.7	75.2 ± 0.5	**1.5**	75.9
Dragonfly [24]	94.5 ± 0.4	1.4	95.1	76.8 ± 0.4	1.4	77.4	88.1 ± 0.4	2.6	88.5	80.1 ± 0.3	3.5	80.5	75.7 ± 0.3	2.1	76.2
ZOOpt [36]	94.6 ± 0.3	1.1	95.1	76.5 ± 0.2	1.2	76.8	88.3 ± 0.3	1.8	88.7	80.5 ± 0.3	2.6	81.0	75.9 ± 0.3	1.7	76.3
BO-TPE [2]	93.0 ± 0.7	1.1	93.6	75.4 ± 0.5	1.1	75.9	87.0 ± 0.6	2.1	87.6	79.6 ± 0.6	2.7	80.2	74.7 ± 0.5	1.7	75.2
SMAC [20]	93.3 ± 0.6	1.2	94.0	75.8 ± 0.4	1.2	76.2	87.3 ± 0.5	2.2	88.0	79.8 ± 0.5	2.7	80.3	75.1 ± 0.4	1.8	75.8
Hyperband (HB) [31]	93.2 ± 0.9	2.4	94.2	74.4 ± 1.0	2.4	75.3	86.4 ± 0.9	2.8	87.5	80.2 ± 0.8	4.0	81.2	75.0 ± 0.7	3.3	75.9
BOHB [12]	93.1 ± 0.8	2.3	94.0	76.6 ± 0.5	2.3	77.3	87.1 ± 0.7	2.7	87.9	80.4 ± 0.7	4.2	80.9	75.2 ± 0.6	3.4	75.8
ASHA [32]	93.8 ± 1.0	**0.9**	94.5	75.5 ± 0.8	**1.0**	76.3	87.7 ± 0.7	1.8	88.5	79.9 ± 1.0	**2.4**	80.8	75.5 ± 0.8	**1.5**	76.5
AME (Ours)	$\mathbf{95.5 \pm 0.3}$	1.1	**95.9**	$\mathbf{77.8 \pm 0.2}$	1.1	**78.1**	$\mathbf{89.5 \pm 0.3}$	2.0	**89.9**	$\mathbf{81.2 \pm 0.3}$	2.6	**81.8**	$\mathbf{76.7 \pm 0.3}$	1.6	**77.1**

Table 2. Performance comparison of different searchers. Since PBT, PB2, BayesOpt, Dragonfly, ZOOpt are not able to optimize discrete hyper-parameters (*e.g.*, head, backbone, optimizer and batch size), only the continuous selections of learning rate and weight decay are considered under the default setting of other hyper-parameters (the best configuration given by AME). The average performance and time of search results are shown in *Search* and *Time*, respectively. The results after retraining with the optimal hyper-parameters are shown in *Retrain*. The experiments are set on three tasks: image classification (Cls.), object detection (Det.) and semantic segmentation (Seg.).

4. Experiments

4.1. Datasets and Settings

Datasets. In order to verify the effectiveness of AME, the experiments of image classification are set up on CIFAR-10/100 [29] and Standford Cars [28], with the average Accuracy (Acc Top-1) as the evaluation indicator. CIFAR-10/100 consists of 50,000 training images and 10,000 test images in 10/100 classes, and Standford Cars contains 8,144 training images and 8,041 test images in 196 classes. The experiments of object detection and semantic segmentation are set up on PASCAL VOC [10, 11]. As for detection, VOC0712 consists of 16,551 training images, 4,952 test images in 20 classes, with the mean Average Precision (mAP) as the evaluation indicator. As for segmentation, VOC2012+Aug consists of 13,495 training images, 1,449 test images in 20 classes, with the mean Intersection over Union (mIoU) as the evaluation indicator.

Settings. Our experiments are implemented based on Ray Tune [33], which is a python library for hyper-parameter tuning. Eight Nvidia Tesla V100 GPUs are used in experiments. All experimental results are averaged after repeating several times. For classification, the maximum number of configurations n is set to 500, and the maximum budget R is set to 200 epoches in Sec. 3.1 and Eq. (2). For detection, n is set to 80 and R is set to 20. For segmentation, n is set to 80 and R is set to 36. Besides, reduction factor η is set to 2 in Eq. (2), the minimum budget r is set to 1 epoch in Eq. (2), the number of input configurations k is set to 10 in Eq. (3) and Alg. 1, the number of multi-head attention blocks N is set to 2 in Fig. 2 and Eq. (4), M is set to 5 in Eq. (5), ε is set to 0.2 in Eq. (6), and ρ is set to 1.5 in Alg. 1. As shown in Tab. 1, there are five kinds of hyper-parameters in the search space of classification: backbone, learning rate, optimizer type, weight decay and batch size. The hyper-parameter that is required to be searched in detection and segmentation also includes the head of the network.

4.2. Performance Analysis

The foremost novelties of this paper are to propose a new type of configuration searcher and an efficient training strategy for it. For fair comparison, the performances of various types of searchers are shown in Tab. 2, Fig. 3 and Fig. 4.

Adaptability to Diverse Tasks. As revealed by Tab. 2, AME shows better performance than other HPO algorithms in all three vision tasks. Specifically, the performances of AME reached 95.5%, 77.8%, 89.5% in image classification, 81.2% in object detection, 76.7% in semantic segmentation, which are significantly higher than other HPO algorithms. Compared with object detection and semantic segmentation, gradient back-propagation in the training process of classification is more stable attributable to the simpler optimization goal. This also indicates that the performances of networks in detection and segmentation are more sensitive to the choice of hyper-parameters during the training process. The improvement brought by our algorithm in detection and segmentation does not exceed that in classification. The reason lies in that the value of n in detection and segmentation is smaller, which will be analyzed in Sec. 4.3. In short, our algorithm is in a position to be effectively applied

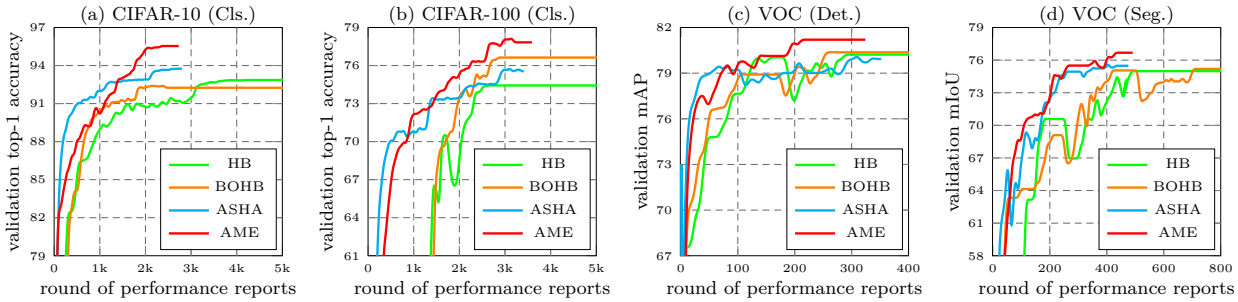

Figure 3. Performance comparison of four algorithms in three vision tasks. The abscissa represents time-related round of performance reports, and the ordinate represents performance. Total number of rounds is simultaneously affected by the maximum number of configurations to be evaluated n and the maximum number of rungs $\lfloor \log_\eta (R/r) \rfloor$ (*see* Eq. (2)).

to visual tasks with various sensitivity levels.

Comparison to Other Algorithms. Although random-search-based methods, *e.g.*, ASHA, are fast, they are unstable (Tab. 2). They have 2.3% (Hyperband) and 1.7% (ASHA) lower in accuracy than AME on classification tasks (CIFAR-10). The reason for the poor performance lies in the fact that the relationship between different configurations is not considered when suggestions of new configurations are given. As a representative of Bayesian Optimization, BOHB (93.1%, 76.6%, 87.1%, 80.4%, 75.2%) only achieves similar performance with random methods, such as Hyperband (93.2%, 74.4%, 86.4%, 80.2%, 75.0%), even if the relationship between configurations is considered. This situation is because of the strong assumptions in Bayesian optimization that have restrictions on the search space. Evolutionary algorithms, including PBT and PB2, are 2.0% and 1.7% lower than AME in accuracy (CIFAR-10), even if the search space is limited, *i.e.*, only learning rate and weight decay are required to be searched.

Efficiency. The performance comparison curves of four methods are plotted in Fig. 3. In terms of speed, the parallel methods (ASHA, AME) are faster than the sequential ones (BOHB, Hyperband). It can be observed that random-search-based ASHA is the fastest in three tasks, followed by AME. This fact proves that the use of bootstrap and random strategy is able to effectively accelerate the training process of reinforcement learning. Moreover, the curve of ASHA fluctuates and the performance is unstable (*see*, Fig. 3(d)), so ASHA may not be able to search for the optimal configuration. In AME, the introduction of attentive sampling has brought a stable performance improvement.

Average Quality of Configurations. The performance of the four algorithms at each rung is visualized in Fig. 4. In order to better focus on the performance difference, a relative form is adopted instead of an absolute form to visualize. It can be seen that the configurations selected by AME have the relatively highest average performance among all rungs. This shows that our proposed transformer-structure agent is capable of locating the high-performance hyper-parameter configurations in a huge search space, by learning the relationship between configurations. It is worth mentioning that the earlier rung during the training process, the greater the advantage of AME, which means that our AME is conducive to the rapid screening of high-performance configurations in the early stage.

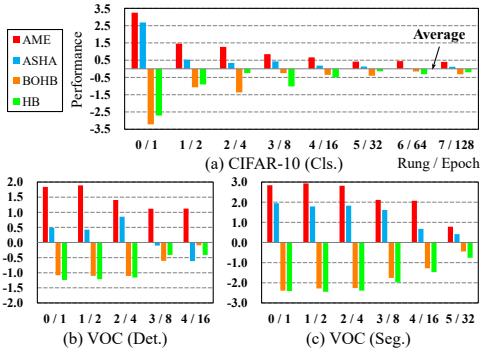

Figure 4. Performance comparison of four algorithms in each rung. Each bar means the average performance of all trials of the corresponding algorithm in the current rung. The zero line refers to the average performance of the four algorithms. The bar charts represent the difference of each algorithm relative to the average.

4.3. Ablation Study

Several ablation experiments are set up on image classification for in-depth analysis of AME, as shown in Fig. 5.

Components of Configuration Searcher. In order to verify the effectiveness of GTrXL in our AME, the components of the configuration searcher are disassembled for experiments, as can be seen in Tab. 3. Experiments prove that the attention module and the memory module improve the accuracy rate of 0.8% and 0.7%, respectively. The experimental results demonstrate that capturing long-term and short-term relationships together facilitates searcher to provide more high-performance configurations for trial scheduler.

Different Configuration Searcher. As shown in Tab. 4, in order to remove the influence of the scheduler, the trained searchers (GTrXL in AME, TPE in BOHB) are taken out separately for comparative experiments with the same input. Although random search does not depend on the input,

Components of Searcher			Scheduler	C10	C100	Cars
Naive FC	Attention	Memory				
✓	✗	✗	ASHA	94.1	76.3	87.9
✓	✓	✗	ASHA	95.1	76.9	88.7
✓	✓	✓	ASHA	**95.5**	**77.8**	**89.5**
✓	✓	✓	HB	95.4	77.4	89.3

Table 3. Experiments on model structure of AME. C10: CIFAR-10. C100: CIFAR-100. Cars: Standford Cars. HB: Hyperband.

Different Searchers	C10	C100	Cars
Input	76.8	60.6	71.5
GTrXL (AME)	**87.2**	**70.5**	**81.4**
TPE (BOHB)	85.9	68.0	79.7
Random	76.1	60.9	71.0

Table 4. Experiments on different searchers with the same input.

Clip	Calculation	C10	C100	Cars
✗	Max	95.2	77.7	89.3
✓	Max	**95.5**	**77.8**	**89.5**
✓	Mean	94.4	76.8	88.7

Table 5. Experiments on reward function in AME.

Number of Input Conf.	$k = 5$	$k = 10$	$k = 20$
CIFAR-10	95.3	**95.5**	94.9
CIFAR-100	77.2	**77.8**	77.0
Standford Cars	88.9	**89.5**	89.0

Table 6. Experiments on number of input configurations in AME.

it is also selected as a benchmark. In the average results of multiple experiments, it can be found that both AME and BOHB can effectively and stably select better new configurations, and AME is higher than BOHB in accuracy.

Different Trial Scheduler. The new searcher proposed in this paper can not only be combined with ASHA, but also with other schedulers (*see* Tab. 3), *e.g.*, Hyperband. Hyperband is a two-layer loop, one layer is to choose different combinations of (n, r), and the other is to perform SHA for each combination. As a sequential method, AME (Hyperband) also achieves comparable performance to AME (ASHA), but the speed is not as fast as AME (ASHA).

Indicator Calculation in Reward Function. The design of the reward function in Eq. (5) is crucial. Therefore, an ablation experiment on the indicator calculation of $\{P_s|s \in \mathcal{S}\}$ is set, revealed by Tab. 5. Experimental results show that using the maximum is better than the mean of 1.0% accuracy, *i.e.*, if the reward function is designed to be more demanding, it has more accurate guidance on capturing the relationship for attentive sampling. The strong restriction with the maximum as threshold on reward function suppresses the negative effects of poor initialization.

Whether to Clip in Reward Function. Although the application of bootstrap and random strategy helps the agent train efficiently, it also makes the training samples noisy. The setting of M in Eq. (5) is to prevent reward from being too large or too small from negatively affecting the training. The experimental results in Tab. 5 tell us that such truncation is useful. The clip operation in reward function brings performance improvement of 0.2%.

Number of Input Configurations k. As shown in Tab. 6, experiments are set up to determine the optimal number of input configurations for the agent. Experiments prove that the value of k is set to 10 as the most appropriate. Both too few and too many input configurations result in a decrease in performance. The appropriate number of configurations allows the agent to accurately achieve attentive sampling.

Total Number of Conf.	$n = 500$	$n = 200$	$n = 80$
Hyperband	93.2	93.1	93.0
BOHB	93.1	93.4	93.2
ASHA	93.8	93.8	93.9
AME (Ours)	**95.5**	**95.4**	**94.3**

Table 7. Experiments on total number of configurations (C10).

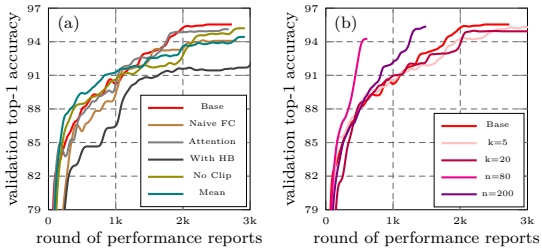

Figure 5. Performance comparison in ablation studies of AME (C10). (a) Results in Tab. 3, Tab. 5. (b) Results in Tab. 6, Tab. 7. Base: AME algorithm with default settings, described in Sec. 4.1.

Total Number of Configurations n. Since the training of searcher in AME is modeled as online learning, the total number of configurations n has a huge impact on it and directly determines whether the training of searchers is sufficient. The results in Tab. 7 show that when n is reduced, the performance of AME drops sharply, and they eventually degenerate into random search. BOHB, Hyperband and ASHA are much less sensitive to changes in n than our algorithm. This also explains why the performance improvements on detection and segmentation of AME are lower than that on classification, revealed by Fig. 3 and Tab. 2.

5. Conclusions

In this paper, an innovative parallel and asynchronous HPO model AME is proposed under the framework of deep reinforcement learning, enhancing the ability to capture both the short-term and long-term relationships to achieve attentive sampling. Bootstrap and random strategy are applied to solve the problem of an insufficient number of samples, which enable the training of searcher to be carried out efficiently. Experiments prove the efficiency of AME on three vision tasks, including classification, detection, and segmentation. Future work will focus on extending AME to more visual and non-visual tasks.

Acknowledgments

This research was supported by the National Key Research and Development Program of China under Grant No. 2018AAA0100400, and the National Natural Science Foundation of China under Grants 62071466, 62076242, and 61976208.

References

[1] A.G. Baydin, R. Cornish, D.M. Rubio, M. Schmidt, and F. Wood. Online learning rate adaptation with hypergradient descent. *ICLR*, 2018. 2

[2] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In *NeurIPS*, 2011. 2

[3] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *JMLR*, 13(2), 2012. 2

[4] Z. Cai and N. Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *CVPR*, 2018. 3, 6

[5] Y. Cao, J. Xu, S. Lin, F. Wei, and H. Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *ICCV*, 2019. 3, 6

[6] L.C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018. 3, 6

[7] E.D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q.V. Le. Autoaugment: Learning augmentation strategies from data. In *CVPR*, 2019. 2

[8] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q.V. Le, and R. Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *ACL*, 2019. 4

[9] T. Domhan, J.T. Springenberg, and F. Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *IJCAI*, 2015. 2

[10] M. Everingham, SM.A. Eslami, L. Van Gool, C.KI. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *IJCV*, 111(1):98–136, 2015. 6

[11] M. Everingham, L. Van Gool, C.KI. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, 2010. 6

[12] S. Falkner, A. Klein, and F. Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. In *ICML*, 2018. 2

[13] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu. Dual attention network for scene segmentation. In *CVPR*, 2019. 3, 6

[14] D. Golovin, B. Solnik, S. Moitra, G. Kochanski, J. Karro, and D. Sculley. Google vizier: A service for black-box optimization. In *SIGKDD*, 2017. 2

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3, 6

[16] Y. He, J. Lin, Z. Liu, H. Wang, L. Li, and S. Han. Amc: Automl for model compression and acceleration on mobile devices. In *ECCV*, 2018. 2

[17] Y.Q. Hu, Y. Yu, W.W. Tu, Q. Yang, Y. Chen, and W. Dai. Multi-fidelity automatic hyper-parameter tuning via transfer series expansion. In *AAAI*, 2019. 2

[18] Y. Huang, Y. Li, Z. Li, and Z. Zhang. An asymptotically optimal multi-armed bandit algorithm and hyperparameter optimization. *arXiv preprint arXiv:2007.05670*, 2020. 2

[19] Z. Huang, X. Wang, L. Huang, C. Huang, Y. Wei, and W. Liu. Ccnet: Criss-cross attention for semantic segmentation. In *ICCV*, 2019. 3, 6

[20] F. Hutter, H.H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *LION*, 2011. 2

[21] F. Itano, M.A.A. de Sousa, and E. Del-Moral-Hernandez. Extending mlp ann hyper-parameters optimization by using genetic algorithm. In *IJCNN*, 2018. 2

[22] M. Jaderberg, V. Dalibard, S. Osindero, W.M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, K. Simonyan, et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017. 2, 3

[23] K. Jamieson and A. Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *AISTATS*, 2016. 2

[24] K. Kandasamy, K.R. Vysyaraju, W. Neiswanger, B. Paria, C.R. Collins, J. Schneider, B. Poczos, and E.P. Xing. Tuning hyperparameters without grad students: Scalable and robust bayesian optimisation with dragonfly. *JMLR*, 21(81):1–27, 2020. 2

[25] Z. Karnin, T. Koren, and O. Somekh. Almost optimal exploration in multi-armed bandits. In *ICML*, 2013. 2

[26] A. Klein, S. Falkner, J.T. Springenberg, and F. Hutter. Learning curve prediction with bayesian neural networks. In *ICLR*, 2017. 2

[27] T. Kong, F. Sun, H. Liu, Y. Jiang, L. Li, and J. Shi. Foveabox: Beyound anchor-based object detection. *IEEE TIP*, 29:7389–7398, 2020. 3, 6

[28] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. In *ICCVW*, 2013. 6

[29] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. *Technical Report*, 2009. 6

[30] H.C.L. Law, P. Zhao, L. Chan, J. Huang, and D. Sejdinovic. Hyperparameter learning via distributional transfer. In *NeurIPS*, 2019. 2

[31] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *JMLR*, 18(1):6765–6816, 2017. 2

[32] L. Li, K. Jamieson, A. Rostamizadeh, E. Gonina, M. Hardt, B. Recht, and A. Talwalkar. A system for massively parallel hyperparameter tuning. In *MLSys*, 2020. 2, 3

[33] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J.E. Gonzalez, and I. Stoica. Tune: A research platform for distributed model selection and training. In *ICMLW*, 2018. 6

[34] T.Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 3, 6

[35] S. Liu, X. Sun, P.J. Ramadge, and R.P. Adams. Task-agnostic amortized inference of gaussian process hyperparameters. In *NeurIPS*, 2020. 2

[36] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 3, 6

[37] P.R. Lorenzo, J. Nalepa, M. Kawulok, L.S. Ramos, and J.R. Pastor. Particle swarm optimization for hyper-parameter selection in deep neural networks. In *GECCO*, 2017. 2

[38] J. Lorraine, P. Vicol, and D. Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *AISTATS*, 2020. 2

[39] W. Ma, T. Tian, H. Xu, Y. Huang, and Z. Li. Aabo: Adaptive anchor box optimization for object detection via bayesian sub-sampling. In *ECCV*, 2020. 2

[40] D. Maclaurin, D. Duvenaud, and R. Adams. Gradient-based hyperparameter optimization through reversible learning. In *ICML*, 2015. 2

[41] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015. 2

[42] E. Parisotto, F. Song, J. Rae, R. Pascanu, C. Gulcehre, S. Jayakumar, M. Jaderberg, R.L. Kaufman, A. Clark, S. Noury, et al. Stabilizing transformers for reinforcement learning. In *ICML*, 2020. 2, 4

[43] J. Parker-Holder, V. Nguyen, and S.J. Roberts. Provably efficient online hyperparameter optimization with population-based bandits. In *NeurIPS*, 2020. 2

[44] F. Pedregosa. Hyperparameter optimization with approximate gradient. In *ICML*, 2016. 2

[45] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: towards real-time object detection with region proposal networks. *IEEE TPAMI*, 39(6):1137–1149, 2016. 3, 6

[46] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 2, 5

[47] A. Shaban, C.A. Cheng, N. Hatch, and B. Boots. Truncated back-propagation for bilevel optimization. In *AISTATS*, 2019. 2

[48] J. Snoek, H. Larochelle, and R.P. Adams. Practical bayesian optimization of machine learning algorithms. In *NeurIPS*, 2012. 2

[49] N. Srinivas, A. Krause, S.M. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *ICML*, 2010. 2

[50] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 2

[51] CF.J. Wu and M.S. Hamada. *Experiments: planning, analysis, and optimization*, volume 552. John Wiley & Sons, 2011. 2

[52] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 3, 6

[53] L. Yang and A. Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, 2020. 1

[54] Z. Yang, S. Liu, H. Hu, L. Wang, and S. Lin. Reppoints: Point set representation for object detection. In *ICCV*, 2019. 3, 6

[55] H. Zhang, C. Wu, Z. Zhang, Y. Zhu, Z. Zhang, H. Lin, Y. Sun, T. He, J. Mueller, R. Manmatha, et al. Resnest: Split-attention networks. *arXiv preprint arXiv:2004.08955*, 2020. 3, 6

[56] S. Zhang, C. Chi, Y. Yao, Z. Lei, and S.Z. Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In *CVPR*, 2020. 3, 6

[57] X. Zhang, X. Chen, L. Yao, C. Ge, and M. Dong. Deep neural network hyperparameter optimization with orthogonal array tuning. In *ICONIP*, 2019. 2

[58] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *CVPR*, 2017. 3, 6

[59] H. Zhao, Y. Zhang, S. Liu, J. Shi, C.C. Loy, D. Lin, and J. Jia. Psanet: Point-wise spatial attention network for scene parsing. In *ECCV*, 2018. 3, 6

[60] C. Zhu, Y. He, and M. Savvides. Feature selective anchor-free module for single-shot object detection. In *CVPR*, 2019. 3, 6

[61] B. Zoph and Q.V. Le. Neural architecture search with reinforcement learning. In *ICLR*, 2016. 2, 4