Applied Intelligence (2023) 53:8978–8993
https://doi.org/10.1007/s10489-022-03790-z

BO-Aug: learning data augmentation policies via Bayesian optimization

Chunxu Zhang[1,2] · **Ximing Li**[1,2] · **Zijian Zhang**[1,2] · **Jiaxu Cui**[1,2] · **Bo Yang**[1,2]

Accepted: 18 May 2022 / Published online: 5 August 2022
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract
Data augmentation has been an essential technique to increase the amount and diversity of datasets, thus improving deep learning models. To pursue further performance, several automated data augmentation approaches have recently been proposed to find data augmentation policies automatically. However, there are still some key issues that deserve further exploration, *i.e.,* a precise policy search space definition, the instructive policy evaluation method, and the low computational cost of policy search. In this paper, we propose a novel method named BO-Aug that attempts to solve the above issues. Empirical verification on three widely used image classification datasets shows that the proposed method can achieve state-of-the-art or comparable performance compared with advanced automated data augmentation methods, with a relatively low cost. Our code is available at https://github.com/Zhangcx19/BO-Aug.

Keywords Data augmentation · Bayesian optimization · Neural network

1 Introduction

Data **A**ugmentation (**DA**) focuses on the generation of semantically preserved data augmented such that its amount and diversity are increased simultaneously [1–7]. DA improves deep learning approaches that prefer large training datasets. Due to its effectiveness, DA has been extensively applied in the various paradigms of supervised learning [8, 9], semi-supervised learning [10, 11], and unsupervised learning [12–14]. For example, DA can be used to define consistency regularization in self-supervised models [15, 16] and predictive tasks in contrastive learning [17].

Generally, DA has the merit of easy implementation. In terms of the image data, *i.e.,* the focus of this paper, there are dozens of easy-to-implement basic DA operations, *e.g.,* Invert, Equalize, and Rotate, as listed in Table 8 in Appendix A. However, as a coin has two sides, applying a single basic DA operation may not be effective enough to enrich the data diversity due to its simplicity. Combining a set of DA operations sounds like a good proposal; however, manually designing an effective DA operation combination demands huge human effort. To make matters worse, a DA operation combination may be data-specific or model-specific, which requires more labor consumption due to the limited reusability.

To resolve the aforementioned problems, much more attention has been recently paid to the emerging **Auto**mated **D**ata **A**ugmentation (**AutoDA**), where given a set of basic DA operations, AutoDA aims to automatically search a candidate set of optimal DA policies for a target dataset [1–3]. In existing AutoDA implementations, each DA policy is formulated as a coding sequence, expressing the order, probabilities, and magnitudes of a set of possible basic DA operations, as illustrated in Fig. 1. Under a certain DA

✉ Bo Yang
ybo@jlu.edu.cn

Chunxu Zhang
cxzhang19@mails.jlu.edu.cn

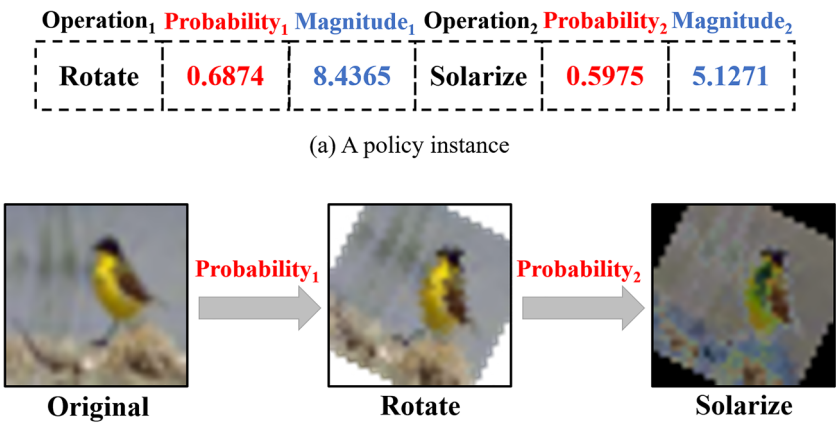
Ximing Li
liximing86@gmail.com

Zijian Zhang
zhangzj2114@mails.jlu.edu.cn

Jiaxu Cui
cjx@jlu.edu.cn

[1] College of Computer Science and Technology, Jilin University, No.2699, Qianjin Street, Chaoyang District, Changchun City, 130012, Jilin Province, China

[2] Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, No.2699, Qianjin Street, Chaoyang District, Changchun City, 130012, Jilin Province, China



(a) A policy instance

(b) Apply the policy on an image

Fig. 1 (a) A policy instance. It consists of two DA operations with associated application parameters. Specifically, it is a 6-dimensional vector. The first dimension indicates the first DA operation type. The second dimension represents the probability that the operation will be executed. The third dimension indicates the application magnitude of the operation. The meaning of the next three dimensions is similar. (b) Apply the policy on an image. Specifically, applying the two operations with associated parameters consecutively. As a result, there are four possible augmented images in total. Here shows the image augmented by both operations in the policy

policy definition, they select optimal DA coding sequences as a turn-based search game. In each round of iteration, a **policy selector** draws a candidate DA coding sequence from the defined policy space, and the candidate is evaluated by a **policy evaluator**. The policy selector is then updated by an online **update** method, where it treats the pair of the candidate and its evaluation score as a new training instance. Repeating this procedure until convergence, one can finally select a set of best DA policies with the highest evaluation scores.

Table 1 summarizes the existing advanced methods in the AutoDA community. Three key issues deserve further exploration. First, for policy space definition. Most existing AutoDA methods search policies in a discrete space, which means they determine the policy application probability and application magnitude as several discrete values according to expert knowledge. It is conceivable that this will hurt policy precision and policy search performance. Therefore, is it possible to propose an AutoDA method that functions within a continuous policy space? Second, for policy evaluation. All the existing AutoDA methods search and evaluate the individual policy independently, but policies perform jointly in the physical application. Searching for an optimal policy set is closer to the application objective than searching for a group of individually optimal policies. However, searching the complete policy set will increase the search space and difficulty significantly. Therefore, it is crucial to design a policy evaluation method to approximate its physical application without increasing the search space. Third, we consider policy search costs. In general, the policy space is always large. As a result, searching for an optimal set of policies from such space would be difficult and time-consuming. A constant concern of the community is how to design a policy search algorithm that can find the optimal DA policies with as little computational cost as possible.

In this paper, we propose a novel AutoDA method, **B**ayesian **O**ptimization **Aug**mentation (**BO-Aug**), aiming to explore the above important community issues. Specifically, we define a continuous policy space. We keep the policy application parameters as real numbers, leading to precise policy. To solve the second problem, we design a novel policy evaluation method. For a candidate policy,

Table 1 Summary of advanced AutoDA methods and proposed BO-Aug

Method	Policy Space	Policy Evaluation Method	Policy Selector
AutoAugment [1]	discrete (10^{32})	Evaluate single policy	Sampling by algorithm
PBA [2]	discrete (10^{61})	Evaluate single policy	Random sampling
FastAA [3]	continuous	Evaluate single policy	Sampling by algorithm
BO-Aug (**Ours**)	continuous	Evaluate policy group	Sampling by algorithm

Most AutoDA methods search policy in discrete space is defined with expert knowledge. All of the methods except evaluate policy individually rather than jointly, while the latter makes it closer to physical policy application. For the policy selector, PBA samples policies randomly, while the others employ algorithms to sample such as RNN in AutoAugment

we evaluate the policy group containing the candidate, to approximate the physical application situation without increasing the search space. These two points help to create a more reliable policy search foundation that facilitates the search process. To further reduce the search cost, we employ Bayesian Optimization (BO) as the search algorithm. BO is an effective global optimization algorithm that is quite suitable for solving complex optimization problems. It fits the real objective function with a probabilistic surrogate model and utilizes historical evaluation information to effectively improve search efficiency. To evaluate the performance of BO-Aug and compare it fairly with other AutoDA methods, we validate BO-Aug on three benchmark image classification datasets. The empirical results verify the proposed method's efficacy and efficiency.

The main contributions can be summarized as follows,

1) We systematically review existing AutoDA methods and formulate the AutoDA process for the first time. In addition, we identify three key issues in the AutoDA community that deserve to be explored.
2) We propose a novel AutoDA method, BO-Aug, which enjoys a more reliable policy search foundation by defining a continuous search space and evaluating the policy group. Furthermore, we take BO as the search algorithm to improve search efficiency.
3) We validate BO-Aug on the image classification task. Empirical verification shows that BO-Aug can achieve state-of-the-art or comparable performance compared with advanced AutoDA methods, with a relatively low cost.

The paper is specifically organized as follows. First, we review the related work on automated data augmentation and Bayesian Optimization in Section 2. Then, we propose the method in Section 3. Section 4 demonstrates the effectiveness of BO-Aug through empirical verification and model analyses on the image classification task. Finally, we conclude the paper in Section 5.

2 Related work

2.1 Data augmentation

In this subsection, we present a thorough review of related work in the DA community. Specifically, we first summarize the existing data augmentation methods, especially the automated DA methods, which are the focus of this paper. Then, we analyze the disadvantages in the current AutoDA research community, which motivates us to propose BO-Aug.

2.1.1 Summary of existing methods

DA is an effective method to enlarge the dataset and increase its diversity. It has been an indispensable component of recent deep learning models to alleviate overfitting and improve generalization [9, 18, 19]. Previous attempts of data augmentation include Cutout [20] and Random Erasing [21], which randomly mask out square regions of an image to improve the performance of the convolutional neural networks. Mixup [22] constructs virtual instances by combining pairs of instances and labels to increase the robustness of neural networks. Generative Adversarial Network (GAN) is a common method used for generating synthetic data [23–25]. For instance, [25] utilizes an attributes preserving GAN to transfer the object from some other datasets to the target scene to alleviate the occlusion issue in the pedestrian detection task. Additionally, there are several works aiming to augment in the latent space by operating the hidden representation vectors. For example, [26] generates new samples by interpolating the hidden representation learned by an auto-encoder. Wang et al. [27] captures the intra-class variations by estimating the covariance matrix of features and then draws the random vectors based on the estimated covariance matrix to augment the samples in that class. Recently, thanks to the advanced performance in architecture search [28–31], several Automated Data Augmentation methods have been proposed and achieve new state-of-the-art performance. Generally, AutoDA encodes the basic image transformation operation into policies and searches the optimal policies for downstream task evaluation. Compared with a single data augmentation operation, AutoDA can find an optimal combination of operations, which increases the augmentation diversity and achieves better performance. Furthermore, the policies found by AutoDA are generally transferable among different architectures and even datasets, which avoids designing different augmentation architectures for specific models manually, as GAN-based and latent space augmentation methods do. In this paper, we focus on AutoDA research and next introduce the representative methods.

AutoAugment [1] formulates AutoDA as an optimization task for the first time and achieves advanced results. Specifically, it defines the optimization problem in a discrete policy search space and introduces Reinforcement Learning (RL) as the search algorithm. PBA [2] claims that different stages during model training should adopt different augmentation operations and proposes to search augmentation schedules with the Population Based Training (PBT) algorithm. FastAA [3] incorporates density matching which constrains the policies minimizing the distance between the augmented data and training data. In detail, FastAA evaluates the searched policy performance by

testing a proxy model trained on training data, which reduces the computational cost significantly. RandAA [32] proposes removing the policy search process and treating DA operation application parameters, including operation number and magnitude, as hyperparameters. Therefore, it adopts the hyperparameter optimization method (*e.g.,* grid search) to find the optimal DA operation application parameters. GAutoAugment [33] claims that the augmented samples should cover the search space and proposes to use a greedy-based search algorithm to expand the policies dynamically. AWS [34] verifies that the augmentation operations play an important role in the late training phase and proposes to adopt the weight sharing strategy to improve the efficiency. In this paper, we take the above advanced AutoDA methods as the baselines.

2.1.2 Problems with existing methods

In this section, we first analyze the key components of an AutoDA method and then summarize the deficiencies in existing AutoDA methods. Generally, the policy search environment and policy search algorithm form the AutoDA method. The policy search environment simulates the physical policy application scenario, which is the foundation of an AutoDA method. Specifically, the policy search space definition and policy evaluation method are the core of the policy search environment. In terms of policy search space definition, most of the existing AutoDA methods search the policies in discrete policy space, such as AutoAugment, PBA, GAutoAugment and AWS. However, considering that the DA operation parameters themselves vary in a continuous space, searching the policies in a discrete space may be insufficient and lead to a sub-optimal solution.

For the policy evaluation method, all the existing methods evaluate individual policies independently. During the policy search process, they train a model based on data augmented with a single policy and take the test accuracy as the policy performance to guide policy search. However, the goal of AutoDA is to select a set of policies that can be used jointly to increase data diversity. As a result, it is conceivable that evaluating policy separately is not reasonable enough. Especially for FastAA, its evaluation objective is essentially finding policies that can transmit the distribution of augmented data close to training data. The motivation does not contribute directly to the target task, accounting for a restrictedly precise evaluation of policy performance.

In addition to the reliable policy search environment, policy search efficiency is also an essential part of AutoDA methods. AutoAugment adopts RNN as the policy selector and takes RL as the search algorithm, inevitably resulting in an unacceptable computational cost. As reported in the paper, AutoAugment requires completely training Wide-ResNet-40-2 15,000 times, which costs approximately 5,000 GPU hours for an NVIDIA Tesla P100 on the CIFAR-10 dataset. PBA incorporates random sampling to select the policy, pursuing efficiency at the cost of target task performance. In addition, RandAA regards the DA operation parameters as hyperparameters and researches them once a new target model is trained, which leads to a considerable extra computational cost, especially when the model is complex.

In summary, there are three problems in existing AutoDA methods:

1. The discrete policy search space is not precise enough, which may harm policy search performance.
2. Evaluating individual policy independently is not reasonable since policies perform jointly in the physical application.
3. The balance between policy performance and search overhead.

2.2 Bayesian optimization

Bayesian Optimization (BO) [35, 36] is an effective global optimization algorithm that has widely facilitated optimizing tasks. It is quite suitable to solve complex optimization problems where the objective functions are implicit or non-convex, multimodal, and expensive to evaluate. By designing the appropriate probabilistic surrogate model and the acquisition function, the BO framework can obtain the optimal solution with a few evaluations. As an effective means to optimize complex black-box problems, BO has been applied successfully in many fields, including experimental design [37], time series forecasting [38], and automatic algorithm configuration [36, 39].

BO fits the real objective function with a probabilistic surrogate model and can effectively utilize historical evaluation information to improve search efficiency. Particularly, it is an iterative process consisting of three steps. First, BO selects the most potential evaluation point from the search space by maximizing the acquisition function. Then, BO evaluates the selected evaluation point according to an evaluation model. Finally, the newly obtained observation is added to the historical observation set, and the probabilistic surrogate model is updated to prepare for the next iteration. In this paper, we employ BO as the policy search algorithm to find policies efficiently.

3 The proposed BO-Aug method

For clarity, we first formulate the general procedure of AutoDA. Then we present the proposed method, **B**ayesian **O**ptimization **Aug**mentation (**BO-Aug**).

3.1 Formulation of autoDA

Given a dataset $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^n$, AutoDA aims to find a candidate set of k optimal DA policies $\mathbb{S} = \{f_i\}_{i=1}^k$, where each of f_i denotes a selected DA policy. Through the DA policy set $\mathbb{S}$, we gain an augmented dataset $\widehat{\mathcal{D}} = \{f(\mathbf{x}_i)\}_{i=1}^n$, $f \in \mathbb{S}$, which can be used to train more effective models for certain target tasks.

Broadly speaking, the general process of AutoDA can be formulated as an iterative search process from a predefined DA policy space $\mathcal{F}$, $\mathbb{S} \subseteq \mathcal{F}$. Each iteration t consists of three key steps. First, the policy selector $P_\theta(\mathcal{F})$, parameterized by θ, finds a new DA policy from $\mathcal{F}$ and updates $\mathbb{S}$. Second, the policy evaluator $E(\cdot)$ estimates the performance score y_t of f_t, *i.e.,* $y_t = E(f_t)$. Finally, the policy selector $P_\theta(\mathcal{F})$ is updated with the policy-evaluation pairs $\Omega = \{(f_i, y_i)\}_{i=1}^t$ by a certain training technique such as reinforcement learning. After multiple iterations, the top-k policies with the highest evaluation scores compose the optimal policy set.

As claimed earlier, BO-Aug aims to address three key issues in AutoDA. To highlight BO-Aug, we define a continuous policy search space, evaluate the policy more physically, and employ the Bayesian Optimization algorithm to search policy efficiently. Below, we introduce each component of BO-Aug in detail.

3.2 Policy search space definition

We define a continuous DA policy search space $\mathcal{F}$, where each DA policy $f \in \mathcal{F}$ consists of 2 basic DA operations. Referring to existing AutoDA methods [1–3, 32], we adopt 14 basic DA operations in total, taken from the Python image library [1]. Detailed descriptions of the operations are listed in Table 8 in Appendix A. Formally, each f is defined as a 5-dimensional vector $[opers, pro_1, m_1, pro_2, m_2]$. The first component $opers$ indicates the two DA operation types.

Specifically, $opers$ ranges from 0 to 196 (*i.e.,* 14×14). The mappings between value and operation types are listed in Table 9 in Appendix B. The second and fourth components pro_1 and pro_2 represent the application probability of two DA operations, respectively, which range from 0 to 1. The third and fifth components m_1 and m_2 are the corresponding application magnitude parameters of the DA operations. For convenience, we scale the magnitudes of all DA operations to [0, 9] during the search process and rescale them to the respective magnitudes in the physical application. To clearly describe the definition, we show a toy example as follows. [106.7859, 0.7195, 7.4252, 0.3982, 3.3927] represents that the policy consists of two operations **Contrast** and **Color**. The application probabilities of the two operations are 0.7195 and 0.3982, respectively, and 7.4252, as well as 3.3927, are the application magnitude parameters of the two operations, respectively.

[1] https://pillow.readthedocs.io/en/stable/

3.3 Policy evaluation method

The policy evaluation score measures policy performance and is essential to guide the policy search process. To design an instructive evaluation method, there is a key issue to be considered: how can the policy be evaluated as accurately as possible? An intuitive idea is to evaluate policy in the same way as its physical application. However, there are two main limitations in practice. First, training a target neural network on the complete dataset usually requires a huge time overhead, so it is impractical to evaluate the policy based on the test performance of the target model trained on the complete dataset. In addition, the goal is to find a set of policies to perform as a combination, while searching the complete policy set increases the search space and difficulty significantly. Existing works search one policy each time and integrate multiple optimal policies as the final set [1–3]. We present a novel policy evaluation method to overcome the above two limitations.

For the first limitation, we evaluate policy by training a smaller proxy model M based on a reduced dataset $\mathcal{D}_{red}$, thereby reducing the evaluation overhead while maintaining the relative accuracy of the evaluation. Specifically, we regard the error rate on a validation set $\mathcal{D}_{val}$ of the proxy model trained on the augmented reduced dataset $\mathcal{D}_{red}$ as the policy performance score. The evaluation process is formalized as,

$$E(f) = Err(f(\mathcal{D}_{red}), \mathcal{D}_{val}, M)) \tag{1}$$

For the second limitation, we present the policy group evaluation method. Specifically, we search for a policy each time and regard the performance of a policy group consisting of current and all previous policies as the estimation. The basic motivation is to evaluate policies as similar as possible to how they are ultimately applied without increasing the search space. Evaluating a policy group is closer to the physical application than an individual policy. Formally, the performance of a policy f_t selected in iteration t is evaluated as follows,

$$y_t = E(\{f_i\}_{i=1,\ldots,t}) \tag{2}$$

Based on this evaluation method, our search objective is to find the policy that could be added to a base policy group and provide the greatest performance boost.

3.4 Policy selector and update method

The combination of a selector and an update method could be regarded as an optimization algorithm. It selects samples from the policy search space and updates itself constantly to approximate the optimization objective. Here we employ **B**ayesian **O**ptimization (BO) as the optimization algorithm. The key idea of BO is to train a probabilistic surrogate model to approximate the implicit objective by iteratively correcting the hypothetical prior based on the increasing supervised information. In the context of AutoDA, the objective function is the mapping from DA policy to policy performance.

In this paper, we adopt the **G**aussian **P**rocess (GP) [40] as the surrogate model, which is a commonly used non-parametric model.

A GP is parameterized by a mean function $m : \mathcal{F} \to \mathbb{R}$ and a semi-positive definite kernel function $k : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$. To model the distribution over the objective function, we first specify the prior of GP, which characterizes the properties of the objective function. Then, we update the prior in light of the current policy-evaluation pair set Ω and compute the posterior. The posterior then could be used to select a new potential policy for evaluation to enlarge the policy-evaluation pair set Ω. The technique to select a new policy according to the posterior is called the acquisition function in the context of BO, which will be discussed later on.

Formally, we define the objective function as:

$$\varphi(f) \sim \mathcal{G}P(m(f), k(f, f')) \tag{3}$$

$\varphi(f)$ is distributed as a GP with mean function m and kernel function k. To maintain generality, the mean function is assumed to be $m(f) = 0$. For kernel function, we adopt the Matern kernel function [40] that enjoys high flexibility. Specifically, we choose the smooth parameter of it to be $5/2$. The kernel function is shown as,

$$k_{Matérn-\frac{5}{2}(r)}(f, f') = (1 + \frac{\sqrt{5}|f - f'|}{l} + \frac{5|f-f'|^2}{3l^2}) \exp(-\frac{\sqrt{5}|f-f'|}{l}) \tag{4}$$

where l is the scale parameter, a hyper parameters of GP. According to existing training data $\Omega = \{(f_i, y_i)\}_{i=1}^{t}$, we can compute a regular prior Gaussian distribution,

$$\varphi \sim \mathcal{N}(\mu, \Sigma) \tag{5}$$

where $\mu_i = m(f_i)$ and $\Sigma_{i,j} = k(f_i, f_j)$ for $i, j = 1, \ldots, t$. For unknown data f_*, we can arrive at its posterior predictive distribution as,

$$\begin{aligned} &\varphi_*|\varphi \sim \mathcal{N}(\mu^t, \Sigma^t), \\ &\mu^t = \mu_* + \Sigma_*^T \Sigma^{-1}(\varphi - \mu) \\ &\Sigma^t = \Sigma_{**} - \Sigma_*^T \Sigma^{-1} \Sigma_* \end{aligned} \tag{6}$$

where μ_* is the unknown data mean, Σ_* is the training-unknown data covariance and Σ_{**} is the unknown data covariance.

Based on the unknown data's posterior predictive distribution $\varphi_*(f) \sim \mathcal{N}(\mu^t, \Sigma^t)$, we could select the new potential data using the acquisition function. Here we adopt the **E**xpected **I**mprovement (EI) function whose expression is shown as,

$$\begin{aligned} f_{t+1} &= \arg\max_{f \in \mathcal{F}} \mathbb{E}_{\varphi_*(f) \sim \mathcal{N}(\mu^t, \Sigma^t)}[\max(\varphi_*(f) - y_{\max}, 0)] \\ &= \arg\max_{f \in \mathcal{F}} (\mu^t - y_{\max}) \Phi(\frac{\mu^t - y_{\max}}{\Sigma^t}) + \Sigma^t \phi(\frac{\mu^t - y_{\max}}{\Sigma^t}) \end{aligned} \tag{7}$$

where $y_{\max} = \max_{i=1,\ldots,t} (y_i)$ is the current maximal function value, $\Phi(\cdot)$ and $\phi(\cdot)$ are the cumulative distribution function and probability density function of the standard Gaussian distribution respectively. Specifically, we optimize the EI acquisition function with the help of the auxiliary optimizer **C**ovariance **M**atrix **A**daptation **E**volution **S**trategies (CMA-ES). This is how the selector samples the new policy from the search space.

Once new policy f_{t+1} is selected, it is evaluated by the policy evaluation method to obtain y_{t+1}. As a result, the training set would be enlarged as $\Omega = \{(f_i, y_i)\}_{i=1,\cdots,t+1}$. Then, we update the prior Gaussian distribution φ according to the enlarged Ω and calculate the new posterior $\varphi_*|\varphi \sim \mathcal{N}(\mu^{t+1}, \Sigma^{t+1})$. This is how the update method works.

3.5 Summary of BO-Aug

With the above components, we have BO-, which we summarize in Algorithm (AutoDA). The goal of AutoDA is to find a policy set to enlarge the target dataset and increase its diversity. We perform BO-Aug multiple times independently, combining the policies with the best estimation in each execution into the policy set, that is, $k = 1$. As a result, the final policy set consists of policies that boost performance based on different base policy groups, increasing the policy set diversity further. In addition, running BO-Aug independently multiple times means that we could search for policies in parallel.

Algorithm 1 BO-Aug.

Input: iteration number T, policy space $\mathcal{F}$.
Output: top-k DA policies in the searched DA policy set $\mathbb{S}$.
Initialization: $\mathbb{S} = \emptyset$; policy-evaluation pair set $\Omega = \emptyset$; policy selector $P_{\mu,\Sigma}(\mathcal{F})$
for $t = 1, \ldots, T$ **do**
  # Select a promising policy
  $f_t = P_{\mu,\Sigma}(\mathcal{F})$ // (7)
  $\mathbb{S} = \mathbb{S} \cup \{f_t\}$
  # Evaluate the selected policy
  $y_t = E(\mathbb{S})$ // (2)
  $\Omega = \Omega \cup \{(f_t, y_t)\}$
  # Update the policy selector
  $Update(P_{\mu,\Sigma}(\mathcal{F}); \Omega)$ // (6)
Return top-k DA policies in $\mathbb{S}$

4 Experiments

In this section, we validate the efficacy and efficiency of BO-Aug on the image classification task. First, we introduce the datasets and baseline methods. Second, we describe the experimental setup, including the program running environment and hyperparameters. Third, we verify the effectiveness of BO-Aug by comparing it with state-of-the-art AutoDA methods in terms of performance and computational cost. Finally, some analyses are provided to help better understand the characteristics of BO-Aug.

4.1 Datasets and baseline methods

As claimed earlier, AutoAugment [1] is the pioneering work in the AutoDA community. The experimental setup has been followed by subsequent works, including four benchmark image datasets in AutoDA, *i.e.,* CIFAR-10 [41], CIFAR-100 [41], SVHN [42] and Imagenet [43]. To make a fair comparison, we also verify the model's performance based on them in this paper. The CIFAR-10 dataset [41] consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class. The CIFAR-100 dataset [41] is similar to the CIFAR-10, except that it has 100 classes, each containing 600 images. The Street View House Numbers (SVHN) dataset [42] contains 99,289 images which are all of the small cropped digits whose size is 32x32. There are 73,257 images for training and 26,032 images for testing, and 531,131 additional images are often used as complementary training data. For the Imagenet dataset, we choose the prevalent subset of Imagenet, named TinyImagenet, for evaluation. In particular, it contains 110,000 64x64 images in 200 classes, with 550 images per class. The statistics of these datasets are presented in Table 2.

Table 2 Statistics of datasets

Dataset	Training set size	Test set size	Classes
CIFAR-10	50,000	10,000	10
CIFAR-100	50,000	10,000	100
SVHN	604,388	26,032	10
TinyImagenet	100,000	10,000	200

Following AutoAugment [1], we evaluate the policy performance during policy search based on a reduced dataset taken from the original dataset. Specifically, we randomly select 4,000 images and 1,000 images from the training set of CIFAR-10 and SVHN to form reduced CIFAR and reduced SVHN, respectively. For the TinyImagenet dataset, we randomly select 10 classes as the reduced TinyImagenet.

We mainly consider the Wide-ResNet [44] and Shake-Shake [45] types of models as the baseline models, which are the most advanced and widely used models on the image classification task. Specifically, we select Wide-ResNet-28-2, Wide-ResNet-28-10, Wide-ResNet-40-2, Shake-Shake (26 2x32d), Shake-Shake (26 2x96d), and Shake-Shake (262x112d) as the baselines without data augmentation. In addition, we also choose Cutout [20], AutoAugment [1], PBA [2], Fast AA [3], RandAugment [32], GAutoAugment [33] and AWS [34] as the baselines with data augmentation. Specifically, given a set of searched policies, the baseline model selects a policy from the policy set randomly to augment each image in each epoch. All the baseline methods are summarized in Table 3.

4.2 Experimental setup

Our experiments are performed on Ubuntu16.04, 4 GeForce GTX 1080 Ti GPU, using the TensorFlow (TF)[2] deep learning framework. There are two parts of hyperparameters in the experiment: hyperparameters during policy search and hyperparameters of baselines for final policy performance verification. For policy search, we perform BO-Aug 8 times independently and select 24 policies in total. We set the iterations for each run as $T = 100$. In each iteration, we adopt Wide-ResNet-40-2 (WRN-40-2) as the proxy model for policy evaluation. It is worth mentioning that, the total policy number and the proxy model are set referring to AutoAugment for a fair comparison. For baseline models, we search two hyperparameters, the learning rate, and weight decay, based on the best validation set accuracy. The remaining hyperparameters (*i.e.,* the type of optimizer, the size of mini-batch, and the training epochs) were kept consistent with

[2] https://tensorflow.google.cn/

Table 3 Summary of all the baseline methods

Baseline Methods	Descriptions
Wide-ResNet-28-10 [44]	
Shake-Shake (26 2x32d) [45]	
Shake-Shake (26 2x96d) [45]	Baseline image classification models without DA policies
Shake-Shake (26 2x112d) [45]	
Cutout [20]	Baseline image classification models with Cutout data preprocess operation
AutoAugment [1]	
PBA [2]	
Fast AA [3]	
RandAugment [32]	Baseline image classification models with DA policies
GAutoAugment [33]	
AWS [34]	

those reported in the original paper [44, 45]. The specific hyperparameter configurations are shown in Table 10 in Appendix C.

For the baseline GAutoAugment, we run the public code to obtain the experimental results on all architectures on all datasets. For the TinyImagenet dataset, we apply the available policies found on the complete Imagenet dataset of the original paper for evaluation; an exception is PBA, which does not conduct experiments on the Imagenet dataset. In addition, we omit the results of AWS on SVHN and TinyImagenet since there are no official codes or publicly available policies of SVHN and Imagenet. Other experimental results of baselines are taken directly from the original paper [1–3, 32, 34]

4.3 Comparison with the state-of-the-arts

4.3.1 Results on four datasets

In this section, we show the empirical results on reduced CIFAR, CIFAR-10, CIFAR-100, reduced SVHN, SVHN and TinyImagenet. Specifically, we performed BO-Aug based on reduced CIFAR, reduced SVHN and reduced TinyImagenet to search policies, and the found policies are shown in Tables 11 in Appendix D, 12 in Appendix E and 13 in Appendix F, respectively. It is noteworthy that the policies searched on reduced CIFAR would be used for reduced CIFAR, CIFAR-10, and CIFAR-100, the policies searched on reduced SVHN would be used for both reduced SVHN and SVHN.

We present the classification error rate in Table 4. Based on the results, we can make three observations:

- Most of the baselines with DA obtain significantly better performance than baseline, which indicates that DA plays a key role in improving model performance. An exception is GAutoAugment, which only benefits the baseline in some cases. The possible reason is that its evaluation of policy is inappropriate for baselines with high capacity. Furthermore, the performance gain from the single DA operation *i.e.,* Cutout, is limited, which illustrates the need for applying multiple DA policies.
- BO-Aug achieves consistent advancement on various datasets and baseline models. In particular, BO-Aug achieves an approximately 1% lower error rate over all base models on CIFAR-10, 2% on CIFAR-100 and 0.3% on SVHN, which are consistently advanced performances. Compared with other AutoDA methods, our methods generally obtain better performance than PBA, Fast AA, RandAugment and GAutoAugment. In particular, BO-Aug also achieves state-of-the-art performance on the TinyImagenet dataset.
- The policies searched based on the reduced dataset benefit itself most and enjoy good transferability. The searched policies could be well transferred across different datasets and model architectures. For example, policies searched on the reduced CIFAR-10 boost all models significantly on CIFAR-10 and CIFAR-100. From the results on reduced datasets, the largest error rate reduction is 7%, achieved by AA and BO-Aug on Shake-Shake (26 2x96d).

4.3.2 Computational cost

We further compare the computational cost of all AutoDA methods. In particular, we report the actual GPU hours on the CIFAR-10 dataset and the implemented GPU models to provide a thorough comparison. Particularly, for AutoAugment and FastAA, we take the reported GPU hours of the baseline from its original paper directly. For PBA and AWS, we report the approximated GPU hours by the description in the original paper. For Greedy AutoAugment

Table 4 Test set classification error rates (%) on reduced CIFAR-10 (R-C), CIFAR-10 (C-10), CIFAR-100 (C-100), reduced SVHN (R-S), SVHN (S) and TinyImagenet (Tiny)

Dataset	Model	Baseline	Cutout	AA	PBA	Fast AA	RandA	GAA	AWS	BO-Aug
R-C	WRN-28-10	18.84	16.50	14.13	12.82†	–	–	22.94	–	13.36‡
	S-S (26 2x96d)	17.05	13.40	10.04†	10.64	–	–	20.54	–	10.26‡
C-10	WRN-28-10	3.87	3.08	2.68	2.58†	2.70	2.70	5.11	1.95†	2.58‡
	S-S (26 2x32d)	3.55	3.02	2.47‡	2.54	2.70	–	2.89	–	2.43†
	S-S (26 2x96d)	2.86	2.56	1.99	2.03	2.00	2.00	2.88	1.65†	1.98‡
	S-S (26 2x112d)	2.82	2.57	1.89†	2.03	2.00‡	–	2.89	–	1.89†
C-100	WRN-28-10	18.80	18.41	17.09	16.73	17.30	16.70	22.98	15.28†	16.58‡
	S-S (26 2x96d)	17.05	16.00	14.28‡	15.31	14.90	–	27.08	14.07†	15.20
R-S	WRN-28-10	13.21	32.5	8.15‡	7.83†	–	–	18.58	–	8.19
	S-S (26 2x96d)	12.32	24.22	5.92†	6.46‡	–	–	9.71	–	7.21
S	WRN-28-10	1.50	1.30	1.07‡	1.18	1.10	1.00†	1.43	–	1.17
	S-S (26 2x96d)	1.40	1.20	1.02†	1.13‡	–	–	1.38	–	1.15
Tiny	WRN-28-2	41.33	39.05	38.07‡	–	38.37	38.67	41.58	–	38.02†
	WRN-40-2	40.10	37.76	36.96‡	–	37.07	37.79	39.99	–	36.84†

Lower is better. We abbreviate AutoAugment as AA, RandAugment as RandA and GAutoAugment as GAA, respectively. The results of baseline methods are directly borrowed from the original paper [1–3, 32]. † and ‡ indicates the best result and second best result, respectively

* Dash indicates that results are not available

and RandAugment, we report the cost by running its public code on our computer.

As shown in Table 5, our method is approximately 140 times faster than AutoAugment for the GPU hours. Additionally, it is worth noting that our method is implemented with GTX 1080 Ti while AutoAugment with Tesla P100. AutoAugment and AWS are the slowest methods; they both employ reinforcement learning to search augment policy, which is known for high computational cost. While AWS is 40 times faster than AutoAugment since it utilizes the shared weight for the proxy task in the policy evaluation process which improves the search efficiency. RandAugment describes the reduced DA operation search space with two hyper-parameters and utilizes the grid search to find out the optimal hyper-parameters, which removes the proxy task for policy evaluation and is more efficient than AutoAugment. However, it is necessary to search for the optimal hyper-parameters for different image classification models which leads to a heavy time complexity especially for large models. PBA, Fast AA, GAutoAugment and BO-Aug are the fastest methods. Although our method is slightly lower than PBA, Fast AA and GAutoAugment, we can obtain a better performance than them in most cases, as shown in Table 4. Overall, our method strikes a good balance between efficiency and performance, which is a promising new AutoDA method.

4.4 Model analyses

In this subsection, we provide some analytical experiments to help better understand the characteristics of BO-Aug.

4.4.1 Ablation study

To better verify the validity of BO-Aug, we perform an extensive ablation study of its two key gradients: continuous policy search space and policy group evaluation.

Discrete policy search space v.s. continuous policy search space.

As claimed earlier, we propose to search policies in the continuous policy space to depict policy more precisely. To verify this intuition, we perform BO-Aug in the discrete policy space and keep the other experimental setups unchanged. Experimental results are shown in Table 6.

Table 5 GPU hours of AutoDA methods on CIFAR-10. We abbreviate AutoAugment as AA, RandAugment as RandA and GAutoAugment as GAA, respectively

Methods	AA	PBA	Fast AA	RandA	GAA	AWS	BO-Aug
GPU hours	5000	10	3.5	98	25	125	36
GPU device	Tesla P100	Tesla P100	Tesla V100	GTX 1080Ti	GTX 1080Ti	Tesla P100	GTX 1080Ti

Table 6 Test error rates (%) based on searched policies from discrete policy space and continuous search space. Lower is better and † indicates the best result

Dataset	Model	Baseline	Discrete policy space	Continuous policy space
CIFAR-10	WRN-28-10	3.87	2.71	2.58†
	S-S (26 2x32d)	3.55	2.54	2.43†
	S-S (26 2x96d)	2.86	2.13	1.98†
	S-S (26 2x112d)	2.82	2.22	1.89†
CIFAR-100	WRN-28-10	18.80	17.20	16.58†
	S-S (26 2x96d)	17.05	16.17	15.20†
SVHN	WRN-28-10	1.50	1.21	1.17†
	S-S (26 2x96d)	1.40	1.22	1.15†

From the results, both results using DA policies from discrete or continuous space are better than baselines. Among them, we can see that performing policy search in the continuous space consistently achieves better performance than in the discrete space, which demonstrates the necessity of maintaining the precision of policies.

Single policy evaluation v.s. policy group evaluation

To ensure the reasonability of policy evaluation, we propose to perform group evaluation during the policy search process. The intuition behind this is to simulate the physical policy application without increasing the policy search space. In this section, we validate the effectiveness of group evaluation by comparing it with the single policy evaluation. Specifically, we augment the reduced dataset with a single policy and take the validation accuracy as the evaluation score, that is, we replace (2) with $y_t = E(\{f_t\})$. Experimental results are shown in Table 7.

As always, both DA results are better than baselines. Searching for policy via group evaluation continuously performs better than the single policy evaluation. This result is in line with our expectation that the evaluation method approaching physical application more prompts AutoDA methods to search for better policies.

4.4.2 Effect of the number of policies

As claimed in AutoAugment, increasing the number of policies used in the target task can increase the data diversity, thus improving the generalization of neural networks. To explore the effect of the number of policies on the target task performance, we investigate the test error of Wide-ResNet-28-10 on CIFAR-10 as a function of the different number of policies used in training. We construct 16 policy sets with different sizes from a pool of 48 good policies searched by BO-Aug. To be specific, we take 3 as the interval of set size. Each set is composed of the former size set and another 3 policies selected randomly from the policy pool. Figure 2 shows the relation between the test error and the number of policies used in model training. From the results we can see that, when the number of

Table 7 Test error rates (%) based on searched policies via single policy evaluation and policy group evaluation

Dataset	Model	Baseline	Single policy evaluation	Group evaluation
CIFAR-10	WRN-28-10	3.87	2.76	2.58†
	S-S (26 2x32d)	3.55	2.54	2.43†
	S-S (26 2x96d)	2.86	2.06	1.98†
	S-S (26 2x112d)	2.82	2.17	1.89†
CIFAR-100	WRN-28-10	18.80	17.10	16.58†
	S-S (26 2x96d)	17.05	15.33	15.20†
SVHN	WRN-28-10	1.50	1.23	1.17†
	S-S (26 2x96d)	1.40	1.18	1.15†

Lower is better and † indicates the best result

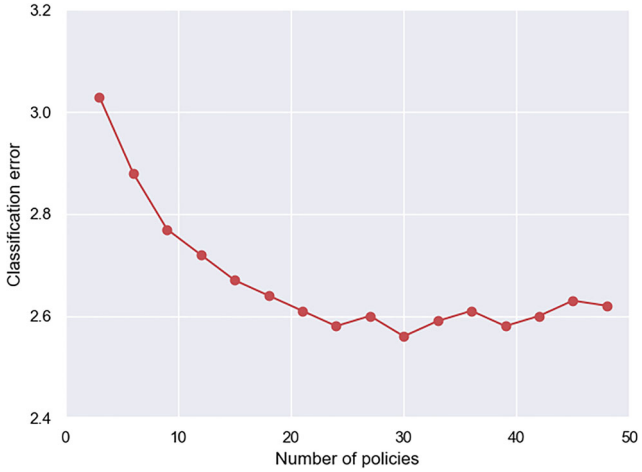


Fig. 2 Classification error (%) of Wide-ResNet-28-10 trained on CIFAR-10 as number of policies used in training

policies is small, using more policies can help improve the model performance. When the number of policies reaches approximately 20, increasing the number of policies will not bring significant performance improvement.

4.4.3 Visualization of the augmented images

In this subsection, we visualize some image examples augmented with the searched policies. During model training, each image will be augmented by a policy randomly selected from the searched policy set in each epoch. As a result, one image can be augmented with different policies during model training. For the same policy, the augmentation could be different depending on the operation application probability. We highlight the stochasticity of policy application which increases the dataset diversity significantly. We take an image of SVHN as an example to show the augmentation in the different epochs of model training, as shown in Fig. 3.

5 Conclusion

In this paper, we formalize the AutoDA process and summarize three key issues worth exploring in the research community, that is, the continuity of the policy search space, the policy evaluation method, and the efficiency of the search algorithm. To solve these issues, we propose a novel AutoDA method BO-Aug. Concretely, we define a continuous policy space for AutoDA and propose the policy group evaluation method to simulate the physical policy application without increasing the search space. To reduce the computational overhead, we employ BO as the search algorithm. Empirical results on the image classification task verify the proposed model's efficacy and efficiency. Compared with the advanced AutoDA methods, BO-Aug achieves a state-of-the-art or comparable performance. Besides, compared with Autoaugment, the most representative AutoDA method, the computational cost of BO-Aug is approximately 140 times faster, and the searched policies based on specific datasets can be transferred across different neural network architectures or even different datasets. In addition, model analyses are provided to help better understand the characteristics of the proposed method, including the ablation study, the effect of

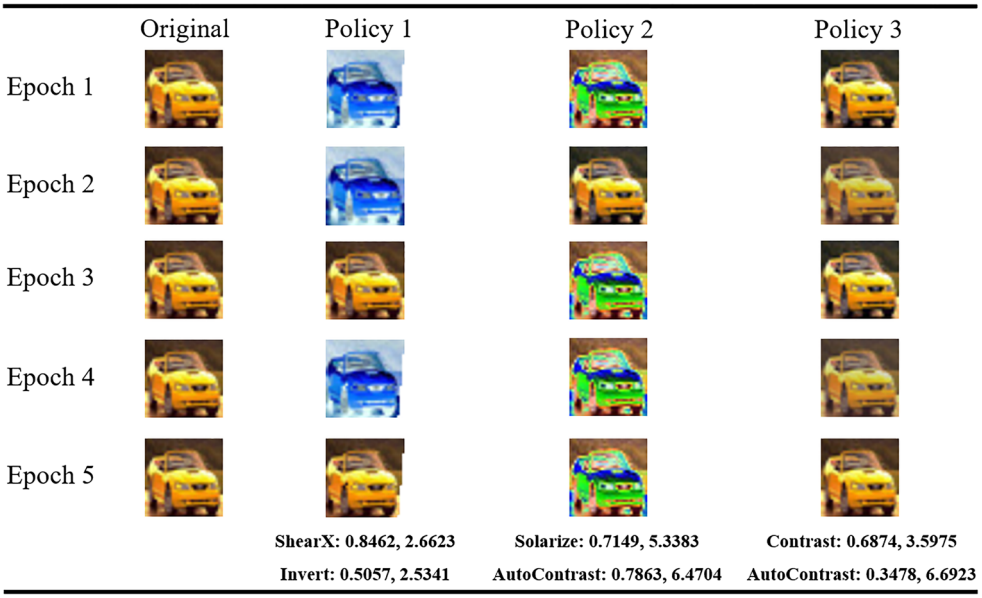


Fig. 3 Three policies found on CIFAR-10 and how an image is augmented in different epochs

the number of policies, and visualization of the augmented image.

Appendix A: Image transformation operations used for DA policies search

All available image transformation functions during the search process are listed below. These functions accept an image and corresponding operation parameters as input, and output a transformed image. The range of magnitudes for each operation is shown in the third column. Some transformations do not use magnitude information (*e.g.,* Invert and Equalize).

Table 8 Image transformation operations used for DA policies search

Number	Operation	Description	Magnitude Range
1	ShearX	Shear the image along the horizontal axis with rate *magnitude*.	[− 0.3, 0.3]
2	ShearY	Shear the image along the vertical axis with rate *magnitude*.	[− 0.3, 0.3]
3	TranslateX	Translate the image in the horizontal direction by *magnitude* number of pixels.	[− 150, 150]
4	TranslateY	Translate the image in the vertical direction by *magnitude* number of pixels.	[− 150, 150]
5	Rotate	Rotate the image *magnitude* degrees.	[− 30, 30]
6	AutoContrast	Maximize the image contrast, by making the darkest pixel black and lightest pixel white.	
7	Invert	Invert the pixels of the image.	
8	Equalize	Equalize the image histogram.	
9	Solarize	Invert all pixels above a threshold value of *magnitude*.	[0, 256]
10	Posterize	Reduce the number of bits for each pixel to *magnitude* bits.	[4, 8]
11	Contrast	Control the contrast of the image. A *magnitude=0* gives a gray image, whereas *magnitude*=1 gives the original image.	[0.1, 1.9]
12	Color	Adjust the color balance of the image, in a manner similar to the controls on a colour TV set. A *magnitude*=0 gives a black & white image, whereas *magnitude*=1 gives the original image.	[0.1, 1.9]
13	Brightness	Adjust the brightness of the image. A *magnitude*=0 gives a black image, whereas *magnitude*=1 gives the original image.	[0.1, 1.9]
14	Sharpness	Adjust the sharpness of the image. A *magnitude*=0 gives a blurred image, whereas *magnitude*=1 gives the original image.	[0.1, 1.9]

Appendix B: Mapping relationship between operation type value of policy vector and operation types

For each policy, we regard its first dimension as the two DA operation types. Considering that there are 14 DA

Table 9 Mapping relationship between operation type value of policy vector and operation types

Opers	Operation_1	Operation_2
[0, 1)	ShearX	
[1, 2)	ShearX	ShearY
[2, 3)	ShearX	TranslateX
[3, 4)	ShearX	TranslateY
[4, 5)	ShearX	Rotate
[5, 6)	ShearX	Solarize
[6, 7)	ShearX	Posterize
[7, 8)	ShearX	Contrast
[8, 9)	ShearX	Color
[9, 10)	ShearX	Brightness
[10, 11)	ShearX	Sharpness
[11, 12)	ShearX	AutoContrast
[12, 13)	ShearX	Invert
[13, 14)	ShearX	Equalize
⋮	⋮	⋮
[106, 107)	Contrast	Color
⋮	⋮	⋮
[195, 196]	Equalize	Equalize

operations in total, we use a number between 0 and 196 to represent two operations. Here we list the mapping relationship of the value and corresponding operation types.

Appendix C: Hyperparameter configuration in the experiments

Once BO-Aug finds the optimal DA policies, we will validate the policies' performance on different target models based on several datasets. Here we give the learning rate and weight decay values during target model training.

Table 10 Hyperparameter configuration in the experiments

Datasets	Model	Learning rate	Weight decay
CIFAR-10	WRN-28-10	0.1	5e-4
CIFAR-10	S-S(26 2x32d)	0.01	1e-3
CIFAR-10	S-S(26 2x96d)	0.02	1e-3
CIFAR-10	S-S(26 2x112d)	0.01	1e-3
CIFAR-100	WRN-28-10	0.1	5e-4
CIFAR-100	S-S(26 2x96d)	0.02	1e-3
Reduced CIFAR-10	WRN-28-10	0.02	1e-3
Reduced CIFAR-10	S-S(26 2x96d)	0.02	1e-3
SVHN	WRN-28-10	0.01	3e-2
SVHN	S-S(26 2x96d)	0.02	5e-5
Reduced SVHN	WRN-28-10	0.1	1e-4
Reduced SVHN	S-S(26 2x96d)	0.01	1e-2
TinyImagenet	WRN-28-2	0.1	5e-4
TinyImagenet	WRN-40-2	0.1	5e-4

Appendix D: Policies found on reduced CIFAR-10

Table 11 Policies found on reduced CIFAR-10

	Operation 1	Operation 2
Policy 1	(Contrast, 0.6874, 3.5975)	(AutoContrast, 0.3478, 6.6923)
Policy 2	(Brightness, 0.2875, 7.8932)	(AutoContrast, 0.9654, 4.9814)
Policy 3	(Solarize, 0.9930, 3.0359)	(Rotate, 0.6376, 3.0957)
Policy 4	(Posterize, 0.1359, 2.2361)	(Sharpness, 0.8544, 3.3977)
Policy 5	(TranslateY, 0.5520, 4.1526)	(Brightness, 0.0143, 3.7034)
Policy 6	(ShearX, 0.9463, 8.9832)	(TranslateX, 0.5740, 1.9317)
Policy 7	(Equalize, 0.3051, 6.1926)	(TranslateX, 0.6216, 0.0089)
Policy 8	(TranslateY, 0.4527, 4.3223)	(AutoContrast, 0.9646, 4.6787)
Policy 9	(ShearX, 0.8462, 2.6623)	(Invert, 0.5057, 2.5341)
Policy 10	(AutoContrast, 0.5957, 8.1685)	(ShearY, 0.7342, 6.5833)
Policy 11	(Color, 0.0895, 0.7037)	(Equalize, 0.5103, 2.3937)
Policy 12	(Invert, 0.1966, 7.3815)	(Solarize, 0.3622, 2.9833)
Policy 13	(ShearY, 0.5345, 5.0416)	(TranslateX, 0.9978, 7.4486)
Policy 14	(Invert, 0.1813, 8.1266)	(Posterize, 0.9963, 8.7576)
Policy 15	(Posterize, 0.5928, 8.1990)	(Brightness, 0.9625, 5.8913)
Policy 16	(Brightness, 0.5037, 3.2453)	(Brightness, 0.9853, 4.1967)
Policy 17	(TranslateY, 0.0772, 4.5035)	(TranslateY, 0.8807, 0.2262)
Policy 18	(TranslateX, 0.2531, 4.3762)	(ShearX, 0.3437, 1.7062)
Policy 19	(AutoContrast, 0.7115, 4.3842)	(AutoContrast, 0.1873, 7.0988)
Policy 20	(Equalize, 0.7708, 8.4010)	(Solarize, 0.2364, 1.3529)
Policy 21	(Solarize, 0.7149, 5.3383)	(AutoContrast, 0.7863, 6.4704)
Policy 22	(AutoContrast, 0.2122, 6.9200)	(TranslateY, 0.0081, 6.5256)
Policy 23	(Equalize, 0.1361, 6.0541)	(TranslateX, 0.9480, 3.6382)
Policy 24	(TranslateX, 0.1695, 5.5734)	(Brightness, 0.5921, 1.2058)

Appendix E: Policies found on reduced SVHN

Table 12 Policies found on reduced SVHN

	Operation 1	Operation 2
Policy 1	(ShearY, 0.7614, 1.9849)	(Brightness, 0.0097, 8.1271)
Policy 2	(Invert, 0.7468, 3.1175)	(ShearY, 0.6576, 4.8592)
Policy 3	(Brightness, 0.9610, 2.3202)	(Equalize, 0.9848, 2.5281)
Policy 4	(Contrast, 0.7195, 7.4252)	(Color, 0.3982, 3.3927)
Policy 5	(TranslateX, 0.3231, 2.1326)	(Solarize, 0.9329, 7.0658)
Policy 6	(Equalize, 0.8227, 3.3000)	(Rotate, 0.7638, 3.0728)
Policy 7	(Equalize, 0.9919, 7.0251)	(TranslateY, 0.6032, 4.9770)
Policy 8	(Solarize, 0.5896, 8.5091)	(Invert, 0.6628, 6.2895)
Policy 9	(Rotate, 1.0000, 7.4837)	(ShearY, 0.2407, 6.0313)
Policy 10	(Invert, 0.8723, 7.2289)	(Posterize, 0.3586, 7.1004)
Policy 11	(Color, 0.0072, 4.7249)	(Invert, 0.9959, 3.8010)
Policy 12	(Solarize, 0.2024, 3.0107)	(Color, 0.7671, 5.0435)
Policy 13	(Invert, 0.6668, 5.4347)	(Rotate, 0.4755, 4.1415)
Policy 14	(Contrast, 0.9712, 8.2252)	(Color, 0.8781, 8.2405)
Policy 15	(Brightness, 0.4671, 8.6339)	(AutoContrast, 0.6229, 5.1071)
Policy 16	(ShearY, 0.9499, 3.2718)	(Brightness, 0.4102, 3.7130)
Policy 17	(Solarize, 0.7952, 7.3589)	(ShearY, 0.5424, 3.7230)
Policy 18	(Contrast, 0.9796, 7.3553)	(Solarize, 0.0061, 2.4950)
Policy 19	(Posterize, 0.9873, 2.2049)	(Invert, 0.4173, 6.7201)
Policy 20	(Invert, 0.6843, 1.2672)	(ShearY, 0.6295, 2.2341)
Policy 21	(Contrast, 0.9862, 3.8489)	(Equalize, 0.7513, 3.1612)
Policy 22	(Color, 0.6640, 2.4163)	(Solarize, 0.3826, 8.3206)
Policy 23	(TranslateY, 0.0706, 6.1889)	(Contrast, 0.0941, 0.0007)
Policy 24	(Equalize, 0.3803, 4.7222)	(TranslateY, 0.5262, 7.7046)

Appendix F: Policies found on reduced TinyImagenet

Table 13 Policies found on reduced TinyImagenet

	Operation 1	Operation 2
Policy 1	(Solarize, 0.3624, 4.222)	(Rotate, 0.6893, 1.1141)
Policy 2	(Equalize, 0.9529, 1.1217)	(TranslateX, 0.9703, 5.9022)
Policy 3	(AutoContrast, 0.9774, 4.7498)	(Invert, 0.7025, 2.7050)
Policy 4	(Invert, 0.3736, 3.1854)	(TranslateX, 0.3630, 4.6174)
Policy 5	(Contrast, 0.4781, 3.7085)	(Contrast, 0.9649, 5.7581)
Policy 6	(Posterize, 0.8782, 4.8429)	(TranslateY, 0.9073, 4.5021)
Policy 7	(Sharpness, 0.9843, 5.5285)	(Contrast, 0.6449, 1.2622)
Policy 8	(AutoContrast, 0.9935, 1.1801)	(Contrast, 0.4658, 5.3732)
Policy 9	(Contrast, 0.0022, 6.4818)	(Contrast, 0.6869, 5.1583)
Policy 10	(Sharpness, 0.8777, 2.2342)	(Invert, 0.9656, 4.4988)
Policy 11	(AutoContrast, 0.2078, 3.6058)	(Contrast, 0.9311, 1.3419)
Policy 12	(ShearY, 0.1684, 0.0941)	(TranslateY, 0.7041, 2.5579)
Policy 13	(Brightness, 0.0850, 4.3849)	(Posterize, 0.6625, 0.5206)
Policy 14	(Equalize, 0.4561, 2.3181)	(Brightness, 0.5368, 0.7457)
Policy 15	(Invert, 0.1822, 0.5172)	(TranslateX, 0.6845, 6.0058)
Policy 16	(TranslateY, 0.1354, 7.7569)	(Equalize, 0.8440, 2.2455)
Policy 17	(Equalize, 0.8356, 3.5384)	(TranslateX, 0.3615, 3.7196)
Policy 18	(TranslateX, 0.8259, 2.3050)	(ShearX, 0.0882, 2.0450)
Policy 19	(Posterize, 0.3470, 6.0685)	(ShearY, 0.8997, 5.0567)
Policy 20	(Invert, 0.7118, 6.9019)	(ShearY, 0.4892, 3.8570)
Policy 21	(TranslateX, 0.2053, 5.6103)	(Brightness, 0.3963, 4.2215)
Policy 22	(TranslateX, 0.3773, 0.6692)	(AutoContrast, 0.5211, 8.9367)
Policy 23	(Invert, 0.3329, 8.8534)	(Solarize, 0.2331, 5.0473)
Policy 24	(TranslateY, 0.9585, 5.5466)	(Solarize, 0.4071, 8.6902)

Acknowledgements This work was supported by the National Key R&D Program of China under Grant Nos. 2021ZD0112501 and 2021ZD0112502; the National Natural Science Foundation of China under Grant Nos. 62172185 and 61876069; Jilin Province Key Scientific and Technological Research and Development Project under Grant Nos. 20180201067GX and 20180201044GX; Jilin Province Natural Science Foundation under Grant No. 20200201036JC; and China Postdoctoral Science Foundation funded project under Grant No.2021M701388.

References

1. Cubuk ED, Zoph B, Mane D, Vasudevan V, Le QV (2019) Autoaugment: learning augmentation policies from data. In: Proceedings of the 32nd IEEE conference on computer vision and pattern recognition, pp 113–123
2. Ho D, Liang E, Chen X, Stoica I, Abbeel P (2019) Population based augmentation: efficient learning of augmentation policy schedules. In: Proceedings of the 36th international conference on machine learning, pp 2731–2741
3. Lim S, Kim I, Kim T, Kim C, Kim S (2019) Fast autoaugment. In: Proceedings of the 33rd advances in neural information processing systems, pp 6665–6675
4. Takase T, Karakida R, Asoh H (2021) Self-paced data augmentation for training neural networks. Neurocomputing 442:296–306
5. Bandara K, Hewamalage H, Liu Y-H, Kang Y, Bergmeir C (2021) Improving the accuracy of global forecasting models using time series data augmentation. Pattern Recogn 120:108148
6. Wang Y, Wei X, Tang X, Shen H, Ding L (2020) Cnn tracking based on data augmentation. Knowl-Based Syst 194:105594
7. Leng Y, Zhao W, Lin C, Sun C, Wang R, Yuan Q, Li D (2020) Lda-based data augmentation algorithm for acoustic scene classification. Knowl-Based Syst 195:105600
8. Yun S, Han D, Oh SJ, Chun S, Choe J, Yoo Y (2019) Cutmix: regularization strategy to train strong classifiers with localizable features. In: Proceedings of the 17th international conference on computer vision, pp 6023–6032
9. Tolstikhin IO, Houlsby N, Kolesnikov A, Beyer L, Zhai X, Unterthiner T, Yung J, Steiner A, Keysers D, Uszkoreit J et al (2021) Mlp-mixer: an all-mlp architecture for vision. Adv Neural Inf Process Syst, vol 34
10. Sohn K, Berthelot D, Carlini N, Zhang Z, Zhang H, Raffel C, Cubuk ED, Kurakin A, Li C (2020) Fixmatch: simplifying semi-supervised learning with consistency and confidence. In: Proceedings of the 34th advances in neural information processing systems, pp 596–608
11. Zoph B, Ghiasi G, Lin T, Cui Y, Liu H, Cubuk ED, Le Q (2020) Rethinking pre-training and self-training. In: Proceedings of the 34th advances in neural information processing systems, pp 3833–3845
12. Xie Q, Dai Z, Hovy EH, Luong T, Le Q (2020) Unsupervised data augmentation for consistency training. In: Proceedings of the 34th advances in neural information processing systems, pp 6256–6268
13. Caron M, Misra I, Mairal J, Goyal P, Bojanowski P, Joulin A (2020) Unsupervised learning of visual features by contrasting cluster assignments. In: Proceedings of the 34th advances in neural information processing systems, pp 9912–9924
14. He K, Fan H, Wu Y, Xie S, Girshick RB (2020) Momentum contrast for unsupervised visual representation learning. In: Proceedings of the 34th advances in neural information processing systems, pp 9726–9735
15. Misra I, Maaten LVD (2020) Self-supervised learning of pretext-invariant representations. In: Proceedings of the 33rd IEEE conference on computer vision and pattern recognition, pp 6706–6716
16. Chen T, Kornblith S, Swersky K, Norouzi M, Hinton GE (2020) Big self-supervised models are strong semi-supervised learners. In: Proceedings of the 34th advances in neural information processing systems, pp 22243–22255
17. Chen T, Kornblith S, Norouzi M, Hinton GE (2020) A simple framework for contrastive learning of visual representations. In: Proceedings of the 37th international conference on machine learning, pp 1597–1607
18. Hu J, Shen L, Sun G (2018) Squeeze-and-excitation networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 7132–7141
19. Tan M, Le Q (2019) Efficientnet: rethinking model scaling for convolutional neural networks. In: International conference on machine learning, pp 6105–6114
20. DeVries T, Taylor GW (2017) Improved regularization of convolutional neural networks with cutout. arXiv:1708.04552
21. Zhong Z, Zheng L, Kang G, Li S, Yang Y (2020) Random erasing data augmentation. In: Proceedings of the AAAI conference on artificial intelligence, vol 34, pp 13001–13008
22. Zhang H, Cisse M, Dauphin YN, Lopez-paz D (2018) Mixup: beyond empirical risk minimization. In: International conference on learning representations

23. Chinbat V, Bae S-H (2022) Ga3n: generative adversarial autoaugment network. Pattern Recogn 127:108637
24. dos Santos Tanaka FHK, Aranha C (2019) Data augmentation using gans. Proc Mach Learn Res 1:16
25. Songyan Liu J-GH, Haiyun Guo Xu, Zhao EA (2020) A novel data augmentation scheme for pedestrian detection with attribute preserving gan. Neurocomputing 401:123–132
26. DeVries T, Taylor GW (2017) Dataset augmentation in feature space. In: Proceedings of the 34th international conference on machine learning, workshop track
27. Wang Y, Pan X, Song S, Zhang H, Huang G, Wu C (2019) Implicit semantic data augmentation for deep networks. Adv Neural Inf Process Syst, vol 32
28. Pham H, Guan M, Zoph B, Le Q, Dean J (2018) Efficient neural architecture search via parameters sharing. In: International conference on machine learning, pp 4095–4104
29. Ding Z, Chen Y, Li N, Zhao D, Sun Z, Chen CP (2021) Bnas: efficient neural architecture search using broad scalable architecture. IEEE Trans Neural Netw Learn Syst
30. Wei C, Niu C, Tang Y, Wang Y, Hu H, Liang J (2022) Npenas: neural predictor guided evolution for neural architecture search. IEEE Trans Neural Netw Learn Syst
31. Chen Y, Gao R, Liu F, Zhao D (2021) Modulenet: knowledge-inherited neural architecture search. IEEE Trans Cybern
32. Cubuk ED, Zoph B, Shlens J, Le QV (2020) Randaugment: practical automated data augmentation with a reduced search space. In: Proceedings of the 33rd IEEE conference on computer vision and pattern recognition, pp 3008–3017
33. Naghizadeh A, Abavisani M, Metaxas DN (2020) Greedy autoaugment. Pattern Recogn Lett 138:624–630
34. Tian K, Lin C, Sun M, Zhou L, Yan J, Ouyang W (2020) Improving auto-augment via augmentation-wise weight sharing. Adv Neural Inf Process Syst 33:19088–19098
35. Cui J, Yang B (2017) Survey on bayesian optimization methodology and applications. In: Journal of software, pp 176–198
36. Lindauer M, Eggensperger K, Feurer M, Biedenkapp A, Deng D, Benjamins C, Ruhkopf T, Sass R, Hutter F (2022) Smac3: a versatile bayesian optimization package for hyperparameter optimization. J Mach Learn Res 23(54):1–9
37. Imani M, Ghoreishi SF (2021) Graph-based bayesian optimization for large-scale objective-based experimental design. IEEE Trans Neural Netw Learn Syst
38. Du L, Gao R, Suganthan PN, Wang DZ (2022) Bayesian optimization based dynamic ensemble for time series forecasting. Inf Sci 591:155–175
39. Turner R, Eriksson D, McCourt M, Kiili J, Laaksonen E, Xu Z, Guyon I (2021) Bayesian optimization is superior to random search for machine learning hyperparameter tuning: analysis of the black-box optimization challenge 2020. In: NeurIPS 2020 competition and demonstration track, pp 3–26
40. Williams CK, Rasmussen CE (2006) Gaussian Processes for Machine Learning. MIT Press Cambridge, MA
41. Krizhevsky A, Hinton G (2009) Learning multiple layers of features from tiny images. Technical report, Citeseer
42. Netzer Y, Wang T, Coates A, Bissacco A, Wu B, Ng AY (2011) Reading digits in natural images with unsupervised feature learning. In: Proceedings of the 28th neural information processing systems workshop on deep learning and unsupervised feature learning
43. Deng J, Dong W, Socher R, Li L-J, Li K, Fei-Fei L (2009) Imagenet: a large-scale hierarchical image database. In: IEEE conference on computer vision and pattern recognition, pp 248–255
44. Zagoruyko S, Komodakis N (2016) Wide residual networks. Proceedings of the 27th British machine vision conference, pp 8701–8712
45. Gastaldi X (2017) Shake-shake regularization. arXiv:1705.07485

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Chunxu Zhang received the BSc degrees in computer science from Jilin University in 2017. She is now a PhD candidate in the College of Computer Science and Technology, Jilin University, China. Her main research interests are related to data augmentation, federated learning and graph representation learning.



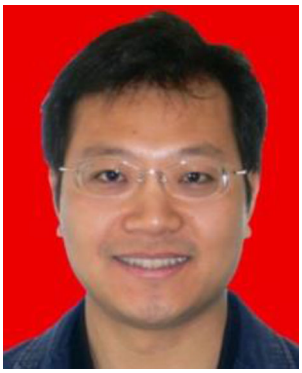
Ximing Li received the PhD degree from Jilin University, China, and currently an Associate Professor from Jilin University, China. His research interests include data mining, machine learning and natural language processing. He has published more than 60 papers at the competitive venues, including ICLR, ACL, IJCAI, AAAI, WWW, CIKM, etc.



Zijian Zhang received the master's degree from Jilin University in 2018, where he is currently pursuing the doctor's degree in computer applied technology with the College of Computer Science and Technology. His research interests include computer vision and urban computing



Jiaxu Cui received the B.Sc., M.S. and Ph.D. degrees from Jilin University, Changchun, China, in 2013, 2016, and 2021, respectively, where he is currently a postdoc in Jilin University. He has published over 10 high-quality papers in international conferences and journals including AAAI, TNNLS, ESWA, KSEM and ICC, and have served as reviewers in several major academic conferences and journals, including AAAI, IJCAI, TNNLS, IJMLC and TBD. His main research interests include complex network analysis and optimization, intelligent optimization and network representation learning. More information can be found at https://csjtx1021.github.io/.



Bo Yang is currently a professor at the College of Computer Science and Technology, Jilin University. He is also the director of the Key Laboratory of Symbolic Computation and Knowledge Engineering, Ministry of Education, China. His current research interests include the areas of data mining, complex network analysis, self-organized and self-adaptive multi-agent systems, with applications to knowledge engineering and intelligent health informatics.