# On Applications of Multi-Armed Bandit and Bayesian Optimization Approaches for AutoML Problems

by

Shan Lu

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Software Engineering and Intelligent Systems

Department of Electrical and Computer Engineering
University of Alberta

# Abstract

Automated Machine Learning (AutoML) aims to alleviate human efforts and automate the time-consuming and iterative processes in the development of machine learning methods. In this thesis, we study two AutoML tasks: automated Data Augmentation (DA) and Hyperparameter Optimization (HPO). Automated DA searches for optimal data augmentation policies and is a widely used regularization technique for training deep neural networks. However, since early approaches, e.g., AutoAugment, cost thousands of GPU hours, there is a recent trend to investigate low-cost search methods that still yield effective augmentation policies. In this thesis, we propose a novel multi-armed bandit algorithm, named Bandit Data Augment (BDA), to efficiently search for optimal and transferable augmentation policies. We design a reward signal based on each batch training step of neural networks to reduce the evaluation cost of augmentation policies. Moreover, we propose the Evolutionary Pruning algorithm to allocate more search resources on potentially optimal operation pairs, leading to sparse selection of operation pairs and generalizable policies. Extensive experiments demonstrate that BDA can achieve comparable or better performance than previous auto-augmentation methods on a wide range of models on CIFAR-10/100, SVHN and ImageNet benchmarks.

Automated HPO algorithms search for optimal hyperparameter configurations used in machine learning and are key to boosting model performance in reality. Many previous HPO methods are based on Bayesian Optimization. However, Bayesian Optimization requires sequentially exploring many data samples to find promising configurations. In this thesis, we propose an efficient batch HPO algorithm that utilizes a combination of Bayesian Optimization and Covariance Matrix Adaptation Evolution Strategy (CMA-ES) as a hybrid sampler to sample configurations while balancing exploitation and exploration. We also propose an ensemble prediction model consisting of various surrogate models to approximate the objective function more accurately. We conduct our experiments on 20 HPO datasets from recommendation system scenarios. Our approach ranked the 4th and 7th places in the training and tournament stages of the Automated HPO Contest in QQ Brower 2021 AI Algorithm Competition at CIKM 2021 AnalytiCup, respectively.

# Preface

Parts of this work were submitted to IEEE Transactions on Neural Networks and Learning Systems and the Web Conference.

# Acknowledgements

I would like to thank all people who contributed to the work described in this thesis. First and foremost, I would like to express my gratitude and appreciation to my supervisor, Dr. Di Niu. His support, comprehensive guidance, and keen scientific insights in this research field help me to learn how to be a good researcher and make my graduate study an inspiring experience for me. Additionally, I would like to thank Mingjun for giving me helpful suggestions and discussions during my research.

I would also like to thank all my friends for supporting me during my graduate study here. I recognize all my friends in the lab, Shengyao, Yakun, Jiuding, Jerry, Mingjun, Qikai, Tong, and Bang. I am lucky to have met you here and thankful for your supports.

Finally, I would like to express my gratitude to my family members. I thank my parents for their constant love, support and encouragement.

# Table of Contents

# List of Tables

# List of Figures

# Abbreviations

| | |
|---|---|
| **AA** | AutoAugment. |
| **AdvAA** | Adversarial AutoAugment. |
| **AutoML** | Automated Machine Learning. |
| **BDA** | Bandit Data Augment. |
| **BO** | Bayesian Optimization. |
| **BOwLS** | Bayesian Optimization with Local Search. |
| **CMA-ES** | Covariance Matrix Adaptation Evolution Strategy. |
| **DA** | Data Augmentation. |
| **DADA** | Differentiable Automatic Data Augmentation. |
| **EI** | Expected Improvement. |
| **ET** | Extra Tree. |
| **FastAA** | Fast AutoAugment. |
| **GBT** | Gradient Boosting Tree. |
| **GP** | Gaussian Process. |
| **HPO** | Hyperparameter Optimization. |

| | |
|---|---|
| **LCB** | Lower Confidence Bound. |
| **MetaAA** | MetaAugment. |
| **PBA** | Population Based Augmentation. |
| **PI** | Probability of Improvement. |
| **RandAA** | RandAugment. |
| **RF** | Random Forest. |
| **RNN** | Recurrent Neural Network. |
| **TPE** | Tree-Structured Parzen Estimator. |
| **TuRBO** | Trust Region Bayesian Optimization. |
| **UCB** | Upper Confidence Bound. |
| **WS-AA** | AutoAugment Via Weight Sharing. |

# Chapter 1

# Introduction

In the process of developing machine learning methods, tasks, such as designing the optimal neural architecture for a machine learning problem, choosing image preprocessing strategies, and fine-tuning hyperparameters, etc., often rely on the past knowledge of human experts. However, the optimal neural architecture or hyperparameter configuration could vary noticeably for different machine learning tasks. Thus, tuning the architecture or hyperparameter configuration for various machine learning tasks based on the past knowledge of human experts is time-consuming and often sub-optimal. Automated Machine Learning (AutoML) [1] aims to automate these time-consuming and iterative tasks using search algorithms. In this thesis, we will investigate two AutoML tasks: automated Data Augmentation (DA) [2] and automated Hyperparameter Optimization (HPO) [3].

## 1.1 Challenges in AutoML Tasks

**Automated Data Augmentation.** Deep learning has achieved state-of-the-art performance on computer vision tasks including image classification benchmarks. With an increasing learning capability of deep neural networks comes along the overfitting issue. Many regularization techniques [4, 5] have been proposed to increase the generalizability of deep neural networks. DA, which applies image transformation functions to render images for increasing the diversity of training data, is one of the widely used regularization techniques for training deep neural networks [6–10]. Before the development of AutoML [1], designing a suitable strategy for augmenting specific image datasets relies on the knowledge of human experts [11–17]. Recently, automated DA, which replaces heuristic augmentation schemes manually tuned by human experts with automated search algorithms, has attracted much attention.

AutoAugment (AA) [2] proposes a reinforcement learning framework to search for optimal augmentation policies. As AA evaluates the performance of augmentation policies by training child networks from scratch, it requires thousands of GPU hours, prohibiting its wide adoption in reality. Many recent efforts for the automated DA problem have focused on searching for effective data augmentation policies at a low cost. Fast AutoAugment (FastAA) [18] utilizes Bayesian Optimization with a density matching heuristic to minimize the difference between the training and augmented validation data. DADA [19] reformulates the search for augmentation policies into an efficient differentiable optimization problem. However, FastAA and DADA do not evaluate the performance of individual policy directly, which may lead to a performance gap between the search and evaluation results of policies, and worse transferability across different datasets. Thus, deriving an efficient reward signal that directly measures the performance of individual augmentation policy is one of the challenges for the automated DA problem.

Previous automated DA works focus on reducing the search cost or increasing the performance by searching for augmentation policies according to specific training stages [20] or training samples [21, 22]. In terms of the transferability of augmentation policies, most previous works demonstrate that augmentation policies can achieve satisfying performance when being transferred across different neural networks on the same dataset. However, very little work has been focused on searching for transferable augmentation policies across different datasets. Additionally, the experimental results of AA [2] and AdvAA [21] show that there is a performance gap when transferring augmentation policies searched on one dataset to another dataset. As augmentation policies with good transferability across different datasets alleviate the need for searching on every specific dataset, searching for stationary augmentation policies that have good transferability across different datasets is another challenge for the automated DA problem.

**Automated Hyperparameter Optimization.** Automated HPO [3] has been developed for AutoML tasks [1], e.g., neural architecture search [23] and hyperparameter tuning [24, 25], to alleviate the time-consuming and iterative processes in model design and tuning. HPO methods, such as Grid Search [26] and Random Search [27], employ a pure exploration strategy that does not exploit rewards of visited configurations and thus does not guarantee to converge to the global optimum. Bayesian Optimization (BO) [28, 29] learns from search history toward finding the global optimum. However, traditional BO approaches need to run many search iterations before finding good configurations. For HPO tasks such as measuring the performance of a hyperparameter configuration on training a deep neural network, evaluating the performance of a configuration often takes a long time. Thus, an efficient HPO algorithm needs to find a globally optimal configuration within a small number of evaluations. In addition, another desirable characteristic for an efficient HPO algorithm is to robustly discover a global optimum for various HPO tasks.

## 1.2 Thesis Scope

In this thesis, we aim to develop efficient search algorithms for solving the automated DA and HPO problems, respectively.

**Automated Data Augmentation.** The efficiency of the reward signal is the key to the efficiency of the search algorithm for the automated DA problem. As AA [2] trains a deep neural network for multiple epochs to obtain one reward signal for the sampled augmentation policy, their proposed reward signal prohibits their search algorithm applied to real-life applications. Besides, FastAA [18] reduces the search cost by approximating the performance of augmentation policies indirectly with the heuristic assumption, which may lead to sub-optimal policies for the evaluation. Thus, in this thesis, we aim to design an efficient reward signal that measures the performance of individual policies based on each batch training step of deep neural networks. Moreover, to address the challenge of searching for effective augmentation policies with good transferability across different datasets, we aim to leverage the multi-armed bandit [30] techniques to train a non-parametric bandit model to learn the explicit ranking of augmentation policies. In addition, we propose an evolutionary pruning algorithm to allocate search resources to potentially optimal augmentation policies, thus further improving the convergence speed of our bandit search algorithm.

**Automated Hyperparameter Optimization.** As traditional BO [28, 29] uses a single probabilistic surrogate model to find hyperparameter configurations maximizing the acquisition function, it needs to explore many configurations before finding good ones. Thus, we propose a hybrid Bayesian Optimization and Evolutionary Strategy (BOES) for efficient HPO in machine learning. Specifically, we propose a hybrid BO and CMA-ES [31, 32] to sample diverse configurations from corresponding distributions which balance the exploration and exploitation better. Moreover, we aim to leverage the prediction power of multiple probabilistic surrogate models through an ensemble model to reduce the estimation variance and is robust for various HPO tasks.

## 1.3 Thesis Outline

The remainder of this thesis is organized as follows. Chapter 2 introduces literature reviews on previous automated DA and HPO methods. Chapter 3 introduces our proposed BDA algorithm for the automated DA problem and demonstrates our experimental results on image classification benchmark datasets under various models.

Chapter 4 introduces our proposed BOES algorithm for the automated HPO problem and our experimental results for the automated HPO problem. Finally, we summarize and discuss future works in Chapter 5.

# Chapter 2

# Related Work

In this chapter, we discuss previous approaches for automated DA and HPO problems, respectively.

## 2.1 Automated Data Augmentation

DA has been an effective method for improving the generalization of deep neural networks on supervised and unsupervised learning tasks [33, 34]. For image classification tasks, image transformation operations such as cropping and flipping [13] are usually applied as a pre-processing step to augment the training data for introducing more diversity. To alleviate the requirements of human efforts and prior knowledge on designing good augmentation policies for each specific task, automated DA, which aims to produce effective augmentation policies automatically using the search algorithm, has been developed recently.

AA [2] proposes a reinforcement learning framework that trains the Recurrent Neural Network (RNN) controller to generate a fixed set of stationary augmentation policies for the target dataset. The reward signal of AA for training the RNN controller is the validation accuracy of the child network trained with the training dataset augmented by the sampled augmentation policy from scratch. As obtaining this reward requires training an image classification child network for multiple epochs, evaluating augmentation policies with this reward signal is extremely time-consuming and inapplicable for real-world applications. Thus, many recent studies have focused on reducing the search cost in automated DA.

## 2.2 Efficient Automated Data Augmentation

### 2.2.1 Methods Generating Stationary Policies

FastAA [18] proposes a density matching heuristic that assumes good augmentation policies will not make the distance between the density of the augmented dataset and the density of the unaugmented dataset large. With this density matching heuristic, the reward signal of FastAA is the performance of a network trained with the unaugmented dataset on the dataset augmented by the sampled policies. FastAA uses this reward signal to train the TPE [35] optimizer to avoid training and evaluating multiple child networks from scratch. DADA [19] relaxes the discrete policy search problem into a differentiable optimization problem and solves it with gradient descent. By encoding individual policies with a joint distribution and optimizing network and augmentation policy parameters jointly, DADA reduces the search cost noticeably. However, FastAA and DADA perform the indirect search and cannot evaluate the performance of individual policies on the target dataset.

Moreover, RandAugment (RandAA) [36] simplifies the search space of AutoAugment into two hyperparameters: the number and the joint magnitude of augmentation operations. RandAA performs the grid search over these hyperparameters. Although RandAA does not require extra search cost on a proxy task, grid search on large datasets like ImageNet still has a large time overhead. Weight-sharing AutoAugment (WS-AA) [37] reduces the search cost of AA by fine-tuning child models from the shared pre-trained weights with a small number of epochs to obtain validation accuracies as rewards. Even though they reduce the search space by decreasing the number of magnitude levels for each augmentation operation from 10 to 3, their reward signal is still time-consuming for high-resolution datasets like ImageNet.

Compared with previous methods generating stationary augmentation policies, BDA reduces the search cost considerably with a computationally efficient reward design, which can be computed in a single batch training step on the validation data. In addition, as FastAA needs to initially train a network without augmentation from scratch to kickstart the density matching process, and RandAA uses grid search to optimize augmentation hyperparameters, they are not efficient on large and high-resolution datasets. On ImageNet, BDA achieves a substantial reduction in search cost as compared to FastAA and AA while still directly measures the effectiveness of individual policies. Note that although DADA is fast, it cannot evaluate the goodness of individual augmentation policies and thus can hardly achieve transferability across datasets.

### 2.2.2 Methods Generating Non-Stationary Policies

Algorithms have also been proposed to search for non-stationary policies, which are augmentation schedules dependent on the training progress or training samples. PBA [20] leverages Population Based Training [38] to convert the augmentation policy search problem into a HPO problem. PBA uses the performance of the network on the validation dataset during the training procedure with sampled augmentation policies and the evolutionary mutation of augmentation hyperparameters to learn a non-stationary augmentation schedule. Moreover, based on the hypothesis that the model can learn more robust features using harder training samples, AdvAA [21] proposes an adversarial framework to produce augmentation policies that generate harder augmented training samples. Specifically, AdvAA trains a target network using images augmented by the augmentation policies produced from the policy network. And the policy network (the RNN controller) is trained to maximize the training loss of the target network, thus producing augmentation policies generating harder training samples. Furthermore, MetaAugment (MetaAA) [22] proposes a sample-aware meta-learning approach to generate good augmentation policies. MetaAA uses the sample re-weighting algorithm [39] to train a policy network taking the image and augmentation policy features as the input and generating the weights of augmented images in the training batch to compute the weighted loss of the target network.

BDA produces stationary augmentation policies. Although a non-stationary policy may yield further performance gain, it is coupled into the network training progress and thus does not easily transfer across different models or datasets. Moreover, compared to stationary augmentation policies that can be easily applied to the image pre-processing procedure before the network training, non-stationary augmentation policies require further changes to training routines and incur additional overhead in practical deployment.

### 2.2.3 Transferable Augmentation Policies

As existing auto-augmentation literature is evolving toward more heavily parameterized policies as in non-stationary or sample-aware methods, another largely ignored yet important issue is to learn generalizable policies and avoid performing repeated search everytime when a new dataset is introduced in a production environment. BDA aims to produce less parameterized policies and generate an explicit ranking of operation pairs together with their magnitudes. By sparse operation selection and explicit policy scoring, BDA enables generalizable policies that have steady performance when transferred across datasets. By finding generalizable policies independent of the model training progress, it is more convenient to employ BDA-discovered stationary policies to improve existing neural network training.

## 2.3 Successive Halving Algorithm

Many previous automated DA approaches [18, 20, 36] convert the automated DA problem into the augmentation HPO problem. Similarly, we convert the automated DA problem into a stochastic multi-armed bandit problem in this thesis. Our proposed BDA algorithm takes the idea of the Successive Halving algorithm [40] from

Hyperband [24] to search for optimal augmentation policies.

The Successive Halving algorithm [40] from Hyperband [24] removes the worst half of hyperparameter configurations from the search space iteratively. By removing worse hyperparameters, more computational resources are allocated exponentially to more promising hyperparameter configurations. In [25] and [41], the authors adopt Successive Halving and improve the configuration selection method of Hyperband with Bayesian Optimization and the Sub-Sampling algorithm, respectively. However, we extend Successive Halving into a novel evolutionary strategy for sparse operation pair selection, where we eliminate worse augmentation operation pairs from the search space based on their UCB values exponentially in every round, while in the meantime increasing the searchable magnitude levels for remaining operation pairs. Thus, the proposed bandit model converges faster as more computational resources are progressively allocated to more promising operation pairs.

## 2.4 Multi-armed Bandits

The multi-armed bandit problem aims to tackle the exploration versus exploitation dilemma. In the stochastic bandit problem, there are $k$ arms where each arm has an independent and identically reward distribution [30]. The agent follows an allocation policy to select an arm to play in each step and receive a reward for the selected arm. The goal of the bandit algorithm is to maximize cumulative rewards.

The UCB1 [30] algorithm is a classical algorithm for the stochastic multi-armed bandit problem. For the data augmentation problem, we formulate each possible augmentation policy as an arm. In each step, the agent chooses the arm $i$ that maximizes the upper confidence bound (UCB):

$$\overline{r}_i + C\sqrt{\frac{2\ln n}{n_i}}, \tag{2.1}$$

where $\overline{r}_i$ is the average reward of the arm $i$, $n_i$ is the number of times that the arm $i$ has been played, and $n$ is the total number of arm plays.

## 2.5 Bayesian Optimization

Bayesian Optimization (BO) [43] a sequential and derivative-free optimization approach for finding the globally optimal solutions for black-box objective functions. The optimization problem solved by BO is defined as follows:

$$\max_{\mathbf{x} \in \mathrm{X}} f(\mathbf{x}), \tag{2.2}$$

where $\mathbf{x}$ is the optimization variables belonging to the search space X, $f(\mathbf{x})$ is the black-box objective function optimized by BO.

## 2.6 Automated Hyperparameter Optimization

BO [28, 29], which learns a probabilistic surrogate model on the objective function, is a widely used global optimization algorithm for HPO problems. Gaussian Process regression [43, 42] assumes a Gaussian Process prior on the objective function and updates its posterior distribution using the historical data. From a set of randomly sampled data points, the Gaussian Process regressor selects the data point that maximizes the acquisition function for evaluation. Besides, TPE [35] fits two Gaussian Mixture Models [44] to learn distributions $g(x)$ and $l(x)$ for data points whose objective function values are higher than and less than a threshold, respectively. TPE samples data points from the learned distribution $l(x)$ and suggests the data point maximizing the expected improvement [45] acquisition function for evaluation.

As traditional BO uses a single probabilistic surrogate model to approximate the objective function, it needs to explore many data points for HPO tasks with a large search space or complex objective function. Recently, many works have focused on further improving the efficiency of vanilla BO methods. BO with Local Search (BOwLS) [46] combines BO with the local search method. BOwLS samples the starting point of the local search from the BO model and uses the local minimum to update the BO model for guiding the next starting point. In addition, Blend Search [47] combines the global search and local search strategies to search in a large search space at a low search cost. Blend Search maintains a pool of search threads consisting of one global search thread and multiple local search threads. Using the estimated future reward as the priority metric, Blend Search selects one thread to sample configuration for evaluation while balancing the exploitation and exploration. Moreover, Trust Region BO (TuRBO) [48] proposes a local strategy for BO to solve HPO with high-dimensional search spaces. TuRBO maintains a collection of local BO models and uses an implicit bandit approach to allocate samples to promising local BO models. Furthermore, prior work finds that optimizing a single acquisition function for different HPO tasks may lead to sub-optimal results [49, 50]. To tackle this problem, HEBO [50] utilizes the multi-objective evolutionary algorithm [51] to optimize multiple acquisition functions simultaneously for sampling better configurations across various HPO tasks.

Compared to BO, Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [31, 32] is more effective for optimizing HPO tasks whose objective functions are ill-conditioned. CMA-ES uses the evolutionary strategy to sample configurations from a multivariate normal distribution of the search space in a stochastic way. The mean and the covariance matrices of the distribution are updated with the maximum-likelihood principle so that sampled configurations converge to the global optimum.

As local search is easier to be stuck at the local optimum, to handle the exploitation-vs-exploration dilemma, BOES proposes a hybrid sampler combining TPE and CMA-ES for sampling more diverse configuration candidates from corresponding learned distributions, respectively. Moreover, instead of approximating the objective function with a single probabilistic surrogate model as traditional BO, BOES proposes an ensemble surrogate model which enhances the robustness and capability of BOES on approximating objective functions across various HPO tasks. Furthermore, BOES optimizes multiple acquisition functions with single-objective optimizers. Compared to HEBO that uses the multi-objective optimizer, our approach achieves better performance when exploring the same number of data points.

# Chapter 3

# The BDA Algorithm for Automated Data Augmentation

Automated DA aims to search for effective augmentation policies to introduce more diversity to the training data for improving the generalizability of deep neural networks. As mentioned in Chapter 1, for the automated DA problem, it is significant to design an efficient reward signal to evaluate the performance of individual augmentation policies. In addition, designing an approach that generates transferable augmentation policies across different datasets is essential as good transferability alleviates the need for searching on every specific dataset. In this thesis, we propose a novel bandit algorithm to search for transferable and generalizable augmentation policies at a low cost. We train a non-parametric bandit model to learn an explicit ranking of augmentation policies, with a bias toward sparse selection of operation combinations and a computationally efficient rewarding scheme to reduce the search cost. Our main contributions can be summarized as follows:

First, we propose a bandit algorithm based on the upper confidence bound (UCB) and the lower confidence bound (LCB) of augmentation policies with a novel evolutionary pruning algorithm that achieves sparsity in selecting operation combinations. In the pruning algorithm, considering the correlation among policies with the same operation pair, policies are aggregated by their operation pair. We reduce the number of active operation pairs in the search space according to their UCB values. In the meantime, we progressively increase the number of searchable magnitude levels for active operation pairs. While prior methods tend to be heavily parameterized and are not designed for discovering transferable augmentation policies across datasets, BDA enables exploring more important augmentation operations with more computational resources, thus leading to generalizable policies.

Second, to reduce the search cost, we design an efficient rewarding scheme to measure the performance of individual augmentation policies based on each single step of training. For each batch training step, our reward signal is the net prediction gain that measures the difference in validation losses between the neural network trained on the augmented batch and the duplicated network trained on the original batch. Compared to the reward signal in AA [2] which requires training child networks from scratch for multiple epochs, and the density matching reward in FastAA [18] that requires the initial training a network from scratch without augmentation, our reward signal reduces the search cost dramatically, especially for large and high-resolution datasets such as ImageNet.

Third, we conduct extensive experiments on image classification benchmark tasks across different models. Experimental results demonstrate that our approach achieves comparable or better performance than previous methods [2, 18, 20, 36] on CIFAR-10/100 [52], SVHN [53] and ImageNet [54] under various models. On ResNet-50, BDA achieves the best ImageNet accuracy compared to prior stationary augmentation policies found in the same search space, with a search cost 536 times smaller than AA and 16 times smaller than FastAA. Moreover, we show that the augmentation policies found by our approach demonstrate superior transferability and generalizability across datasets, achieving comparable performance to direct search on target datasets.

## 3.1 Search Space

Following the design in AA [2], there are 15 augmentation operations in the search space, e.g., Shear X/Y, Translate X/Y, Rotate, etc. Similar to AA, we adopt an

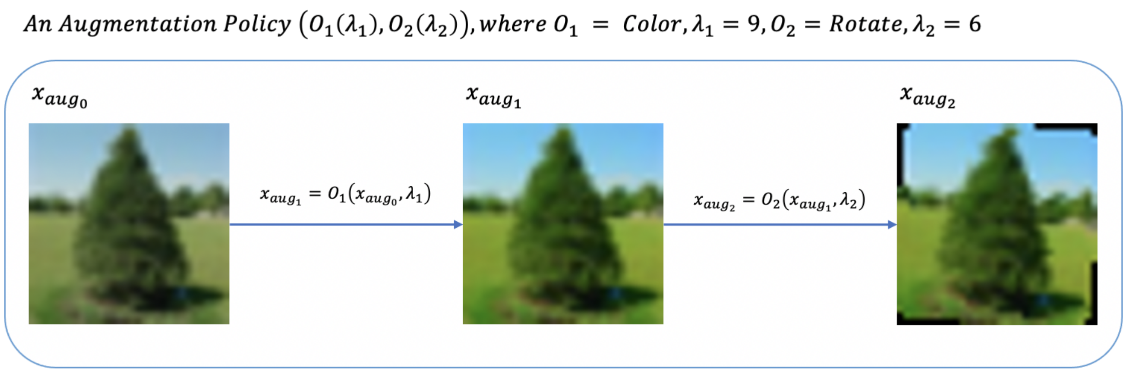

An Augmentation Policy $(O_1(\lambda_1), O_2(\lambda_2))$, where $O_1$ = Color, $\lambda_1 = 9$, $O_2$ = Rotate, $\lambda_2 = 6$
$x_{aug_0}$
$x_{aug_1}$
$x_{aug_2}$
$x_{aug_1} = O_1(x_{aug_0}, \lambda_1)$
$x_{aug_2} = O_2(x_{aug_1}, \lambda_2)$

Figure 3.1: The example of applying the given augmentation policy to an image and generating the resulting augmented image.

augmentation policy consisting of two sub-policies. Each sub-policy is defined as $O(\lambda)$, where $O$ is one of 15 augmentation operations, and $\lambda$ is the magnitude level of the operation $O$. There is a total of 10 levels for the magnitude, range from 0 to 9. Thus, the augmentation policy is defined as $(O_1(\lambda_1), O_2(\lambda_2))$, where $O_1$ and $O_2$ can be one of 15 augmentation operations from the search space, $\lambda_1$ and $\lambda_2$ can be one of magnitude levels between 0 and 9. Similarly, the augmentation operation pair is defined as $(O_1(\cdot), O_2(\cdot))$.

Given an image $x$ and an augmentation policy $(O_1(\lambda_1), O_2(\lambda_2))$, the augmented image $x_{aug}$ is defined as:

$$x_{\text{aug}} = O_2(O_1(x, \lambda_1), \lambda_2). \tag{3.1}$$

We illustrate the procedure of applying an augmentation policy to an image $x$ in Figure 3.1.

## 3.2 Bandit-formulation of Data Augmentation

The multi-armed bandit problem aims to tackle the exploration versus exploitation dilemma. The agent follows an allocation policy to select an arm to play and receives a reward in each step. The goal of the bandit algorithm is to maximize cumulative rewards [30].

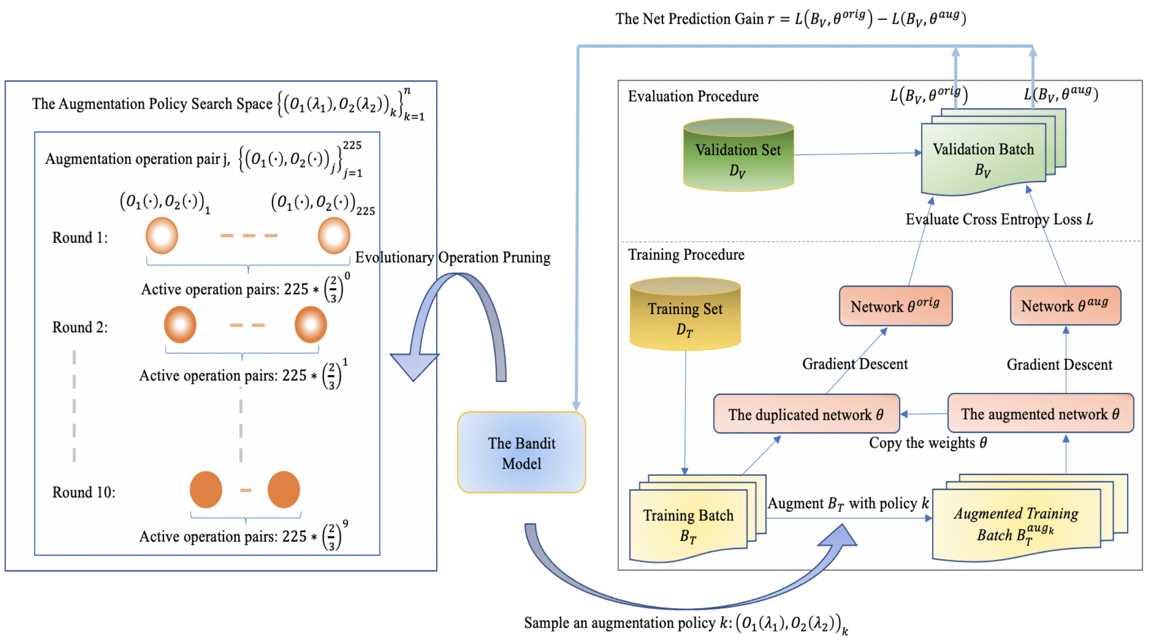

The Net Prediction Gain $r = L(B_V, \theta^{orig}) - L(B_V, \theta^{aug})$
The Augmentation Policy Search Space $\{(O_1(\lambda_1), O_2(\lambda_2))_k\}_{k=1}^{n}$
Augmentation operation pair j, $\{(O_1(\cdot), O_2(\cdot))_j\}_{j=1}^{225}$
$(O_1(\cdot), O_2(\cdot))_1$
$(O_1(\cdot), O_2(\cdot))_{225}$
Round 1:
Active operation pairs: $225 * \left(\frac{2}{3}\right)^0$
Round 2:
Active operation pairs: $225 * \left(\frac{2}{3}\right)^1$
Round 10:
Active operation pairs: $225 * \left(\frac{2}{3}\right)^9$
Evolutionary Operation Pruning
The Bandit Model
Evaluation Procedure
Validation Set $D_V$
Validation Batch $B_V$
$L(B_V, \theta^{orig})$
$L(B_V, \theta^{aug})$
Evaluate Cross Entropy Loss $L$
Training Procedure
Training Set $D_T$
Network $\theta^{orig}$
Network $\theta^{aug}$
Gradient Descent
Gradient Descent
The duplicated network $\theta$
The augmented network $\theta$
Copy the weights $\theta$
Training Batch $B_T$
Augment $B_T$ with policy $k$
Augmented Training Batch $B_T^{aug_k}$
Sample an augmentation policy $k$: $(O_1(\lambda_1), O_2(\lambda_2))_k$

Figure 3.2: Our proposed Bandit framework. For each batch training step, the bandit model samples an active augmentation policy $k$. The reward is the loss difference between the network trained with the original batch and the network trained with the augmented batch on a validation batch. We perform Evolutionary Operation Pruning to eliminate worse operation pairs and increase searchable magnitude levels of promising ones. The whole training and pruning procedures are repeated for 10 rounds.

In the case of automated DA, we propose a non-parametric bandit model to search for optimal augmentation policies that maximize the cumulative net prediction gain of augmentation policies on the validation data. Specifically, we treat each augmentation policy as an arm in our bandit model. Thus, there are $(15 \times 10)^2$ arms in the search space for the bandit model.

An overview of our proposed framework is provided in Figure 3.2. We utilize a bandit model to perform policy search during the training of the deep neural network. For each update of the bandit model, it samples a policy to augment a training batch, and receives the reward signal derived by updating the network separately with the augmented batch and the original batch, and calculating the net prediction gain as the validation loss difference of the two updated networks. We also adopt a 10-round Evolutionary Operation Pruning procedure where for each round, one-third of

**Algorithm 1:** The Bandit Data Augment Algorithm

```
   Input: Training set $D_T$, validation set $D_V$ and the search space of
          augmentation policies $\{(O_1(\lambda_1), O_2(\lambda_2))_k\}_{k=1}^{n}$
   Output: The bandit model $BM$
 1 Initialize the network $\theta$
   /* Initialize the bandit model $BM$                                  */
 2 for every policy $p \in 1, \ldots, n$ do
 3     Sample a training batch $B_T$ from $D_T$ and a validation batch $B_V$ from $D_V$
 4     Compute_Reward_Update($B_T$, $B_V$, $\theta$, $k$, $BM$)
   /* The search procedure t is repeated for T search rounds          */
 5 for $t = 1, \ldots, T$ do
 6     Initialize the network $\theta$
 7     for $e = 1, \ldots, max\_epoch$ do
 8         for every training batch $B_T \in D_T$ do
 9             Sample a policy $k$ using Eq. (3.3)
10             Sample a validation batch $B_V$ from $D_V$
               /* Compute and update the reward for the sampled policy
                  $k$                                                  */
11             Compute_Reward_Update($B_T$, $B_V$, $\theta$, $k$, $BM$)
       /* Deactivate worse augmentation operation pairs and evolve
          magnitude levels of active ones                              */
12     Evolutionary_Operation_Pruning($t$, $BM$)
```

operation pairs are deactivated and searchable magnitude levels of active operation pairs are expanded. Finally, the bandit model filters out incompetent operations and derives effective policies.

## 3.3 The BDA Algorithm

Our BDA algorithm is described in Algorithm 1 as follows:

Given a training set $D_T$, a validation set $D_V$, and the search space of augmentation policies, we first initialize each augmentation policy in the bandit model with the net prediction gain computed using sampled training and validation batches in lines 1-4. Besides, the search procedure $t$ is repeated for $T$ search rounds in lines 5-12. Specifically, we first initialize a network $\theta$ at the beginning of each round. Then, for every batch training step, the bandit model samples an augmentation policy $k$ to train this network with the augmented batch. In the meantime, we train a duplicated network with the original batch and use the net prediction gain as the reward to update this sampled policy in the bandit model. At the end of each round of the search, in line 12, we use Evolutionary Operation Pruning algorithm to deactivate worse operation pairs and allocate more computational resources to potential optimal augmentation policies. Moreover, different from UCB1 algorithm that selects the arm with the maximum UCB, to increase the diversity of augmented data and encourage the exploration during the search procedure, in line 9, the bandit model samples an active policy $k$ with the probability of negative LCB $Prob_{\text{lcb}_k}$ [55],

$$V_{\text{lcb}_k} = \frac{r_{\text{total}_k}}{n_k} - C\sqrt{\frac{2\ln n}{n_k}}, \tag{3.2}$$

$$Prob_{\text{lcb}_k} = \frac{\exp\left(-V_{\text{lcb}_k}\right)}{\sum_{j=1}^{n}\exp\left(-V_{\text{lcb}_j}\right)}, \tag{3.3}$$

where $V_{\text{lcb}_k}$ is the LCB value of policy $k$, $r_{\text{total}_k}$ is the total reward of policy $k$, $n$ is the overall number of plays done so far, $n_k$ is the number of times that policy $k$ has been played.

### 3.3.1 Reward Signal

BDA aims to find a set of augmentation policies that maximize the cumulative net prediction gains of augmented batches. In previous reinforcement learning approaches, AA [2] obtains one reward signal by training a child image classification network from scratch with training data augmented by the sampled augmentation policy. This design of the reward signal makes obtaining enough rewards for training the policy network (the RNN controller) very time-consuming. To reduce the computational cost on obtaining the reward signal, inspired by the curriculum learning literature [56], the authors train adversarial bandit algorithm Exp3.S with the prediction gain, which is the loss change of the network on a sample $x$, before and after training on $x$.

**Algorithm 2:** Compute_Reward_Update

```
Input: Training batch $B_T$, validation batch $B_V$, network $\theta$, policy $k$, bandit
       model $BM$
1  Copy weights of network $\theta$ to a duplicated network.
2  Train this duplicated network with the original batch $B_T$ to obtain the
     network $\theta^{\text{orig}}$
3  Compute the cross-entropy loss of $\theta^{\text{orig}}$ on the validation batch $L(B_V, \theta^{\text{orig}})$
4  Obtain the augmented batch $B_T^{\text{aug}_k}$ by applying the policy $k$ to every image
     in $B_T$ using Eq. (3.1)
5  Train the network $\theta$ with the augmented batch $B_T^{\text{aug}_k}$ to obtain the network
     $\theta^{\text{aug}}$
6  Compute the cross-entropy loss of $\theta^{\text{aug}}$ on the validation batch $L(B_V, \theta^{\text{aug}})$
7  Compute the net prediction gain $r_k$ for the policy $k$ using Eq. (3.4)
8  Map $r_k$ to $r_{k_{\text{scaled}}} \in [0, 1]$ using Eq. (3.5)
9  Update the bandit model $BM$:
10     $n = n + 1$            /* The overall number of plays $n$             */
11     $n_k = n_k + 1$    /* The number of times policy $k$ has been played */
12     $r_{\text{total}_k} = r_{\text{total}_k} + r_{k_{\text{scaled}}}$    /* The total reward of the policy $k$    */
```

Since the prediction gain of each augmentation policy will decrease during the network training, it is not suitable to use the prediction gain as the reward signal in the stochastic multi-armed bandit problem where each policy's rewards are drawn from a stationary distribution. Thus, we design our reward signal as the net prediction gain of the augmented network on the validation data compared to the duplicated network trained with the original data.

Given a training batch $B_T$ and a validation batch $B_V$, the reward signal, the net prediction gain, is defined as

$$r = L(B_V, \theta^{\text{orig}}) - L(B_V, \theta^{\text{aug}}), \tag{3.4}$$

where $\theta^{\text{orig}}$ is the weights of the duplicated network trained with the original batch $B_T$, $\theta^{\text{aug}}$ is the weights of the network $\theta$ trained with the augmented batch $B_T^{\text{aug}}$, $L$ is the cross-entropy loss of network on the validation batch $B_V$.

The right part of Figure 3.2 illustrates how the net prediction gain (the reward signal) is computed in a single batch training step. Our algorithm for computing the net prediction gain can be summarized as follows:

In line 1 of Algorithm 2, to avoid the risk that the network trained without augmentation could diverge from the network trained with augmented data due to the randomness of stochastic gradients, we copy the weights of the augmented network $\theta$ to a duplicated network before each batch training step. Then, in lines 2-6 of Algorithm 2, we train the duplicated network and the augmented network $\theta$ with the original batch $B_T$ and the augmented batch $B_T^{\text{aug}_k}$, compute the net prediction gain using the cross-entropy loss difference of the network trained on the original batch $\theta^{\text{orig}}$ and the network trained on the augmented batch $\theta^{\text{aug}}$ on the validation data. Moreover, as the net prediction gain does not belong to the $[0, 1]$ interval, to avoid fine-tuning the exploration coefficients of UCB and LCB, in line 8 of Algorithm 2, we use the reservoir sampling algorithm to map the net prediction gain to the $[0, 1]$ interval with Eq. (3.5). And we update the bandit model for the policy $k$ in lines 9-12 of Algorithm 2.

Since the net prediction gain measures how much the validation loss that the network trained with data augmented by the sampled policy can reduce compared to the network trained with the original batch in a single batch training step, this reward signal reflects the effectiveness of the sampled augmentation policy on the generalization of the network. By obtaining each augmentation policy's long-term cumulative net prediction gain over the search, we can determine the relative goodness of augmentation policies as stationary policies. Thus, this per-batch training reward signal is a merit of our BDA algorithm, as it avoids training child networks for multiple epochs to obtain one reward, which escalates evaluation costs.

Furthermore, as the exploration coefficients of UCB and LCB are affected by the magnitude of the net prediction gain, to alleviate the effort of fine-tuning these hyperparameters, following the reward scaling method used in [56], we implement the reservoir sampling algorithm to scale the net prediction gain $r$ to the interval $[0, 1]$. Let $\{r_i\}_{i=1}^{t-1}$ be an array that stores the history of net prediction gains, $r^{20^{\text{th}}}$ and $r^{80^{\text{th}}}$ be the 20th and 80th percentiles of this array. At time step $t$, the net prediction gain

$r_t$ is scaled as

$$r_{t_{\text{scaled}}} = \begin{cases} 0.0, & \text{if } r_t < r^{20^{\text{th}}}, \\ 1.0, & \text{if } r_t > r^{80^{\text{th}}}, \\ \frac{r_t - r^{20^{\text{th}}}}{r^{80^{\text{th}}} - r^{20^{\text{th}}}}, & \text{otherwise.} \end{cases} \tag{3.5}$$

### 3.3.2 Evolutionary Operation Pruning

We propose the Evolutionary Operation Pruning algorithm to allow the bandit model to allocate more computational resources to potentially optimal operation pairs, thus reducing the search cost. From the idea of the Successive Halving algorithm [40] used in Hyperband [24], to allocate more resources to more promising configurations and stop worse performing configurations early, the authors abandon the worst half configurations at each round of the search. Different from the HPO problem where there are no apparent correlations between different hyperparameters, we design a hierarchical format for augmentation policies with the correlations among augmentation policies with the same operation pair yet different magnitude levels taken into consideration.

In the Evolutionary Operation Pruning algorithm, instead of deactivating worse augmentation policies iteratively, we aggregate augmentation policies by their operation pair and perform the operation pair pruning to deactivate worse performing augmentation operation pairs from the search space gradually. In addition, as it is beneficial to spend less exploration on the less significant regions of the search space, we only increase searchable magnitude levels of potentially optimal operation pairs after each round of pruning. As worse operation pairs are deactivated in the next round of the search, this algorithm allows us to achieve sparsity in augmentation operation pairs. Our proposed Evolutionary Operation Pruning algorithm can be summarized as follows:

As shown in lines 1-2 of Algorithm 3, we first compute the UCB value for every

**Algorithm 3:** Evolutionary_Operation_Pruning

```
  Input: The search round $t$, the bandit model $BM$
1 Deactivate all augmentation policies
2 Compute the UCB value for every augmentation operation pair
    $(O_1(\cdot), O_2(\cdot))_{j=1}^{225}$ using Eq. (3.6)
  /* The number of active operation pairs at the search round $t$ */
3 $n_{\text{active\_op\_pairs}} = 225 \times (\frac{2}{3})^t$
4 Sort operation pairs according to their UCB values and activate top
    $n_{\text{active\_op\_pairs}}$ operation pairs
5 for every operation pair $j \in$ active operation pairs do
6     if $t \geq 2$ then
7         Increase searchable magnitude levels of active augmentation operation
            pair $j$
8     Activate augmentation policies with the operation pair $j$ and its
        searchable magnitude levels
```

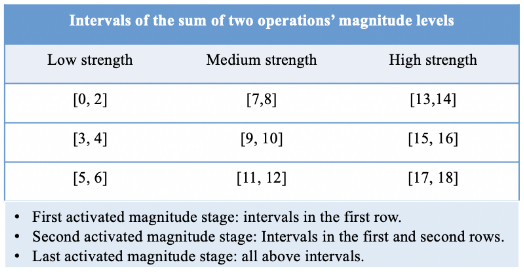
augmentation operation pair:

$$V_{\text{ucb}_j} = \frac{\sum_{k \in j} r_{\text{total}_k}}{\sum_{k \in j} n_k} + C\sqrt{\frac{2 \ln n}{\sum_{k \in j} n_k}}, \tag{3.6}$$

where $V_{\text{ucb}_j}$ is the UCB value of the operation pair $j$, $k \in j$ represents the policy $k$ with the operation pair $j$.

Next, in lines 3-4 of Algorithm 3, we reduce the number of active operation pairs in the previous search round by one-third. Then, we active potentially optimal operation pairs according to their UCB values for the next round of the search. Moreover, in lines 5-8, after the second round of the search procedure, we increase the searchable magnitude levels of every active operation pair. That is, augmentation policies whose operation pair is active and the sum of magnitude levels are searchable are active policies in the next round of the search.

Inspired by the evolutionary strategy, it is beneficial to allocate more search resources to potentially optimal search regions. Thus, we activate a subset of magnitude levels for every operation pair at the early stage of the search procedure. After several rounds of search, we gradually increase the number of searchable magnitude levels of augmentation operation pairs with better performance, which allows our bandit



| Intervals of the sum of two operations' magnitude levels | | |
|---|---|---|
| Low strength | Medium strength | High strength |
| [0, 2] | [7,8] | [13,14] |
| [3, 4] | [9, 10] | [15, 16] |
| [5, 6] | [11, 12] | [17, 18] |

- First activated magnitude stage: intervals in the first row.
- Second activated magnitude stage: Intervals in the first and second rows.
- Last activated magnitude stage: all above intervals.

Figure 3.3: There are three activated magnitude stages for each augmentation operation pair. For each augmentation operation pair, every activated magnitude stage contains augmentation policies of this operation pair with the sum of two augmentation operations' magnitude levels belonging to the listed low strength, medium strength, and high strength intervals.

model to spend more search resources on more promising operation pairs and achieve faster convergence. Specifically, we divide augmentation policies with the same operation pair into three magnitude stages. The intuition of setting searchable values of magnitude levels in each magnitude stage is that we divide all possible values of the operation pair's magnitude levels sum into low, medium, and high strengths. Every magnitude stage contains an interval of the operation pair's magnitude levels sum from each strength. After the second round of the search procedure, we increase the magnitude stage of each active operation pair by one and activate its augmentation policies in its current magnitude stage. As illustrated in Figure 3.3, at the first stage, active operation pairs whose sum of magnitude levels belong to the set $\{0, 1, 2, 7, 8, 13, 14\}$ are searchable. At the next stage, active operation pairs whose sum of magnitude levels belong to the set $\{3, 4, 9, 10, 15, 16\}$ are also searchable. At the last stage, all magnitude levels of active operation pairs are searchable.

By reducing the number of active augmentation operation pairs iteratively, we achieve the sparsity in operation pairs and allocate more computational resources to potentially optimal operation pairs. Furthermore, instead of allowing the bandit model to select all magnitude levels of active augmentation operation pairs, we limit the searchable magnitude levels at the early stage of the search procedure. During the search, only active operation pairs can have more searchable magnitude levels.

Therefore, the bandit model wastes less computational resources on less promising operation pairs and converges faster.

## 3.4 Experiments and Results

In this section, we evaluate the performance of BDA on CIFAR-10/100 [52], SVHN [53] and ImageNet [54] datasets across different models including Wide-ResNet [57], Shake-Shake [58], and PyramidNet [59]. For a fair comparison between algorithms, we compare our results with stationary policies produced by methods in the same search space that most other papers used, including AA [2], FastAA [18], RandAA [36], and DADA [19]. Since the search space of WS-AA [37] is different, containing only three magnitude levels, we do not compare our approach with them. We also include results of non-stationary policies found by PBA [20], AdvAA [21], and MetaAA [22]. Furthermore, we conduct experiments to demonstrate the transferability of our policies across datasets. Additionally, we perform the ablation study to demonstrate the effectiveness of our proposed Evolutionary Operation Pruning algorithm.

### 3.4.1 Implementation Details

**CIFAR-10/100 Search** For CIFAR-10 and CIFAR-100 datasets, we adopt two data split settings in the search procedure: (1) We split the training dataset of CIFAR-10/100 into a training set $D_T$ with 40k images and a validation set $D_V$ with 10k images. (2) We include 4k images in the training set $D_T$ and 46k images in the validation set $D_V$. The batch sizes of the training batch $B_T$ and the validation batch $B_V$ are set to 32 and 128, respectively. We set *max_epoch* to 200 and $T$ to 10. Additionally, we use an SGD optimizer with a learning rate of 0.1, a weight decay of 0.0002, and a cosine learning rate scheduler to optimize the Wide-ResNet 40-2 network. Since we scale the net prediction gain to the $[0, 1]$ interval, the exploration coefficients $C$ for UCB and LCB in Eqs. (3.2) and Eqs. (3.6) are set to 1.

**SVHN Search** We perform the search procedure on the reduced SVHN dataset. We randomly select 4k samples as the training set $D_T$ and include the rest of samples from the original training dataset in the validation set $D_V$. We set the batch sizes $B_T$ and $B_V$ to 32 and 128, respectively. We set *max_epoch* to 200 and $T$ to 10. The exploration coefficients $C$ for UCB and LCB are set to 1. In addition, same as the setting in our CIFAR-10/100 search, we use an SGD optimizer with the same hyperparameters to train the Wide-ResNet 40-2 network.

**ImageNet Search** Following the setting in Fast AutoAugment, we perform the search procedure on the reduced ImageNet dataset. We randomly select 6k samples as the training set $D_T$ from 120 classes, include the rest of samples in the same 120 classes in the validation set $D_V$. The batch sizes $B_T$ and $B_V$ are 32 and 128, respectively. We set *max_epoch* to 90 and $T$ to 10. The exploration coefficients $C$ for UCB and LCB are set to 1. We use the same hyper-parameters for training ResNet-50 as Fast AutoAugment.

**Evaluation of Augmentation Policies** For the segmentation of augmentation policies for evaluation, we choose the union of policies with UCB values or average reward values approximately greater than their 80th percentiles. That is, we select $1,500$ policies with top UCB values and $1,500$ policies with top average reward values, and use the combined policies for evaluating augmentation policies searched on the CIFAR-10/100, SVHN and ImageNet datasets. We follow Fast AutoAugment [18] and adopt the same evaluation settings.

### 3.4.2 Results and Analysis

**CIFAR-10** Table 3.1 summarizes our results on CIFAR-10 under different models. Among the listed approaches that yield stationary policies, BDA achieves the best performance on Wide-ResNet 40-2, Wide-ResNet 28-10, Shake-Shake-96d, and PyramidNet+ShakeDrop. Besides, on Shake-Shake-32d and Shake-Shake-112d, BDA

Table 3.1: CIFAR-10 top-1 test accuracy (%), search cost (GPU hours). For AA, FastAA, DADA, PBA, MetaAA, and BDA, the augmentation policies are searched on the reduced CIFAR-10 dataset with Wide-ResNet-40-2. These policies are transferred to other models for evaluation. For RandAA and AdvAA, the search procedure is performed on each model separately on CIFAR-10, which does not show the transferability of augmentation policies across different models.

| | | Stationary | | | | | | Non-stationary | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Model | No Augment | AA | FastAA | RandAA | DADA | BDA-4k[1] | BDA-40k[2] | PBA | AdvAA | MetaAA |
| Wide-ResNet-40-2 | 94.7 | 96.3 | 96.3 | – | 96.4 | $96.46 \pm 0.069$ | $\mathbf{96.53 \pm 0.074}$ | – | – | 96.79 |
| Wide-ResNet-28-10 | 96.1 | 97.4 | 97.3 | 97.3 | 97.3 | $97.49 \pm 0.115$ | $\mathbf{97.58 \pm 0.092}$ | 97.42 | 98.10 | 97.76 |
| Shake-Shake-32d | 96.4 | **97.5** | **97.5** | – | 97.3 | $97.39 \pm 0.075$ | $97.39 \pm 0.085$ | 97.5 | 97.64 | – |
| Shake-Shake-96d | 97.1 | 98.0 | 98.0 | 98.0 | 98.0 | $\mathbf{98.05 \pm 0.026}$ | $98.04 \pm 0.056$ | 97.97 | 98.15 | 98.29 |
| Shake-Shake-112d | 97.2 | **98.1** | **98.1** | – | 98.0 | $98.03 \pm 0.016$ | $98.03 \pm 0.045$ | 97.97 | 98.22 | 98.28 |
| PyramidNet+ShakeDrop | 97.3 | 98.5 | 98.3 | 98.5 | 98.3 | $98.53 \pm 0.024$ | $\mathbf{98.54 \pm 0.048}$ | 98.54 | 98.64 | 98.57 |
| Search Cost (hours) | – | 5000 | 3.5 | – | 0.1 | 9 | 103 | 5 | – | 18 |

1 BDA-4k represents evaluation results using augmentation policies searched with 4k training images.
2 BDA-40k represents evaluation results using augmentation policies searched with 40k training images.

performs better than DADA and achieves comparable performance compared with AA and FastAA.

When compared with non-stationary policy schedules, BDA performs better than PBA on Wide-ResNet 28-10, Shake-Shake-96d, and Shake-Shake-112d. Besides, on Shake-Shake-32d and PyramidNet+ShakeDrop, BDA achieves comparable performance compared with PBA. For other listed non-stationary policy methods, MetaAA and AdvAA achieve better performance across listed models. However, MetaAA trains the network for 600 epochs until convergence, while other stationary methods train the network for only 200 epochs for evaluation. For AdvAA, it requires to perform policy search every time it is applied to a different model, causing larger overhead in real-life application scenarios. Additionally, unlike the evaluation procedure of our method, AA, FastAA, and DADA, which sample one policy to transform each image, AdvAA uses the batch augment trick [60] to augment each image into 8 transformed images. Thus, it is not a fair comparison between AdvAA and other methods, including AA, FastAA, DADA, and BDA. Furthermore, non-stationary policy sched-

Table 3.2: CIFAR-100 top-1 test accuracy (%), search cost (GPU hours). For FastAA, DADA, MetaAA and BDA, augmentation policies are searched on the reduced CIFAR-100 dataset with Wide-ResNet-40-2. These policies are transferred across different models for evaluation. Moreover, RandAA and AdvAA perform the search procedure on each model separately on CIFAR-100.

| | | Stationary | | | | | | Non-stationary | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Model | No Augment | AA | FastAA | RandAA | DADA | BDA-4k | BDA-40k | PBA | AdvAA | MetaAA |
| Wide-ResNet-40-2 | 74.0 | 79.3 | 79.4 | – | 79.1 | $79.45 \pm 0.115$ | $\mathbf{79.61 \pm 0.261}$ | – | – | 80.60 |
| Wide-ResNet-28-10 | 81.2 | 82.9 | 82.8 | 83.3 | 82.5 | $\mathbf{83.48 \pm 0.222}$ | $83.44 \pm 0.253$ | 83.27 | 84.51 | 83.79 |
| Shake-Shake-96d | 82.9 | **85.7** | 85.4 | – | 84.7 | $85.01 \pm 0.287$ | $84.95 \pm 0.091$ | 84.69 | 85.90 | 85.97 |
| Search Cost (hours) | – | – | 3.5 | – | 0.2 | 11 | 89 | 5 | – | 18 |

ules are searched based on the progress of the model training procedure. Even though it is natural to assume that some policies may be good at different training stages of the neural network. However, compared with stationary policies, which learn the relative overall effectiveness of augmentation policies on the distribution of dataset, the transferability of non-stationary schedules across different models and datasets will be affected if the number of training epochs is noticeably different.

**CIFAR-100** Table 3.2 summarizes our results on CIFAR-100 under different models. On Wide-ResNet 40-2 and Wide-ResNet 28-10, BDA achieves the best performance compared with the listed methods generating stationary policies. On Shake-Shake-96d, our method performs better than DADA. When compared with methods generating non-stationary policies, BDA performs better than PBA on Wide-ResNet 28-10 and Shake-Shake-96d.

**SVHN** Table 3.3 summarizes our results on SVHN on Wide-ResNet 28-10. As shown in the table, BDA achieves better perfomance than DADA and PBA. In addition, BDA achieves comparable performance compared with AA and FastAA.

**ImageNet** Table 3.4 summarizes our results on ImageNet under ResNet-50 and ResNet-200. On the typical benchmark model ResNet-50, BDA achieves noticeably higher accuracy than other algorithms under the stationary policy category, including

Table 3.3: SVHN top-1 test accuracy (%), search cost (GPU hours). For AA, FastAA, DADA, PBA and BDA, augmentation policies are searched on the reduced SVHN dataset.

| | | Stationary | | | | Non-stationary |
|---|---|---|---|---|---|---|
| Model | No Augment | AA | FastAA | DADA | BDA-4k | PBA |
| Wide-ResNet-28-10 | 98.5 | **98.9** | **98.9** | 98.8 | $98.85 \pm 0.0001$ | $98.82 \pm 0.022$ |
| Search Cost (hours) | – | 1000 | 1.5 | 0.1 | 11 | – |

Table 3.4: ImageNet top-1/top-5 test accuracy (%), search cost (GPU hours). For AA, FastAA, DADA, and BDA, the policies are obtained from the search on reduced ImageNet with 6k samples from 120 classes using ResNet-50. The policies are transferred to ResNet-200 for evaluation. For RandAA and AdvAA, the search is performed on each model separately on ImageNet.

| | | Stationary | | | | | Non-stationary | |
|---|---|---|---|---|---|---|---|---|
| Model | Baseline | AA | FastAA | RandAA | DADA | BDA | AdvAA | MetaAA |
| ResNet-50 | 76.3/93.1 | 77.6/93.8 | 77.6/93.7 | 77.6/93.8 | 77.5/93.5 | **$78.115 \pm 0.009$ / $93.868 \pm 0.007$** | 79.40/94.47 | 79.74/94.64 |
| ResNet-200 | 78.5/94.2 | 80.0/95.0 | **80.6 / 95.3** | – | – | $80.144 \pm 0.008/95.094 \pm 0.094$ | 81.32/95.30 | 81.43/95.52 |
| Search Cost (h) | – | 15000 | 450 | – | 1.3 | 28 | 1280 | 480 |

AA, FastAA, RandAA, and DADA, at a low search cost. On ResNet-200, our approach achieves a better result than AA. Although BDA performs slightly worse than FastAA on ResNet-200, our approach only takes 28 GPU hours and is 16 times faster than FastAA. Moreover, as a non-stationary augmentation method, AdvAA requires searching on the entire training dataset, resulting in a high search cost, which is 46 times slower than BDA. Additionally, MetaAA takes 3x training time than a standard training of a task network, which is 17 times slower than BDA.

**Search Cost** The last rows from Table 3.1, Table 3.2, Table 3.3, and Table 3.4 compare the search costs of the augmentation methods on the corresponding datasets. For evaluations on CIFAR-10 and CIFAR-100, augmentation searching methods that yield stationary policies, e.g., AA, FastAA, and DADA, perform the search procedure on the reduced CIFAR-10 dataset with 4k training images. Under this search setting, BDA takes 9 GPU hours to search on CIFAR-10, which is 555 times faster than

AA. Similarly, for the search on CIFAR-100, BDA takes 11 GPU hours. Although the search cost is higher than FastAA and DADA, BDA achieves better performance on Wide-ResNet-40-2/28-10 within reasonable time. In addition, for the search on SVHN, BDA takes 11 GPU hours, which is 91 times faster than AA.

Furthermore, for high-resolution image datasets like ImageNet, our approach takes only 28 GPU hours in the search procedure, which is 536 times faster than AA and is 16 times faster than FastAA. For FastAA, the requirement of training a task network without augmentation from scratch reduces it's efficiency for ImageNet. Additionally, although RandAA does not require the extra computational cost on the search procedure, it uses grid search to optimize the hyperparameters for data augmentation, whose cost could be extremely high for large and high-resolution image datasets like ImageNet, which takes several days to train. Comparably, our approach is efficient for large and high-resolution image datasets like ImageNet.

### 3.4.3 Transferability and Interpretability

**Transferability Across Different Datasets** Table 3.5 demonstrates the transferability of augmentation policies searched by BDA on a low-resolution image dataset to another image dataset with the same resolution and a high-resolution image dataset. As AA and PBA demonstrate their performance on CIFAR-100 using CIFAR-10 discovered policies, for a fair comparison with them, we show the transfer results of BDA on CIFAR-100 using CIFAR-10 discovered policies. Specifically, on the target dataset CIFAR-100, BDA performs better than AA and PBA on Wide-ResNet-40-2 and Wide-ResNet-28-10 using CIFAR-10 discovered policies.

Additionally, in Table 3.5, we compare our results evaluated with augmentation policies searched on CIFAR-100 with results evaluated with policies searched directly on CIFAR-10 and SVHN (image datasets with the same low-resolution) and ImageNet (a high-resolution image dataset). As shown in Table 3.5, CIFAR-100 discovered policies achieve comparable performance to the directly learned policies under

Table 3.5: The performance of transfering policies across datasets. Transfer evaluations use policies obtained from the search on reduced source datasets CIFAR-10/100 with 40k images using Wide-ResNet-40-2.

| Source → Target Datasets | Model | AA | PBA | BDA-transfer | BDA-direct |
|---|---|---|---|---|---|
| CIFAR-10 → CIFAR-100 | Wide-ResNet-40-2 | 79.3 | – | $79.454 \pm 0.0871$ | $79.61 \pm 0.261$ |
| | Wide-ResNet-28-10 | 82.9 | 83.27 | $83.424 \pm 0.107$ | $83.44 \pm 0.253$ |
| | Shake-Shake-96d | 85.7 | 84.69 | $85.013 \pm 0.117$ | $84.95 \pm 0.091$ |
| CIFAR-100 → CIFAR-10 | Wide-ResNet-40-2 | – | – | $96.44 \pm 0.074$ | $96.53 \pm 0.074$ |
| | Wide-ResNet-28-10 | – | – | $97.53 \pm 0.085$ | $97.58 \pm 0.092$ |
| | Shake-Shake-112d | – | – | $98.14 \pm 0.064$ | $98.03 \pm 0.045$ |
| | PyramidNet | – | – | $98.58 \pm 0.0$ | $98.54 \pm 0.048$ |
| CIFAR-100 → SVHN | Wide-ResNet-28-10 | – | – | $98.881 \pm 0.0296$ | $98.85 \pm 0.0001$ |
| CIFAR-100 → ImageNet | ResNet-50 | – | – | $78.187 \pm 0.109$ | $78.115 \pm 0.009$ |
| | ResNet-200 | – | – | $80.18 \pm 0.07$ | $80.144 \pm 0.008$ |

listed models on CIFAR-10, SVHN and ImageNet. This demonstrates the transferability of BDA-discovered augmentation policies across datasets with the same low-resolution images and the transferability from a low-resolution dataset to a high-resolution dataset, as well as models. Besides, Table 3.5 demonstrates the transferability of BDA-discovered policies from a dataset with a smaller number of classes (CIFAR-100) to another dataset with a larger number of classes (ImageNet).

For other stationary and non-stationary augmentation methods, e.g., AA and AdvAA, separate searches on different datasets are required to achieve comparable results. Specifically, in AA [2], Sec. 5 states "when training on SVHN, using the best policy learned on reduced CIFAR-10 does slightly improve generalization accuracy compared to the baseline augmentation, but not as significantly as applying the SVHN-learned policy". That is, for AA, CIFAR-10 discovered policies are not as good as SVHN discovered policies on augmenting SVHN dataset. Besides, in AdvAA [21], Table 6 shows the gap between the performance of the direct search and the policy transfer on different datasets. For instance, policies learned on ImageNet perform worse than those from direct searches by 0.5% and 0.99% on CIFAR-10 and CIFAR-100, respectively. Thus, augmentation policies found by BDA demonstrate a better transferability across datasets with different resolutions.

In contrast to conventional methods which may overfit policies to a specific dataset, BDA is less parameterized by imposing sparsity with Evolutionary Operation Pruning in operation pair selection, thus yielding better generalizability. Therefore, the transferability across different datasets is a merit of BDA but not demonstrated by prior works, as it saves the need for searching on specific datasets.

**Operation pair performance** In addition to searching for optimal augmentation policies, BDA can discover good and bad augmentation operation pairs for the dataset, which provides the explainability of the performance of specific augmentation operation pairs. Considering augmentation policies with the same operation pair yet different magnitude levels may have a similar performance on augmenting the dataset, we aggregate policies of the same operation pair to obtain the average reward for each operation pair. We show the performance of different operation pairs across CIFAR-10, CIFAR-100, ImageNet datasets in Figure 2. We can observe that the three datasets share some good and bad operation pairs. Specifically, operation pairs in the right parts (from Posterize to Cutout horizontally) of CIFAR-10/100 heatmaps share a similar pattern, and are mostly suitable for augmenting CIFAR-10/100. Besides, the upper right corners of CIFAR-100 and ImageNet heatmaps show a similar pattern, which indicates that these operation pairs in this corner are likely transferable between CIFAR-100 and ImageNet. Moreover, all heatmaps indicate that Invert should not be paired with operations other than Solarize, Equalize or Invert for augmenting CIFAR-100 and ImageNet. The presence of such common knowlege shared between datasets explains why BDA-discovered policies with sparse operation pair selection can transfer across different datasets.

Furthermore, to validate that the performance of the operation pair can be reflected

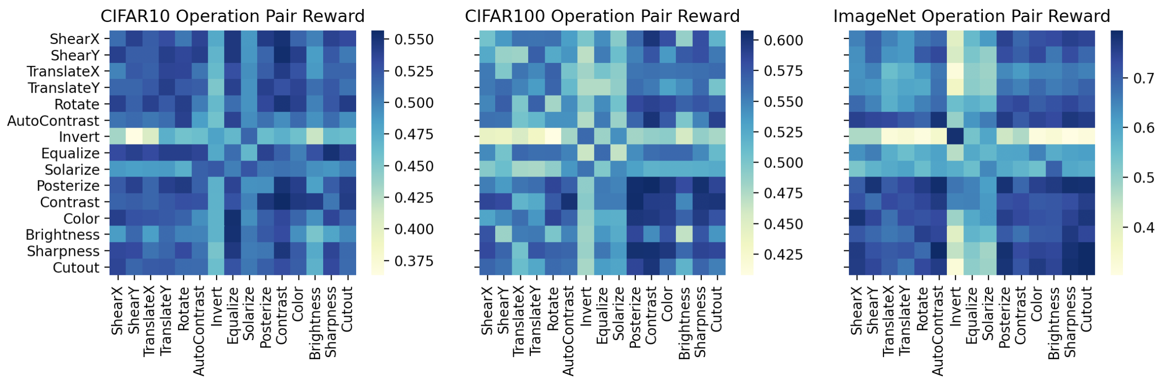


Figure 2: The heatmaps of average rewards of operation pairs obtained by searching on CIFAR-10, CIFAR-100 and ImageNet, where y-axis represents the first operation and x-axis represents the second operation.

Table 3.6: The performance on CIFAR-100 using operation pairs with top 5 average rewards and operation pairs with bottom 5 average rewards found on CIFAR-100, where the magnitude level of each operation is randomly sampled.

| Model | Top 5 Operation Pairs | Bottom 5 Operation Pairs |
|---|---|---|
| Wide-ResNet-40-2 | $76.68 \pm 0.196$ | $19.58 \pm 0.516$ |
| Wide-ResNet-28-10 | $80.39 \pm 0.02$ | $21.79 \pm 0.22$ |

by its average reward, we show the performance of CIFAR100-discovered operation pairs with top 5 average rewards and operation pairs with bottom 5 average rewards, respectively, on CIFAR-100 across listed models in Table 3.6. As shown in Table 3.6, augmenting CIFAR-100 with operation pairs with bottom 5 average rewards can noticeably decrease the network's performance on image classification. Thus, our design, which aggregates augmentation policies with the same operation pair and computes the average reward of the operation pair as the performance of the operation pair, is plausible.

### 3.4.4 Ablation Study

To demonstrate the effectiveness of the Evolutionary Operation Pruning algorithm, we perform the search procedure on CIFAR-100 with 40k training images using the BDA algorithm with a regular Successive Halving strategy [40] instead of Evolutionary

Table 3.7: Ablation study: top-1 test accuracy (%) on CIFAR-100. Evaluations use augmentation policies searched on the reduced CIFAR-100 dataset with 40k training images.

| Model | BDA-NoPrune | BDA-40k-SH | BDA-40k |
|---|---|---|---|
| Wide-ResNet-40-2 | $78.91 \pm 0.19$ | $79.23 \pm 0.28$ | **79.61 ± 0.26** |
| Wide-ResNet-28-10 | $82.79 \pm 0.29$ | $83.21 \pm 0.12$ | **83.44 ± 0.25** |

Operation Pruning. Specifically, in each round of the search, the method BDA-40k-SH abandons the worst 1/3 of augmentation policies $(O_1(\lambda_1), O_2(\lambda_2))$ (arms) from the entire search space of operations as well as their magnitudes. Additionally, we show the performance of BDA without pruning arms. That is, the method BDA-NoPrune does not abandon any augmentation policies during the search procedure. As shown in Table 3.7, BDA-40k achieves consistently better results than BDA-40k-SH and BDA-NoPrune.

Given the same search time, without pruning worse performing augmentation policies during the search procedure, BDA-NoPrune wastes computational resources on less promising augmentation polices, thus it achieves worse performance on CIFAR-100 across listed models compared with BDA-40k-SH and BDA-40k. Moreover, although BDA-40k-SH also uses a Successive Halving strategy to allocate more computational resources to potentially more promising augmentation policies, this method does not treat augmentation policies in a hierarchical format. With every augmentation operation pair treated equally, this method does not have a bias toward maintaining specifically useful operation pairs and fails to achieve sparsity in operation pair selection. Besides, without increasing the searchable magnitude levels of potentially useful operation pairs during the search procedure, BDA-40k-SH will lead to quick reductions of searchable augmentation policies. Therefore, through achieving sparse selection of operation pairs and evolving searchable magnitude levels of active operation pairs, the Evolutionary Operation Pruning algorithm can effectively enhance the performance of the proposed bandit model.

## 3.5 Summary

In this chapter, we first introduce the search space of augmentation policies. Then, we formulate the automated DA problem as a multi-armed bandit problem and give an overview of the proposed framework. Moreover, we discuss the proposed BDA algorithm, our design of the reward signal, and the Evolutionary Operation Pruning algorithm in detail. Finally, we demonstrate and analyze our experimental results on image classification benchmark datasets [52–54] across different models. In addition, we perform experiments to demonstrate the transferability of our searched policies across different datasets and perform the ablation study to demonstrate the effectiveness of our proposed method.

# Chapter 4

# The BOES Algorithm for Hyperparameter Optimization

For the automated HPO problem, as mentioned in Chapter 1, Bayesian Optimization (BO) is widely adopted for searching for the globally optimal configuration. In every search iteration, BO suggests the hyperparameter configuration that maximizes the acquisition function [45] computed by a probabilistic surrogate model [35, 42] for evaluation.

However, there are several limitations of traditional BO for HPO tasks. First, as a single surrogate model cannot efficiently approximate the objective function, vanilla BO approaches need to run many search iterations before finding good configurations, especially for high dimensional hyperparameters or a large search space. Second, as the optimal acquisition functions for different HPO tasks are different, optimizing a single acquisition function for different HPO tasks may lead to sub-optimal solutions. HEBO [50] uses a multi-objective evolutionary optimizer to optimize multiple acquisition functions simultaneously to sample better configurations across various HPO tasks. However, as training a multi-objective optimizer requires more training data compared to a single-objective optimizer, HEBO may need to explore more configurations to find the optimal one. Finally, traditional BO sequentially suggests a single configuration for evaluation in each search iteration [61], and can not leverage the currently widespread parallel computing resources [62] for faster evaluation.

Batch BO [63–65] reduces the number of search iterations by suggesting a batch of configurations in each iteration to be evaluated in parallel.

In this thesis, we propose the BOES algorithm to efficiently and robustly solve various HPO tasks in machine learning. BOES is a batch algorithm that can leverage parallel computing resources and perform global search of hyperparameters within a small number of iterations. BOES uses a hybrid sampler that samples more diverse configuration candidates both from a BO acquisition function and from a distribution statistically learned via an evolutionary strategy. These candidates are then ranked by an ensemble surrogate model that leverages the power of multiple probabilistic models to reduce estimation variance. Due to the ensemble surrogate model, BOES is robust and can be used to solve a range of different HPO tasks. Meanwhile, compared to the multi-objective HEBO [50] (the best solution from the NeuralIPS 2020 black-box optimization challenge), BOES uses a single-objective hybrid sampler and is more efficient; it reaches better performance than HEBO at the same number of evaluations. Our main contributions are summarized as follows:

First, we propose a hybrid sampler consisting of TPE [35], CMA-ES [31, 32], and Latin hypercube [66] to sample multiple configuration candidates to be evaluated by an ensemble surrogate model. This hybrid sampler can better balance exploitation and exploration than using a single sampler during the search. While the Latin hypercube sampler randomly samples configurations that are well spread across the whole search space, the TPE and CMA-ES samplers suggest configurations from learned distributions of the reward history.

Second, we propose an ensemble surrogate model consisting of different probabilistic models to predict the performance of hyperparameter configurations generated by the hybrid sampler and suggest a batch of best configurations for parallel evaluation. The ensemble model can better generalize to any objective (reward) function which could have an arbitrary dependence on various performance metrics in a real-world HPO task, without assuming knowledge of its priors.

We conduct extensive experiments on 20 HPO datasets that are derived from realistic yet anonymized industrial tasks, including optimizing hyperparameters for models or strategies in recommendation systems, etc., in the Automated HPO Contest in QQ Brower 2021 AI Algorithm Competition at CIKM 2021 AnalytiCup. Our proposed method ranked the 4th and 7th places in the final contest of this competition, for the training and tournament stages, respectively.

## 4.1 Batch Hyperparameter Optimization

The HPO task aims to solve the following optimization problem:

$$\max_{\mathbf{x} \in \mathrm{X}} f(\mathbf{x}), \tag{4.1}$$

where $\mathbf{x}$ is the feature vector of the hyperparameter configuration belonging to the search space X, $f(\mathbf{x})$ is the reward function (the objective function) maximized by the search algorithm.

For the batch HPO algorithm, in each search iteration, the search algorithm receives a reward history of explored configurations. Based on the reward history, the search algorithm suggests a batch of configurations that are to be explored and receive the corresponding rewards in the next iteration. The goal of the search algorithm is to find the hyperparameter configuration with the maximum reward under limited search iterations.

## 4.2 The BOES Algorithm

### 4.2.1 Algorithm Overview

Our proposed BOES algorithm is described in Algorithm 4 as follows: When no hyperparameter configuration is explored, in line 4, we use the Latin Hypercube sampling to sample *num_suggest* configurations that are well spread across the search space. Otherwise, in line 6, we use the configuration feature $\mathbf{x}$ and the corresponding reward $f(\mathbf{x})$ from the reward history to train the TPE sampler [35], the CMA-ES

**Algorithm 4:** The BOES Algorithm

```
   Input: The objective function $f$, the search space X
   Output: The hyperparameter configuration $\mathbf{x}$ with the maximal reward in $H$
1  Initialize the Ensemble Model $EM$, the TPE sampler $s_1$, the CMA-ES
     sampler $s_2$, the reward history list $H = \{\}$
2  for iteration $t = 1, \ldots,$ total_iterations do
3      if $len(H) = 0$ then
4          suggested_configs =
             LatinHypercubeSample(X, num_suggest, rand_seed)
5      else
6          Fit the TPE sampler $s_1$, the CMA-ES sampler $s_2$, every surrogate
             models $\theta$ in the Ensemble Model $EM$ to the reward history $H$
7          suggested_configs = $\{\}$
8          for $i = 1, \ldots,$ num_suggest do
9              suggested_configs =
                 suggested_configs $\bigcup$ {Suggest_Config($s_1, s_2, EM$,
                 X, num_tpe, num_cma, num_latin)}
10     for $i = 1, \ldots,$ num_suggest do
           /* $\mathbf{x}_i$ is the ith configuration in suggested_configs   */
11         $H = H \bigcup \{(\mathbf{x}_i, f(\mathbf{x}_i))\}$
12     Run the early-stopping strategy to prune configurations with worse
         intermediate rewards (for the final contest)
13 $\mathbf{x} = \text{argmax}_{\mathbf{x} \in H} f(\mathbf{x})$
```

sampler [31, 32], and surrogate models in the Ensemble model. Then, in lines 8-9, we use Algorithm 5 Suggest_Config to obtain *num_suggest* configurations suggested from the Ensemble model. As shown in Algorithm 5, first, we sample the given numbers of configurations with the TPE sampler, the CMA-ES sampler, and the Latin Hypercube sampler [66], respectively. In lines 3-4 of Algorithm 5, given the sampled configurations and the acquisition function, we obtain the top configurations composed of the configuration with the highest Probability of Improvement (PI) [45] value suggested by every surrogate model in the Ensemble model. Algorithm 5 returns the configuration with the highest frequency in the top configurations. After obtaining *num_suggest* configurations to be explored, in lines 10-11 of Algorithm 4, we get the corresponding rewards from the objective function and update the reward history. Moreover, we use a heuristic early-stopping strategy to prune configurations

```
Algorithm 5: Suggest_Config
  Input: the TPE sampler $s_1$, the CMA-ES sampler $s_2$, the Ensemble Model
         $EM$, the search space X, the number of configurations sampled by
         TPE, CMA-ES and Latin Hypercube: $\mathrm{num\_tpe}, \mathrm{num\_cma}, \mathrm{num\_latin}$
  Output: The hyperparameter configuration $\mathbf{x}$ suggested by the Ensemble
          Model $EM$
  /* Sample configs using TPE, CMA-ES, LatinHypercube sampling  */
1 $\mathrm{sampled\_configs} = \{\mathrm{TPESample}(X, \mathrm{num\_tpe}, s_1)\}$
    $\bigcup\{\mathrm{CMAESSample}(X, \mathrm{num\_cma}, s_2)\}$
    $\bigcup\{\mathrm{LatinHypercubeSample}(X, \mathrm{num\_latin}, \mathrm{rand\_seed})\}$
2 $\mathrm{top\_configs} = \{\}$
3 for surrogate model $\theta$ in the Ensemble model $EM$ do
4     $\mathrm{top\_configs} = \mathrm{top\_configs} \bigcup \mathrm{argmax}_{\mathbf{x} \in \mathrm{sampled\_configs}} \mathrm{PI}(\mathbf{x}|\theta)$
5 $\mathbf{x} = \mathrm{argmax}_{\mathbf{x} \in \mathrm{top\_configs}} freq(\mathbf{x})$
```

with worse intermediate rewards in the final contest. Thus, the algorithm wastes fewer search resources on less promising configurations. Algorithm 4 returns the hyperparameter configuration with the highest accurate reward in the reward history.

### 4.2.2 The Hybrid Sampler

To obtain a batch of suggested hyperparameter configurations, we use the hybrid sampler consisting of TPE [35], CMA-ES [31, 32], Latin Hypercube [66] samplers to sample a set of configuration candidates and feed them to the Ensemble model to obtain each suggested configuration. With the help of the TPE sampler, hyperparameter configurations that maximize the Expected Improvement (EI) [45] acquisition function can be sampled from the learned posterior distribution of the objective function $p_{s_1}(f|H)$. In addition, instead of sampling configuration candidates from only one learned distribution, by adding configurations sampled from the multivariate gaussian distribution of the trained CMA-ES sampler and the configurations sampled with Latin Hypercube (pure exploration sampling) to the configuration candidates, the hybrid sampler balances the exploitation and the exploration better by providing more knowledge and diversity from different sampling methods to the configuration candidates.

### 4.2.3 The Ensemble Surrogate Model

As we need to approximate various objective functions accurately with a small number of explored configurations, it is significant to leverage the capability of multiple probabilistic surrogate models. We use an ensemble model consisting of an Extra Tree (ET) [67], a Random Forest (RF) [68], a Gradient Boosting Tree (GBT) [69], and a Gaussian Process (GP) [42] regressors as probabilistic surrogate models. Each surrogate model gives a vote to the configuration with the highest PI [45] acquisition score from the given set of sampled configuration candidates. We use the max-voting technique from ensemble learning [70, 71] to select the configuration that receives the most votes as the best configuration to be explored in the next iteration. As the Ensemble model selects the configuration based on the PI acquisition function from the configuration candidates sampled from different learned distributions, our proposed algorithm implicitly optimizes multiple acquisition functions (EI and PI) which is more effective than optimizing a single acquisition function. In addition, the combination of the hybrid sampler and the Ensemble surrogate model provides a robust and effective way for solving the exploitation-vs-exploration dilemma in the HPO problem.

### 4.2.4 The Early-stopping Strategy

We propose a heuristic early-stopping (EarlyStop) strategy to stop exploring unpromising configurations for better search resources allocation in the final contest where the explored configuration receives the accurate reward until 14 successive search iterations. Specifically, we prune configurations whose intermediate reward's upper confidence bound is less than the maximal value of the accurate rewards of explored configurations. In addition, we prune configurations with more than 3 iterations whose intermediate reward is less than the 75% percentile of the latest intermediate rewards of explored configurations.

## 4.3 Experiments and Results

In this section, we evaluate our proposed BOES algorithm on 20 HPO datasets aiming to optimize different objective functions for realistic industrial tasks of recommendation systems from the HPO Contest in the QQBrower 2021 AI Algorithm Competition. We first introduce the preliminary and the final HPO tasks, as well as the search space, for this contest. Then, we analyze our performance compared to various baseline models, such as Random Search [27], GP [42], TPE [35], etc., and HEBO [50] (the best solution from the NeurIPS 2020 black-box optimization challenge).

### 4.3.1 Hyperparameter Optimization Tasks

**The Preliminary Contest** In the preliminary contest, the total number of iterations that the search algorithm can run is 20. In each iteration, the search algorithm suggests 5 hyperparameter configurations from the search space under 30 seconds, which will receive the corresponding accurate rewards of the suggested configurations in the next iteration. Thus, the number of configurations that can be explored by the search algorithm is 100.

**The Final Contest** In real-life scenarios, as computing the accurate reward for the hyperparameter configuration is very time-consuming, the search algorithm usually uses the intermediate rewards to approximate the performance of the configuration and prune those with worse performance early. Based on this scenario, in the final contest, each hyperparameter configuration will receive its accurate reward until having 14 successive iterations, where each intermediate iteration returns an average reward and the 95% confidence interval of the reward for this configuration. The total number of iterations that the search algorithm can run is 140. As only the configurations with the accurate rewards (the final reward received at the 14th iteration) are considered in the final evaluation, the number of configurations that can be accurately explored by the search algorithm is from 0 to 50.

Table 4.1: The average accuracy of 20 datasets under 10 random seeds in the preliminary contest.

| Random | GP | RF | ET | GBT | Ensemble | TPE | Ensemble + TPE | BOES - Latin | BOES + LocalSearch | BOES |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.1 | 0.299 | 0.488 | 0.631 | 0.460 | 0.718 | 0.459 | 0.735 | 0.768 | 0.784 | **0.827** |

**Search Space** There are 20 optimization tasks in the online test evaluation for this contest. For each optimization task, the number variables of the hyperparameter configuration is in the range of 2 (inclusive) to 6 (inclusive). Moreover, the number of choices of each hyperparameter variable is in the range of 0 (exclusive) to $20,000$ (inclusive). Thus, the size of the search space is in the following range:

$$10^4 \leq dim(\mathrm{X}) \leq 10^{25}. \tag{4.2}$$

## 4.3.2 Results and Analysis

**The Preliminary Contest.** Table 4.1 summarizes our results on 20 datasets where the average accuracy is averaged over all datasets and repeated under 10 random seeds. As shown in Table 4.1, the performance of the ensemble surrogate model consisting of GP, RF, ET, and GBT is noticeably better than the performance of each single surrogate model. These results demonstrate our proposed ensemble surrogate model can leverage the power of single surrogate models and approximate various objective functions better.

Moreover, as the surrogate models mentioned above suggest hyperparameter configurations from a randomly selected set of configuration candidates, we also conduct experiments using methods that sample configuration candidates based on the historical rewards. As shown in Table 4.1, the proposed hybrid sampler of BOES, which consists of TPE, CMA-ES, and Latin Hypercube, performs noticeably better than using a single BO or Evolutionary sampler to sample candidates for the ensemble surrogate model. Since the hybrid sampler samples promising configuration candidates from distributions learned from TPE and CMA-ES models, it exploits the knowledge learned from the explored configurations. Also, Latin Hypercube samples configu-

Table 4.2: The average accuracy of 20 datasets under 10 random seeds in the final contest.

| HEBO + no EarlyStop | BOES + no EarlyStop | BOES + SH | BOES + proposed EarlyStop |
|---|---|---|---|
| 0.291 | 0.551 | 0.494 | **0.781** |

rations that are well spread across the whole search space provide the exploration to the configuration candidates. Thus, our proposed hybrid sampler demonstrates a robust and good ability to balance the exploitation and the exploration during the search procedure under limited search resources. Furthermore, we show that adding the local search to our proposed global search method is not beneficial to optimize the given recommendation system datasets in this contest.

**The Final Contest** Table 4.2 summarizes our results in the final contest compared to HEBO [50] without EarlyStop, the classic pruning strategy Successive Halving (SH) [40], and proposed heuristic EarlyStop strategy. Under the experimental setting of no EarlyStop strategy where only 50 configurations are explored, our proposed approach achieves significantly higher performance than HEBO. This demonstrates our proposed method is more efficient and effective compared to HEBO under small search iterations. HEBO uses the multi-objective optimization method to optimize multiple acquisition functions simultaneously that may require more training examples. Instead, our proposed hybrid sampler and ensemble model implicitly optimize multi-objective acquisition functions through training the TPE sampler and the ensemble model with EI and PI acquisition functions, respectively. Moreover, our proposed heuristic EarlyStop strategy performs noticeably better than the SH pruning strategy in the final contest. This result demonstrates the capability of our heuristic EarlyStop strategy to prune unpromising configurations based on intermediate rewards and allocate search resources to important search regions.

## 4.4 Summary

In this chapter, we first introduce the batch HPO task. Then, we describe our proposed BOES algorithm for the automated HPO problem in detail. Finally, we demonstrate the performance of our proposed algorithm on 20 HPO datasets from recommendation system scenarios provided in the Automated HPO Contest in the QQBrower 2021 AI Algorithm Competition.

# Chapter 5

# Conclusion and Future Work

In this thesis, we study two problems from the AutoML research area: automated DA and automated HPO. For the automated DA problem, we propose a bandit-based search algorithm to search for the optimal set of augmentation policies. For the automated HPO problem, we propose a hybrid Bayesian Optimization and Evolutionary strategy sampler and an ensemble probabilistic surrogate model to search for the globally optimal hyperparameter configuration more efficiently.

## 5.1 Automated Data Augmentation

For the automated DA problem, we propose a novel bandit-based search algorithm for searching for effective and transferable stationary augmentation policies across different datasets.

Our extensive experiments demonstrate that BDA can generate effective augmentation policies within a reasonable search cost for the target dataset. Specifically, on CIFAR-10/100 datasets, BDA achieves comparable or better results than previous automated augmentation methods, e.g., AA, FastAA, RandAA, DADA, and PBA, across various models with a low search cost. Moreover, on high-resolution image datasets like ImageNet, BDA achieves the best performance on ResNet-50 than previous methods producing stationary policies, including AA, FastAA, RandAA, and DADA. Additionally, BDA reduces the search cost noticeably on large and high-resolution image datasets. On ImageNet, BDA is 536 times faster than AA and is 16 times fast than FastAA. Moreover, in contrast to previous approaches that are highly parameterized and does not demonstrate good transferability of augmentation policies across different datasets, BDA is a non-parametric search algorithm for discovering augmentation policies with good transferability across datasets, which is not shown by prior methods.

## 5.2 Automated Hyperparameter Optimization

For the automated HPO problem, we propose an efficient batch HPO algorithm combining Bayesian Optimization and Evolutionary Strategy.

Our extensive experiments on the datasets of recommendation systems demonstrate the effectiveness of BOES. In addition, compared with HEBO [50] (the best solution from the NeuralIPS 2020 black-box optimization challenge), BOES performs noticeably better in the final contest. This result demonstrates that our method performs better under limited search iterations. Futhermore, BOES ranked the 4th and 7th places in the training and tournament stages of the final HPO Contest in QQ Brower 2021 AI Algorithm Competition, respectively.

## 5.3 Future Work

**Automated Data Augmentation.** For the automated DA problem, as it is beneficial to search for different augmentation policies for different images, we would focus on deriving a less-parameterized search algorithm with the effect of data variation considered. We would try the contrastive learning technique [72] to extract robust image representations. Then, we would try the contextual-based bandit algorithm to take the image features as the input, or we would combine the clustering algorithm and our proposed bandit-based search algorithm BDA to search for augmentation policies for each image cluster.

**Automated Hyperparameter Optimization.** For the automated HPO problem, we would try combining the TuRBO [48] technique and our proposed BOES algorithm to develop an efficient search algorithm for large-scale high-dimensional HPO problems.

# Bibliography

[1] F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated machine learning: methods, systems, challenges*. Springer Nature, 2019.

[2] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, “Autoaugment: Learning augmentation policies from data,” *arXiv preprint arXiv:1805.09501*, 2018.

[3] M. Feurer and F. Hutter, “Hyperparameter optimization,” in *Automated machine learning*, Springer, Cham, 2019, pp. 3–33.

[4] W. S. Sarle, “Stopped training and other remedies for overfitting,” *Computing science and statistics*, pp. 352–360, 1996.

[5] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, “Regularization of neural networks using dropconnect,” in *International conference on machine learning*, PMLR, 2013, pp. 1058–1066.

[6] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, “Mixup: Beyond empirical risk minimization,” *arXiv preprint arXiv:1710.09412*, 2017.

[7] T. DeVries and G. W. Taylor, “Improved regularization of convolutional neural networks with cutout,” *arXiv preprint arXiv:1708.04552*, 2017.

[8] H. Inoue, “Data augmentation by pairing samples for images classification,” *arXiv preprint arXiv:1801.02929*, 2018.

[9] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, “Cutmix: Regularization strategy to train strong classifiers with localizable features,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 6023–6032.

[10] M. Paschali *et al.*, “Data augmentation with manifold exploring geometric transformations for increased performance and robustness,” *arXiv preprint arXiv:1901.04420*, 2019.

[11] P. Y. Simard, D. Steinkraus, J. C. Platt, *et al.*, “Best practices for convolutional neural networks applied to visual document analysis.,” in *Icdar*, Citeseer, vol. 3, 2003.

[12] H. S. Baird, H. Bunke, and K. Yamamoto, *Structured document image analysis*. Springer Science & Business Media, 2012.

[13] I. Sato, H. Nishimura, and K. Yokoi, “Apac: Augmented pattern classification with neural networks,” *arXiv preprint arXiv:1505.03229*, 2015.

[14] D. Han, J. Kim, and J. Kim, “Deep pyramidal residual networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5927–5935.

[15] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li, “Bag of tricks for image classification with convolutional neural networks,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 558–567.

[16] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin, “Albumentations: Fast and flexible image augmentations,” *Information*, vol. 11, no. 2, p. 125, 2020.

[17] L. Perez and J. Wang, “The effectiveness of data augmentation in image classification using deep learning,” *arXiv preprint arXiv:1712.04621*, 2017.

[18] S. Lim, I. Kim, T. Kim, C. Kim, and S. Kim, “Fast autoaugment,” in *Advances in Neural Information Processing Systems*, 2019, pp. 6665–6675.

[19] Y. Li, G. Hu, Y. Wang, T. Hospedales, N. M. Robertson, and Y. Yang, “Differentiable automatic data augmentation,” in *European Conference on Computer Vision*, Springer, 2020, pp. 580–595.

[20] D. Ho, E. Liang, X. Chen, I. Stoica, and P. Abbeel, “Population based augmentation: Efficient learning of augmentation policy schedules,” in *International Conference on Machine Learning*, PMLR, 2019, pp. 2731–2741.

[21] X. Zhang, Q. Wang, J. Zhang, and Z. Zhong, “Adversarial autoaugment,” *arXiv preprint arXiv:1912.11188*, 2019.

[22] F. Zhou *et al.*, “Metaaugment: Sample-aware data augmentation policy learning,” *arXiv preprint arXiv:2012.12076*, 2020.

[23] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, “Efficient neural architecture search via parameters sharing,” in *International Conference on Machine Learning*, PMLR, 2018, pp. 4095–4104.

[24] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, “Hyperband: A novel bandit-based approach to hyperparameter optimization,” *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6765–6816, 2017.

[25] S. Falkner, A. Klein, and F. Hutter, “Bohb: Robust and efficient hyperparameter optimization at scale,” in *International Conference on Machine Learning*, PMLR, 2018, pp. 1437–1446.

[26] P. Lerman, “Fitting segmented regression models by grid search,” *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 29, no. 1, pp. 77–84, 1980.

[27] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization.,” *Journal of machine learning research*, vol. 13, no. 2, 2012.

[28] J. Snoek, H. Larochelle, and R. P. Adams, “Practical bayesian optimization of machine learning algorithms,” *Advances in neural information processing systems*, vol. 25, 2012.

[29] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.

[30] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multi-armed bandit problem," *Machine learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[31] N. Hansen, "The cma evolution strategy: A comparing review," *Towards a new evolutionary computation*, pp. 75–102, 2006.

[32] N. Hansen, "The cma evolution strategy: A tutorial," *arXiv preprint arXiv:1604.00772*, 2016.

[33] A. Mikołajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," in *2018 international interdisciplinary PhD workshop (IIPhDW)*, IEEE, 2018, pp. 117–122.

[34] Q. Xie, Z. Dai, E. Hovy, M.-T. Luong, and Q. V. Le, "Unsupervised data augmentation for consistency training," *arXiv preprint arXiv:1904.12848*, 2019.

[35] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," *Advances in neural information processing systems*, vol. 24, 2011.

[36] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 702–703.

[37] K. Tian, C. Lin, M. Sun, L. Zhou, J. Yan, and W. Ouyang, "Improving auto-augment via augmentation-wise weight sharing," *arXiv preprint arXiv:2009.14737*, 2020.

[38] M. Jaderberg *et al.*, "Population based training of neural networks," *arXiv preprint arXiv:1711.09846*, 2017.

[39] M. Ren, W. Zeng, B. Yang, and R. Urtasun, "Learning to reweight examples for robust deep learning," in *International Conference on Machine Learning*, PMLR, 2018, pp. 4334–4343.

[40] K. Jamieson and A. Talwalkar, "Non-stochastic best arm identification and hyperparameter optimization," in *Artificial Intelligence and Statistics*, PMLR, 2016, pp. 240–248.

[41] 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.

[43] P. I. Frazier, "A tutorial on bayesian optimization," *arXiv preprint arXiv:1807.02811*, 2018.

[42] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer school on machine learning*, Springer, 2003, pp. 63–71.

[44] D. A. Reynolds, "Gaussian mixture models.," *Encyclopedia of biometrics*, vol. 741, pp. 659–663, 2009.

[45] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht, "The marginal value of adaptive gradient methods in machine learning," *arXiv preprint arXiv:1705.08292*, 2017.

[46] Y. Gao, T. Yu, and J. Li, "Bayesian optimization with local search," in *International Conference on Machine Learning, Optimization, and Data Science*, Springer, 2020, pp. 350–361.

[47] C. Wang, Q. Wu, S. Huang, and A. Saied, "Economic hyperparameter optimization with blended search strategy," in *International Conference on Learning Representations*, 2020.

[48] D. Eriksson, M. Pearce, J. Gardner, R. D. Turner, and M. Poloczek, "Scalable global optimization via local bayesian optimization," *Advances in Neural Information Processing Systems*, vol. 32, pp. 5496–5507, 2019.

[49] W. Lyu, F. Yang, C. Yan, D. Zhou, and X. Zeng, "Batch bayesian optimization via multi-objective acquisition ensemble for automated analog circuit design," in *International conference on machine learning*, PMLR, 2018, pp. 3306–3314.

[50] A. I. Cowen-Rivers *et al.*, "Hebo: Heteroscedastic evolutionary bayesian optimisation," *arXiv e-prints*, arXiv–2012, 2020.

[51] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[52] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[53] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.

[54] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[55] H. Chen *et al.*, "Anti-bandit neural architecture search for model defense," *arXiv preprint arXiv:2008.00698*, 2020.

[56] A. Graves, M. G. Bellemare, J. Menick, R. Munos, and K. Kavukcuoglu, "Automated curriculum learning for neural networks," *arXiv preprint arXiv:1704.03003*, 2017.

[57] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[58] X. Gastaldi, "Shake-shake regularization," *arXiv preprint arXiv:1705.07485*, 2017.

[59] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[60] E. Hoffer, T. Ben-Nun, I. Hubara, N. Giladi, T. Hoefler, and D. Soudry, "Augment your batch: Better training with larger batches," *arXiv preprint arXiv:1901.09335*, 2019.

[61] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *International conference on learning and intelligent optimization*, Springer, 2011, pp. 507–523.

[62] R. Bekkerman, M. Bilenko, and J. Langford, *Scaling up machine learning: Parallel and distributed approaches*. Cambridge University Press, 2011.

[63] J. González, Z. Dai, P. Hennig, and N. Lawrence, "Batch bayesian optimization via local penalization," in *Artificial intelligence and statistics*, PMLR, 2016, pp. 648–657.

[64] N. Awad *et al.*, "Squirrel: A switching hyperparameter optimizer," *arXiv preprint arXiv:2012.08180*, 2020.

[65] M. Sazanovich, A. Nikolskaya, Y. Belousov, and A. Shpilman, "Solving blackbox optimization challenge via learning search space partition for local bayesian optimization," in *NeurIPS 2020 Competition and Demonstration Track*, PMLR, 2021, pp. 77–85.

[66] J. C. Helton and F. J. Davis, "Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems," *Reliability Engineering & System Safety*, vol. 81, no. 1, pp. 23–69, 2003.

[67] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine learning*, vol. 63, no. 1, pp. 3–42, 2006.

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

[69] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[70] R. Polikar, "Ensemble learning," in *Ensemble machine learning*, Springer, 2012, pp. 1–34.

[71] Z.-H. Zhou, "Ensemble learning," in *Machine Learning*, Springer, 2021, pp. 181–210.

[72] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*, PMLR, 2020, pp. 1597–1607.