Differentiable RandAugment: Learning Selecting Weights and Magnitude Distributions of Image Transformations

Anqi Xiao, Biluo Shen, Jie Tian, *Fellow, IEEE*, and Zhenhua Hu, *Senior Member, IEEE*

***Abstract*— Automatic data augmentation is a technique to automatically search for strategies for image transformations, which can improve the performance of different vision tasks. RandAugment (RA), one of the most widely used automatic data augmentations, achieves great success in different scales of models and datasets. However, RA randomly selects transformations with equivalent probabilities and applies a single magnitude for all transformations, which is suboptimal for different models and datasets. In this paper, we develop Differentiable RandAugment (DRA) to learn selecting weights and magnitudes of transformations for RA. The magnitude of each transformation is modeled following a normal distribution with both learnable mean and standard deviation. We also introduce the gradient of transformations to reduce the bias in gradient estimation and KL divergence as part of the loss to reduce the optimization gap. Experiments on CIFAR-10/100 and ImageNet demonstrate the efficiency and effectiveness of DRA. Searching for only 0.95 GPU hours on ImageNet, DRA can reach a Top-1 accuracy of 78.19% with ResNet-50, which outperforms RA by 0.28% under the same settings. Transfer learning on object detection also demonstrates the power of DRA. The proposed DRA is one of the few that surpasses RA on ImageNet and has great potential to be integrated into modern training pipelines to achieve state-of-the-art performance. Our code will be made publicly available for out-of-the-box use.**

***Index Terms*— Data augmentation, automated machine learning, differentiable optimization, random augmentation.**

Manuscript received 16 June 2022; revised 25 February 2023; accepted 21 March 2023. Date of publication 17 April 2023; date of current version 27 April 2023. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 92059207, Grant 62027901, Grant 81930053, and Grant 81227901; in part by the Chinese Academy of Sciences (CAS) Youth Interdisciplinary Team under Grant JCTD-2021-08; in part by the Zhuhai High-Level Health Personnel Team Project under Grant Zhuhai HLHPTP201703; and in part by the Cloud Tensor Processing Unit (TPUs) from Google's TPU Research Cloud (TRC). The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Giulia Fracastoro. *(Corresponding authors: Jie Tian; Zhenhua Hu.)*

Anqi Xiao, Biluo Shen, and Zhenhua Hu are with the CAS Key Laboratory of Molecular Imaging, Beijing Key Laboratory of Molecular Imaging, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, and also with the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 101408, China (e-mail: xiaoanqi2020@ia.ac.cn; shenbiluo2019@ia.ac.cn; zhenhua.hu@ia.ac.cn).

Jie Tian is with the CAS Key Laboratory of Molecular Imaging, Beijing Key Laboratory of Molecular Imaging, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, also with the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 101408, China, also with the Beijing Advanced Innovation Center for Big Data-Based Precision Medicine, School of Engineering Medicine, Beihang University, Beijing 100191, China, and also with the Engineering Research Center of Molecular and Neuro Imaging of Ministry of Education, School of Life Science and Technology, Xidian University, Xi'an 710071, China (e-mail: tian@ieee.org).

This article has supplementary downloadable material available at https://doi.org/10.1109/TIP.2023.3265266, provided by the authors.

Digital Object Identifier 10.1109/TIP.2023.3265266

I. Introduction

DATA augmentation, which mainly consists of geometric transformations (*rotate*, *translate*, etc.) and color transformations (*invert*, *contrast*, etc.), is a commonly used tool to generate additional data from the original. It increases diversity of the training dataset without collecting extra data. This technique is widely used in computer vision tasks to help the training of deep neural networks without severely changing high-level semantics in images. It can be seen as a regularization method to alleviate the over-fitting problem as well. Many synthetic data augmentation strategies [1], [2], [3], [4] have been designed and achieved great success in recent years. However, these designs usually require expert knowledge, large amounts of experimental trials and prior information to seek a proper configuration. Improper application or choice of augmentation even introduces outliers to the training data, which harms the final performance [3], [5], [6].

With the advances of automated machine learning (AutoML), automatically exploring data augmentation strategies directly from the dataset becomes popular. The process to explore an optimal augmentation strategy, including a set of parameters or rules, is called search. For example, AutoAugment [7] (AA) focuses on the learning of augmentations based on reinforcement learning (RL). Compared with traditional data augmentation, AA requires less expert knowledge and prior information to achieve impressive results by automatically searching for the amounts of transformations and their combinations. It is an offline search method that decouples the process for searching the augmentation strategies and training the target model. The searched strategies are called policy, which can also be easily transferred to new classification tasks for wider applications. However, the search cost of AA is extremely expensive, even on a proxy task that adopts a subset of the target dataset. As a result, the following works try to improve the search efficiency. Differentiable methods [8], [9], [10] appear and significantly reduce the search cost of policy learning to a few hours. Trends to the development of automatic data augmentation show the potential for differentiable methods. Nevertheless, these methods generally slightly

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

sacrifice performance, especially on complex datasets. Besides, current differentiable methods mainly focus on learning a set of fixed magnitudes for transformations, which limits the scale of the augmented image space and the upper bound for model performance.

Online automatic data augmentation methods use a different manner that learns augmentation strategies together with model training [5], [11], [12], [13], [14] to improve model performance. These methods avoid extra search overhead before training while expanding the augmented image space. However, the frameworks of online augmentation works are complex, and the search overhead remains high. Although search methods combined with meta-learning [15] and bi-level optimization [16] reduce the time for online search, these methods suffer from obvious performance decrements that lose one of the most important benefits of online learning. Besides, the augmentation policy of the online search methods can hardly be transferred to different tasks. These shortages limit the wide application of online augmentation methods.

Recent works, such as RandAugment [5] (RA), adopt randomness to improve performance for wide applications. RA uniformly samples combinations of transformations in the search space to augment images. The simple design achieves unexpected impressive results. Due to the simplicity and effectiveness, RA has been integrated into other works (e.g., DeiT [17], Swin Transformer [18]) as an augmented training strategy. However, to achieve optimal performance, RA requires an offline grid search on the whole training dataset to find the proper policy parameters, which is time-consuming even with a dramatically reduced search space. Although RA has the capability to achieve satisfactory performance with manually selected parameters to avoid the search, the simple search space limits the upper bound of the augmented image space. UniformAugment [14] and TrivialAugment [13] are two other methods that benefit from randomness with no search process, while they are not flexible enough for different target models and tasks that rely on inconsistent types and magnitudes of transformations.

In this work, we take the advantages of differentiable augmentation and effectiveness of random factors, and develop Differentiable RandAugment (DRA), an offline automatic augmentation method that can effectively learn policy parameters, including selecting weights and magnitude distributions of transformations, with a small search cost to expand the augmented image space of RA. Our DRA treats each augmentation transformation as a module in the model, with a weight indicating the probability in forward sampling and a magnitude following a learnable normal distribution to control the deformation. Besides, we introduce the gradient of transformations to reduce the bias of gradient estimation during the search. We also revisit the optimization objective of the differentiable augmentation search and find the inconsistency between updating model parameters and policy parameters. To reduce the gap, we propose a loss function with KL divergence, which measures the similarity of augmented and original images after model inference. Experiments on CIFAR-10/100 [19] and ImageNet [20] show that our DRA achieves better accuracy compared with some mainstream methods within a short search time. Transfer learning on object detection using COCO [21] further demonstrates the effectiveness of DRA. We emphasize that our DRA outperforms RA by 0.28% under the same settings, with only 0.95 GPU hours overhead on a single Tesla P100 on ImageNet using ResNet-50. Compared with prior works, it has a flexible design to be adaptive to different tasks, with a small search cost to improve performance of the offline automatic data augmentation. The pipeline of DRA is shown in Fig. 1.

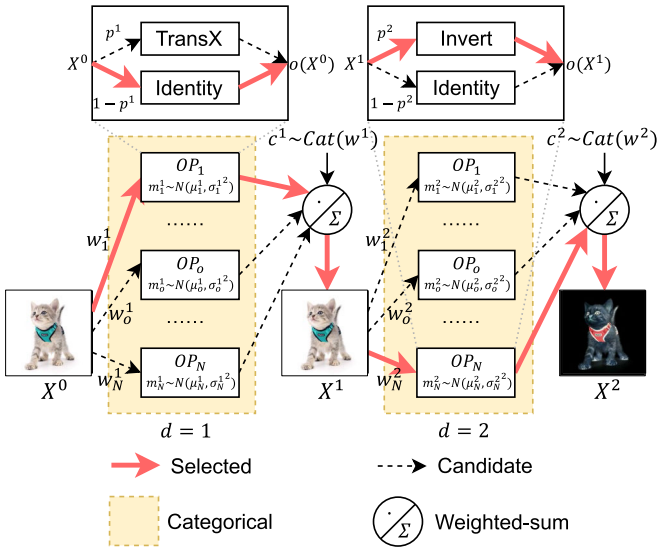

Fig. 1. A simplified example of the pipeline of Differentiable RandAugment in the forward. One yellow box represents an augmentation layer, which samples an operator from categorical distribution according to the weights of operators w. Each operator inside an augmentation layer owns a learnable normal distribution of magnitude. The sampled operator has probability p^d to be applied in the d^{th} layer, where $p^d \sim U\left(p_{min\_t}, p_{max\_t}\right)$ is randomly sampled before each iteration. The output of each augmentation layer is a weighted sum of the calculation results of all operators, where the weights are the sampled results of categorical distribution. *Cat* is short for categorical distribution.

The contributions of our work can be summarized as follows:

1) We propose a differentiable automatic data augmentation method DRA that models the magnitudes of transformations following a learnable normal distribution, which achieves better performance on classification and object detection compared with RA. The search overhead is also reduced to only 0.95 GPU hours on ImageNet with a single Tesla P100.
2) DRA adapts the search space based on RA and applies operator-sharing strategies to reduce both the search cost and difficulty, which is different from many previous methods that are based on the search space of AA.
3) We revisit the inconsistency between updating model parameters and policy parameters in policy learning, and introduce KL divergence as a loss item in the outer loop of bi-level optimization to reduce the optimization gap.

II. Related Works

A. Data Augmentation

Data augmentation (DA) has been widely used in computer vision tasks to improve the robustness of models, especially in image classification. The most widely used augmentations

include cropping, flipping, and resizing, which usually couple random factors to transform images. In recent years, several novel augmentation methods [1], [2], [3], [4], [6], [22], [23], [24] are designed according to expert or domain knowledge, and improve the performance and robustness of models. Apart from supervised training, DA has also been applied in other areas, such as contrastive learning [25], [26], [27] and reinforcement learning [28]. Although these augmentation methods achieve great success in many tasks, they require careful design with painstaking labor and large amounts of trials. Improper usage of augmentation shows no effect or even hurts performance. For example, Cutout [1] significantly decreased performance on reduced SVHN [29] as reported by Cubuk et al. [7]. As a result, the automatic design of data augmentation might be more suitable for different tasks.

B. Automatic Data Augmentation

Since Google proposed neural architecture search [30] (NAS), learning architectures and hyperparameters of deep neural networks automatically became popular. Inspired by NAS, automatic data augmentation appears and achieves great success in computer vision tasks. Current mainstream automatic data augmentation methods can mainly be divided into two types: offline [7], [8], [9], [10], [31], [32] and online [5], [11], [12], [13], [14], [33], [34].

The offline methods attempt to search for proper combinations of different image transformations, namely policy. Many offline data augmentation methods search the optimal policy on a proxy task to reduce the huge calculation cost, assuming that the policy found on the proxy performs as well as the one found on the whole dataset and target model. For instance, AutoAugment [7] (AA) is proposed based on reinforcement learning to automatically find a policy for optimal data augmentation. The policy contains a series of sub-policies, with a list of sequentially applied operators containing the name, applying probability, and a level indicating the magnitude of the operator. It achieves excellent performance on image classification in several datasets. Nevertheless, the search time is too expensive to be widely applied to different datasets. Although the learned policy can be transferred to other tasks, it is not as good as the one directly searched on the target task. Some algorithms are proposed afterward to speed up the search procedure. PBA [32] proposes to search for an augmentation schedule instead of a fixed policy. Fast AutoAugment [31] (Fast AA) avoids the trials on each policy to accelerate the search procedure. Although their search cost is severely reduced compared with AA, it is still expensive for the wide applications of these methods.

Recently, inspired by DARTS [35], differentiable methods for automatic data augmentation appear and show great efficiency in policy search. Faster AutoAugment [10] (Faster AA) regards data augmentation as filling the missing points in the training dataset. DADA [8] uses a GDAS-like [36] sampling strategy in the forward, and applies the RELAX estimator [37] to estimate unbiased gradients of policy parameters in the backward. DDAS [9] directly derives the search formula from training loss without the Gumbel-Softmax estimator for a more accurate gradient estimation. These differentiable methods use gradient update strategy to solve the optimization problem and reduce the search cost in order to be affordable. However, they mainly adopt a fix magnitude that limits the scale of augmented space and the upper bound for model performance.

On the other hand, online data augmentation methods adjust policy parameters dynamically during model training. OHL-Auto-Aug [12] and Adversarial Augment [11] jointly adjust policy parameters and model parameters on the target dataset, which dynamically augment images during training and adjust the augmentation policy without retraining the model. These online search methods achieve superior results compared with offline methods, but the search overhead remains large; meanwhile, the policy is hardly transferred to other tasks or also suffers from obvious performance degradation. Shortages limit the wide application of these online search methods. Meta online data augmentation [15] and online bi-level optimization [16] for data augmentation search also use differentiable learning to reduce the overheads of online methods, while they severely sacrifice performance.

Another simple but effective way to augment images is to introduce random factors for transformations. RandAugment [5] (RA), UniformAugment [14] (UA) and TrivialAugment [13] (TA) carefully design the augmentation ranges and uniformly sample transformations to augment the input data. The simple design shows amazing performance in classification tasks. However, as these random factors are non-specific to datasets, there exists a great chance for improvement. RA also suffers from heavy constraints and a large search cost to find optimal policy parameters.

Very recently, DAAS [33] and DHA [34] jointly optimized policy parameters with architecture parameters or even hyperparameters. These methods try to find an optimal augmentation strategy that fits the searched model. However, they suffer from the same limitations as online search methods.

III. Differentiable RandAugment

In this section, we first reformulate automatic data augmentation and define our search space based on RandAugment. Then, we introduce the relaxation and approximation for differentiable learning. After that, we revisit the optimization objective of the differentiable augmentation search, and add KL divergence as a loss item to alleviate the inconsistency between updating policy parameters and model parameters.

A. Reformulate Data Augmentation

Data augmentation can be presented as a sequence of operations to transform the input image, which can also be viewed as sequentially piled layers in the model before normalization. Let D be the length of the operation sequence, which we also call total augmentation depth as follows. Take the d^{th} augmentation layer as an example. Let $X^0 \in \mathbb{R}^{H\times W}$ be the original image, where H and W are the height and width of the image, respectively. $X^{d-1} \in \mathbb{R}^{H\times W}$ is the input of the d^{th} augmentation layer, and $X^d \in \mathbb{R}^{H\times W}$ is the output of the

d^{th} augmentation layer. The augmentation of one layer can be denoted as

$$X^d = \sum_{o \in O} o\left(X^{d-1}\right) \cdot c_o^d, \quad d = 1, 2, \cdots, D, \tag{1}$$

where

$$c^d \sim Categorical\left(w^d\right), \tag{2}$$

and

$$o\left(X^{d-1}\right) = \begin{cases} \mathbf{1}, & if\ c_o^d = 0 \\ OP_o\left(X^{d-1}, m_o^d\right), & if\ c_o^d = 1\ and\ p < p^d \\ X^{d-1}, & otherwise \end{cases} \tag{3}$$

Here, $\mathbf{1} \in \mathbb{R}^{H \times W}$ is a matrix with the same shape of X^{d-1} and all elements of value 1, $c^d = \left[c_1^d, c_2^d, \cdots, c_N^d\right] \in \mathbb{R}^N$ is a one-hot vector that denotes the sampled categorical result, where N denotes the number of candidate operators. $w^d = \left[w_1^d, w_2^d, \ldots, w_N^d\right] \in \mathbb{R}^N$ denotes the weights of operators in the d^{th} layer, $OP_o(\cdot)$ denotes the o^{th} transform operator in O, where $O = \{OP_1, OP_2, \ldots, OP_N\}$ denotes the set of candidate operators in the search space. m_o^d denotes the magnitude to apply the o^{th} operator in the d^{th} augmentation layer. The matrix $\mathbf{1}$ is introduced to avoid the calculation of operators that are not sampled, while adding the calculation of selecting weights of these unsampled operators in the forward without changing the results. Thus, selecting weights of unsampled operators can also be updated during back propagation. Equation (3) shows that the sampled operator in the d^{th} augmentation layer has p^d probability to be applied, while $1 - p^d$ probability unchanged. Note that p^d is a sampled value in our DRA, which follows a uniform distribution. This design follows the implementation of RA in TensorFlow models.[1] We sample a new $p \sim U(0, 1)$ and $p^d \sim U\left(p_{min\_t}, p_{max\_t}\right)$ before each augmentation layer for each minibatch, where p_{min_t} and p_{max_t} are two hyperparameters defined before the search.

In RA, operators have the same probability, or they follow a categorical distribution to be sampled as proposed by Wightman et al. [38] when manually designed weights are given. Here, the weights w^d are learnable parameters in our search space, which determine the selecting weights of the operators. This design is the same as the one mentioned in RA, while the authors only explored the impact of differentiable selecting weights without considering magnitudes. Note that unlike methods based on the search space of AA that separately learn the parameters of each sub-policy [7], [8], [10], [31], our DRA has an operator-sharing strategy in the d^{th} augmentation layer for sub-policies to reduce the number of learnable parameters. This design allows sampling operators within an augmentation layer, thus facilitating propagating gradients of the operators between augmentation layers. The illustration of the AA and RA search space is shown in Supplementary Materials Fig. S1.

Besides, to improve the generalization ability of the target model, the magnitude of each operator in DRA follows a learnable distribution to generate more transformations of the input images. This design is motivated by the idea that randomized magnitude can improve the diversity of the data, meanwhile the augmented images should follow a similar distribution to the original ones to alleviate the over-transformation problem [10], [39]. Since the magnitude controls the deformation to the original image, with 0.0 indicating no change and 1.0 indicating the maximum, using different magnitudes for the same operator to generate different transformations of the input is an intuitive idea. The effectiveness of variant magnitudes in data augmentation has also been shown in TA and UA. Thus, we assume that introducing randomness to the magnitudes can yield abundant transformations to improve model performance. On the other hand, since fixed magnitudes in sub-policy-based methods [7], [31] achieve good results, we assume that augmented images that are similar to images generated by the learned sub-policies can yield better performance. As a result, we adopt a magnitude sampling strategy that samples magnitude from a specific normal distribution rather than uniform distribution within the feasible region. We model the magnitude of each operator in each augmentation layer following a separate learnable normal distribution. Under this setting, the augmented images are expected to have more variants that are close to the one transformed with the mean magnitude, while there still exist fewer variants having larger deformations. However, when the standard deviation of magnitude has a large value, the magnitude distribution becomes smoother. The smoother distribution will generate diverse transformations, which may increase the number of over-transformed images. To alleviate the over-transformation raised by the uncontrolled sampling from normal distribution, we minimize KL divergence that measures the distance between the distributions of augmented samples and the original ones during policy parameter learning, which is discussed in Section III-C. The magnitude in our method can be denoted as

$$m_o^d \sim N\left(\mu_o^d, \sigma_o^{d^2}\right), \tag{4}$$

where μ_o^d and σ_o^d are the mean and standard deviation of the magnitude of the o^{th} operation in the d^{th} augmentation layer, respectively. We emphasize that the design of magnitudes following normal distributions is the core idea of our work, which is different from previous works such as fixed magnitudes in RA or learnable magnitudes in Faster AA and DADA.

In our search space, we have weight parameters, means of magnitudes, and standard deviations of magnitudes that are learnable. These learnable parameters are called policy parameters in this paper. Note that the probability p^d of applying the operator in the d^{th} augmentation layer is not in our search space, which reduces the number of learnable parameters and further eases the burden of the search algorithm. The total number of learnable policy parameters is $3 \times D \times N$.

B. Estimate Gradient of Policy Parameters

1) Relax Weight Parameters: Since sampling is not differentiable w.r.t. weight parameters, relaxation is conducted in the backward propagation to make weight parameters differentiable. In the relaxed setting, Gumbel-Softmax [40], [41]

[1] https://github.com/tensorflow/models.git

estimator is introduced and (1) becomes

$$X^d = \sum_{o \in O} o\left(X^{d-1}\right) \cdot \hat{c}_o^d, \quad d = 1, 2, \cdots, D, \tag{5}$$

where

$$\hat{c}^d \sim RelaxCategorical\left(w^d, \tau\right) \tag{6}$$

and

$$\hat{c_o}^d = \frac{\exp\left(\left(w_o^d + g_o^d\right)/\tau\right)}{\sum_{o' \in O} \exp\left(\left(w_{o'}^d + g_{o'}^d\right)/\tau\right)}. \tag{7}$$

Here, $\hat{c}^d = \left[\hat{c}_1^d, \hat{c}_2^d, \ldots, \hat{c}_N^d\right] \in \mathbb{R}^N$ that makes w_o^d differentiable denotes the relaxation of c^d, $g_o^d = -log\left(-log\left(u\right)\right)$ where $u \sim U\left(0, 1\right)$, and $\tau > 0$ is the temperature to control how similar the relaxed distribution is to the expected value. Smaller τ results in a more one-hot-like relaxation. The relaxed equation is calculated in the backward. While in the forward, we adopt calculations following GDAS [36], where the equations are

$$X^d = \sum\nolimits_{o \in O} o\left(X^{d-1}\right) \cdot h_o^d, \quad d = 1, 2, \cdots, D, \tag{8}$$

$$h^d = \mathcal{H}\left(\arg\max_o \hat{c}^d\right). \tag{9}$$

Here, $h^d = \left[h_1^d, h_2^d, \ldots, h_N^d\right] \in \mathbb{R}^N$, and $\mathcal{H}$ is a one-hot operation. The one-hot processing makes the forward consistent with the original sampling, meanwhile avoiding the calculation of operators with a small weight in (5) to reduce memory consumption.

2) Learn Magnitude Distributions: Unlike optimizing weight parameters, the gradient of magnitude parameters needs approximation, because some operators (e.g., *posterize*, *solarize*) are not differentiable w.r.t. magnitudes. Thus, we use the straight-through estimator [42] to evaluate the gradient of magnitudes. The straight-through estimator can be denoted as

$$\frac{\partial OP_o\left(X^{d-1}, m_o^d\right)}{\partial m_o^d} = \mathbf{1}, \tag{10}$$

when assuming each pixel in the image is independent. Here, $OP_o\left(X^{d-1}, m_o^d\right) \in \mathbb{R}^{H \times W}$ is the augmented image of X^{d-1} by operator o in the d^{th} augmentation layer with a magnitude m_o^d. Equation (10) means the gradient of magnitude w.r.t. each pixel of the augmented image is 1.

To pass the gradient of magnitudes in the back propagation, magnitudes are specially calculated in the forward and the original X^{d-1} becomes

$$\hat{X}^{d-1} = X^{d-1} + m_o^d - StopG(m_o^d), \quad d = 1, 2, \cdots, D, \tag{11}$$

where $\partial X^{d-1} / \partial m_o^d = \mathbf{0}$ and $\mathbf{0} \in \mathbb{R}^{H \times W}$ is a matrix with the same shape as X^{d-1} and all elements have a value 0, and $StopG\left(\cdot\right)$ is the stop gradient operation that gets the value of the input without passing the gradient, which is a constant in (11). Note that we do not stop the gradient of X^{d-1} to pass the gradient of operator w.r.t. X^{d-1}. A reasonable update strategy is to only update the magnitudes of sampled operators in the forward path, thus (3) becomes

$$o\left(X^{d-1}\right) = \begin{cases} \mathbf{1}, & if\ h_o^d = 0 \\ OP_o\left(\hat{X}^{d-1}, m_o^d\right), & if\ h_o^d = 1\ and\ p < p^d \\ X^{d-1}, & otherwise \end{cases} \tag{12}$$

As mentioned in Section III-A, the magnitude of each operator in each augmentation layer of DRA follows a normal distribution. However, the sampling operation is not differentiable w.r.t. magnitude parameters. Therefore, we use the reparameterization trick [43] to make magnitude parameters differentiable. Reparameterization can be denoted as

$$\hat{m}_o^d = \mu_o^d + \epsilon \cdot \sigma_o^d, \quad \epsilon \sim N\left(0, 1\right), \tag{13}$$

which shifts the sampling operation to a standard normal distribution and makes magnitude parameters differentiable.

Note that when using magnitudes following normal distributions, RA can be seen as a special case of DRA, with equivalent w and μ for each operator, and σ of a value equal to 0 for each operator. However, RA is not in our search space if not using the expanded transformation range, due to the over-range of magnitudes of RA with a maximum magnitude level of 30 rather than 10, where 10 indicates the border of the transformation range.

3) Use the Operator Gradient: Data augmentation is usually decoupled with model training (e.g., augmentation based on Pillow[2]), which introduces the straight-through gradient estimator to policy parameters if not editing the backward process. However, the straight-through estimator is biased. To reduce the impact of this bias, Hataya et al. [10] uses operators in Kornia [44], a PyTorch-based [45] differentiable computer vision library, to augment data and calculate the operator gradients w.r.t. the input of the operator. Similarly, we rewrite the operators using TensorFlow [46] and encapsulate them into differentiable layers. For operators that are not differentiable to the input (e.g., *posterize*, *equalize*), we use the straight-through estimator to pass the gradient directly.

C. Revisit the Optimization Objective in the Differentiable Data Augmentation Search

Bi-level optimization becomes the standard and widely applied optimization objective for differentiable learning. It separates the training dataset into two equal subsets for alternatively updating model parameters to the optimal (the inner loop) and one iteration of architecture parameters (the outer loop) to achieve the minimum loss w.r.t. architecture parameters. As proposed in DARTS [35], the one-step gradient update can reduce the expensive inner optimization cost in bi-level optimization, which uses the results of one iteration in the inner loop to approximate the optimal model parameters. In differentiable automatic data augmentation, policy parameters show similar functions to architecture parameters. As a result, policy parameters can be optimized using the same strategy. However, we revisit the optimization objective of the

[2]https://python-pillow.org

differentiable data augmentation search and find an inconsistency between the outer optimization and inner optimization. Specifically, the optimization objective of the differentiable data augmentation search can be written as

$$\begin{aligned} &\min_{T} \ L_{val}\left(\theta^{*}\right) \\ &s.t. \ \theta^{*} = arg\min_{\theta} L_{train}\left(\theta, T\right), \end{aligned} \tag{14}$$

where L_{val} and L_{train} denote validation and training loss, respectively. Note that the training loss and validation loss in (14) are calculated on the two halves of the proxy dataset rather than on the target dataset. θ denotes the model parameters, and T denotes policy parameters. However, T is not included in the calculation of the loss of the outer loop L_{val}, which makes T indifferentiable in outer optimization. An approximation uses $L_{val}\left(\theta^{*}, T\right)$ instead of $L_{val}\left(\theta^{*}\right)$, which introduces a gap between the real target and approximation.

Recently proposed contrastive learning focuses on the distributions of augmented views using contrastive loss, with the aim of maximizing the agreement of similar images while expanding the differences between views from different inputs [25], [26], [27]. Inspired by the idea, we hypothesize that to achieve good performance, the augmented images should have similar logits after Softmax to the original ones, meanwhile keeping the validation loss low. Therefore, a metric to measure the similarities and distances between augmented images and original images after inference is required to achieve the expectation. This idea is consistent with the previous viewpoint that data augmentation is a process to fill in the missing points of the original data distribution through density matching [10]. KL divergence, which is the most widely used metric to measure the differences between two distributions [47], is selected in our setting. It has also been used in adaptive knowledge distillation for different training samples, which shows a similar nature to our DRA [39], [48]. To reduce the gap between the original optimization objective and the approximation, we add KL divergence to the loss function. Thus, the optimization objective becomes

$$\begin{aligned} &\min_{T} \ L_{val}\left(\theta^{*}, T\right) + \lambda \cdot KL(p^{ori} || p^{aug}) \\ &s.t. \ \theta^{*} = arg\min_{\theta} L_{train}\left(\theta, T\right), \end{aligned} \tag{15}$$

where p^{ori} denotes the logits of the original image after Softmax, p^{aug} denotes the logits of the augmented image after Softmax, and λ controls the weight of KL divergence. The proposed loss is expected to reduce the gap between the augmented image space and the original one, achieving density matching to yield better performance. Meanwhile, the usage of KL divergence to measure the differences between the original and augmented image from the same one also reduces the general risk of over-transformation.

D. The Relationship to DADA, DDAS, and Faster AA

Prior offline works DADA, DDAS, and Faster AA share similar spirits to DRA, all of which use differentiable learning to estimate the policy parameters. Apart from the learnable magnitude distributions specially proposed in DRA, various designs between these works are different as well. To highlight innovations of DRA, we list some key differences here.

DADA adopts AA-based search space that uses separate sub-policies to augment images. It uses the RELAX estimator with second order gradient estimation to learn a more accurate policy. In contrast, DRA uses RA-based search space that shares operators in each augmentation layer to reduce the number of learnable parameters. The gradients are estimated directly through back propagation without second order gradient estimation. Meanwhile, DRA uses KL divergence to reduce the impact of biased straight-through gradient estimation for a more accurate policy.

DDAS directly uses the expectation of the training loss to derive the formulas of policy parameters without gradient estimators. It adopts a repeated augmentation strategy for the same minibatch to estimate loss expectation. The same operator with different magnitudes is treated as different candidate operators for training to avoid the estimation of indifferentiable magnitudes. In contrast, DRA only requires augmenting the same minibatch once to reduce the search cost when applying the operator requires much calculation. In addition, gradient estimators are kept to learn the magnitudes for more flexible policies.

Faster AA also adopts RA-based search space to learn policy parameters. It passes the weighted sum of transformed images in each augmentation layer to estimate the gradients during search, which is very close to the design in DARTS for feature aggregation to pass the information flow between inner nodes. Besides, it adopts adversarial learning through Wasserstein GAN to estimate the distance between the augmented images and original images to achieve density matching, where the estimation of the distance is based on two different minibatches from the training dataset. This design avoids the nested loop in bi-level optimization, where the outer loop has no gradient for the policy parameters in the basic design. In contrast, DRA uses one-hot relaxed categorical sampling for the operators in each augmentation layer in the forward that only transforms the image once, which reduces the computation time, especially on the large dataset ImageNet. Besides, it adopts the bi-level optimization strategy with an approximation to estimate the gradient of policy parameters, solving the problem in the outer loop. KL divergence is also adopted to achieve density matching without any additional model to estimate the distribution distance.

IV. Experiments

In this section, we conduct classifications on CIFAR-10/100 and ImageNet, and compare the performance of DRA and some other augmentation methods. The results of these compared methods are from the original papers, if not specifically mentioned. These results are expected to be excellent, since the authors usually tuned the settings to adapt to the proposed methods. Thus, the comparison with DRA is relatively fair and acceptable. Since RA is the most concerned method that has a similar augmentation pipeline of DRA, we also re-implemented RA under our settings for comparison. Note that the original RA has only 14 operators in the search

TABLE I
TYPE, NAME, AND TRANSFORMATION VALUE RANGE OF EACH OPERATOR IN DRA USING TENSORFLOW. VALUES BEFORE AND AFTER "/" INDICATE CIFAR AND IMAGENET RANGE, RESPECTIVELY. FOR COLOR BLENDING OPERATORS, VALUE 1.0 INDICATES ORIGINAL IMAGE, WHILE 0.1 AND 1.9 INDICATE MAXIMUM TRANSFORM

Type	Operator Name	Transformation Range
Geometric	ShearX	[-0.3, 0.3]
	ShearY	[-0.3, 0.3]
	TranslateX	[-10/-100 px, 10/100 px]
	TranslateY	[-10/-100 px, 10/100 px]
	Rotate	[-30°, 30°]
Color	Brightness	[0.1, 1.9]
	Color	[0.1, 1.9]
	Sharpness	[0.1, 1.9]
	Contrast	[0.1, 1.9]
	Cutout	[0.0, 0.2]
	Solarize	[0, 256]
	Posterize	[0, 4]*
	Equalize	-
	AutoContrast	-
	Invert	-
	SolarizeAdd	[0, 110]

px: Pixel. *: Equivalent to range [4, 8] in Pillow.

space, while there are 16 operators in the re-implemented RA and our DRA. Our DRA shows superior performance compared with other methods, especially on ImageNet. To further demonstrate the generalization ability of DRA in downstream tasks, we conduct transfer learning on COCO and compare the performance of RetinaNet [49] and GFLV2 [50] with different pre-trained backbones based on basic settings, RA, and DRA. We also visualize the changes of policy parameters during the search process and the searched results for a better understanding of DRA.

A. Implementation Details

Our search is conducted on the proxy task using a split from the original training dataset with fewer search epochs, without using the original validation dataset to update any learnable parameter. The proxy task greatly decreases the search cost in a widely affordable manner, especially on large-scale datasets like ImageNet. Half of the proxy dataset is used for updating policy parameters, while another half is used for updating model parameters. Note that unlike previous methods, we do not use a proxy model to search for policy parameters.

We have 16 operators in our candidate operator set O, which generally follows the implementation of RA in TensorFlow models. The names and transformation ranges of operators are listed in Table I. We group these operators into two parts:

- **Geometric Operators.** Geometric shape or the position of the image are transformed. (e.g., TranslateX, ShearY, Rotate)
- **Color Operators.** The general geometric shape of the image remains unchanged, while the pixel values of part of the image or the whole image are transformed (e.g., Solarize, Sharpness, Equalize).

Note that the ranges for *translate* operators use pixels instead of the ratios in DRA. We use WRN-28-10 [51] and PyramidNet+ShakeDrop [52], [53] to evaluate DRA on CIFAR, while ResNet-50 [54], ResNet-200, and vision Transformer DeiT-Tiny-16-224 [17] without distillation on ImageNet. Detailed hyperparameter settings for all classification experiments are listed in Supplementary Materials Table SIII. Note that we tune the initial value of μ to 1.5 when using ResNet-200, since the model has a stronger ability to capture features compared to ResNet-50. The transformation range is also extended to [0, 3], where the magnitude $m = 3.0$ in DRA is equivalent to the magnitude level $M_{RA} = 30$ in RA. The range of the sampled magnitude is truncated to [0.0, 1.0] or [0.0, 3.0], except the magnitudes of *shear*, *translate*, *rotate* and color blending operators (including *brightness*, *color*, *sharpness*, and *contrast*) that have 50% probabilities to be negative. This is because we use 0-1 normalized transform intensity, where 0 means no transformation applied and 1 means the maximum transformation in the range. Besides, to reduce the memory consumption of large batches on ImageNet settings, we use the Inplace-ABN technique [55] during training to save memory without obvious influence on model performance. All Transformers are trained from scratch. The warmup of learning rates starts from 0 for all training processes. We use TensorFlow-based operators with a gradient for the differentiable search, while operators without gradient are used for fast training.

For object detection, we select ResNet-50 as the backbone, and apply transfer learning that only uses different pre-trained weights for backbones. For RetinaNet, we use the horizontal flip as the augmentation, and set the batch size to 32, weight decay to 1×10^{-4}, epochs to 24, and randomly pad the input to a resolution of 896×896. We use SGD with a multistep learning rate schedule that is warmed up for 500 iterations to a learning rate of 0.02, and divide it by 10 after 16 and 22 epochs. For GFLV2, since the detection head has a stronger ability to capture features in the input, we adopt augmented settings to train it. We use horizontal flip and random resized crop from a range of 0.1 to 2.0 to augment the input, and set the batch size to 64, weight decay to 0.05, epochs to 50, and crop the input to a resolution of 1024×1024. We use AdamW for faster convergence with a learning rate warmed up for 1 epoch from 0 to 8×10^{-4} followed by a cosine learning rate schedule completed at 0. We compare three augmentation settings for pre-training the backbones including the basic, RA, and DRA. The pre-training settings are the same as mentioned in the ImageNet classification experiments, except that we do not use Inplace-ABN and only train the basic one for 120 epochs.

We implement our experiments on TensorFlow 2.3 and Python 3.7. All search experiments are conducted on a single Tesla P100, and other experiments are conducted on TPU v2 (8 Cores).

B. Results on CIFAR-10/100

CIFAR-10 and CIFAR-100 are two small datasets with balanced class distributions. They both have a resolution of

TABLE II

ACCURACY (%) OF DIFFERENT AUGMENTATION METHODS ON CIFAR-10/100. RESULTS OF MEAN ± STD ARE REPORTED ON THE AVERAGE OF THREE RUNS. PYRAMIDNET IS SHORT FOR PYRAMIDNET+SHAKEDROP. BEST RESULTS ARE IN BOLD

	Base-line	Cutout	AA	Fast AA	Faster AA	DADA	DDAS	RA	RA*	DRA
CIFAR-10										
WRN-28-10	96.1	96.9	97.4	97.3	97.4	97.3	97.3	97.3	97.27±0.06	**97.44±0.10**
PyramidNet	97.3	97.7	98.5	98.3	-	98.3	-	98.5	98.50±0.03	**98.54±0.03**
CIFAR-100										
WRN-28-10	81.2	81.6	82.9	82.8	82.7	82.5	83.4	83.3	83.82±0.20	**83.85±0.16**
PyramidNet	86.0	87.8	**89.3**	88.3	-	88.8	-	-	88.91±0.12	89.17±0.09

*: Reimplemented under our settings with 16 operators in the search space.

TABLE III

SEARCH TIME (GPU HOUR) OF DIFFERENT AUGMENTATION METHODS ON DIFFERENT DATASETS

	AA	PBA	Fast AA	Faster AA	DADA	DDAS	RA†	DRA
GPU Device	Tesla P100	Titan XP	Tesla V100	Tesla V100	Titan XP	RTX 2080Ti	RTX 2080Ti	Tesla P100
CIFAR-10/100	5000	5	3.5	0.23	0.1/0.2	0.15	33	0.40‡
ImageNet	15000	-	450	2.3	1.3	1.2	4750	0.95*

†: Reported in DDAS. ‡: Evaluated using WRN-28-10. *: Evaluated using ResNet-50.

32×32, with 50,000 images in the training dataset and 10,000 in the test set. For both CIFAR-10 and CIFAR-100, we follow previous works and conduct the search on 4,000 randomly split samples from the training dataset. The basic augmentation includes random cropping and random horizontal flipping. DRA is applied after basic augmentation and before Cutout. Table II and Table III compare the performance and search cost of DRA and other data augmentation methods, respectively. We run each experiment three times and report the mean and standard deviation of the accuracy. Note that we re-implement RA under our framework for a fair comparison. Besides, since the original RA does not report the hyperparameter settings and performance of PyramidNet+ShakeDrop on CIFAR-100, we use the same hyperparameter settings as CIFAR-10 with the number of operators in one sub-policy $N_{RA} = 3$ and magnitude for all operators $M_{RA} = 7$.

As shown in Table II, DRA improves the accuracy on both CIFAR-10 and CIFAR-100 compared with RA and other methods. We note that the re-implemented WRN-28-10 using RA under our settings has an obvious performance increase compared with the reported one on CIFAR-100, which may arise from the operators that are only in the search space of DRA working well on CIFAR-100. We also notice that DRA performs slightly worse than AA on CIFAR-100 using PyramidNet+ShakeDrop, while better using WRN-28-10. The inconsistency may arise from two aspects. PyramidNet+ShakeDrop benefits more from the separate sub-policies in AA on CIFAR-100 that consider the detailed impact of previous transformations for PyramidNet+ShakeDrop to distinguish from. In contrast, WRN-28-10 benefits less due to its limited ability to distinguish the detailed changes. Besides, different policy models have different impact on DRA, where PyramidNet+ShakeDrop benefits less from DRA compared with WRN-28-10. Further analysis of the impact of different proxy model structures on DRA is shown in the Section V-F.

The search costs of different augmentation methods are shown in Table III. We notice that DRA has a smaller search time difference between CIFAR and ImageNet compared with other methods. The reasons are from two aspects.

On one hand, ImageNet has a resolution of 224×224 during the search, resulting in a longer time for transformations. DRA adopts RA-based search space that has fewer policy parameters to learn compared to AA-based search space, resulting in less search time. Meanwhile, DRA uses one-hot sampling for the operators in each augmentation layer in the forward that only transforms the image once, which is more efficient than Faster AA that uses a weighted sum of all transformations. While compared with DDAS, DRA does not require repeated augmentation for the same minibatch. As a result, DRA is more efficient on ImageNet.

However, DRA uses WRN-28-10 as the proxy model, while the counterparts adopt WRN-40-2. Since the proxy dataset from CIFAR is small, the efficiency of DRA is not obvious, while the influence of a larger proxy model dominates the search time. Although the search time of DRA is slightly longer than other differentiable methods, it is still within 0.5 GPU hours, which is short and affordable for wide applications.

C. Results on ImageNet

ImageNet is a large-scale dataset with an almost balanced class distribution. It has 1.3 M images in 1,000 classes from daily life, which is totally different from the ones in CIFAR. As mentioned by DeVries and Taylor [1] and Cubuk et al. [7], augmentation that performs well on one dataset may not work

TABLE IV
ACCURACY (%) OF DIFFERENT AUGMENTATION METHODS ON IMAGENET. RESULTS OF MEAN±STD ARE REPORTED ON THE AVERAGE OF THREE RUNS. BEST RESULTS ARE IN BOLD

	Base-line	AA	Fast AA	Faster AA	DADA	DDAS	RA	RA*	DRA
ResNet-50									
Top-1	76.3	77.6	77.6†	76.9	77.5	78.0	77.6‡	77.91±0.12	**78.19±0.08**
Top-5	93.1	93.8	93.7†	93.5	93.5	-	93.8‡	93.88±0.05	**94.01±0.01**
ResNet-200									
Top-1	78.5	80.0	80.6†	-	-	80.5	-	81.10±0.26	**81.16±0.05**
Top-5	94.2	95.0	95.3†	-	-	-	-	95.50±0.05	**95.58±0.01**
DeiT-Ti-16									
Top-1	71.03*	-	-	-	-	-	72.2⁎	73.29±0.05	**73.45±0.05**
Top-5	89.47*	-	-	-	-	-	91.1⁎	91.12±0.04	**91.16±0.06**

†: Training for 200 epochs. ‡: Training for 180 epochs with image resolution 224×244. *: Reimplemented under our settings. ⁎: Having a standard deviation of 0.5.

well on another with different data distributions. Thus, directly conducting a search on the task dataset is a proper choice. Thanks to the efficiency of DRA, we can directly search on the proxy of ImageNet rather than transferring the policy searched on CIFAR. Following AA, we randomly sample 120 classes from 1,000 classes in the training dataset of ImageNet with 50 images per class as the proxy dataset, and search for the policy parameters on it. The Top-1 and Top-5 accuracy of DRA and other augmentation methods are shown in Table IV, which is also evaluated on three runs. Similar to experiments on CIFAR, we reimplement RA under our framework for a fair comparison. Note that the original training hyperparameter settings of RA are different from other works, so we re-select the value of RA hyperparameters and use $N_{RA} = 2$ and $M_{RA} = 10$ in the following experiments as the baseline. This setting is similar to the one in the original paper ($N_{RA} = 2$, $M_{RA} = 9$), and has been adopted in other works [56], [57]. We also report the mean and standard deviation under three runs.

As shown in Table IV, DRA has the best accuracy compared with other methods with obvious improvement up to 0.28% Top-1 accuracy over the re-implemented RA on ResNet-50. Even the re-implemented RA has great improvement over the original one, which attributes to the longer training time for the model to converge with augmented images. Note that the re-implemented RA under our settings shows competitive Top-1 performance to the re-implemented one reported by Liu et al. [9], demonstrating that the result is reasonable. DRA also achieves the best performance on ResNet-200, demonstrating the capability of DRA to adapt to different target models. With DRA, the model learns more solid features that yield better performance. However, the improvement over RA on ResNet-200 is not as obvious as the one on ResNet-50. We guess that this is due to the stronger ability of ResNet-200 to capture features in the input data, which may prefer larger variances of the magnitudes of operators. With a larger value of the initial standard deviation of magnitude σ of DRA, the performance of ResNet-200 may further increase.

To further evaluate the effect of DRA on non-convolutional neural networks, we conduct experiments on vision Transformer DeiT-Tiny with a patch size of 16, which is denoted as DeiT-Ti-16 in Table IV. No distillation is applied during training. The input resolution is set to 224. We reimplement DeiT-Ti-16 with the same training hyperparameters except for three augmentation settings, including the basic, RA, and DRA. Since the comparison is designed to reveal the power of DRA compared with RA on Transformers, other augmentations apart from the basic random resized cropping and horizontal flipping are not applied. The hyperparameter settings generally follow the original DeiT-Basic. Note that for RA we set $N_{RA} = 2$ and $M_{RA} = 9$, with a magnitude standard deviation $\sigma = 0.5$. Apart from the additional usage of a standard deviation that follows the setting in DeiT rather than the original RA, we also set RA with a random applying probability $p^t \sim U(0.2, 0.8)$ for each operator during each iteration to be consistent with DRA. The models are trained from scratch, where each experiment is evaluated on three runs.

As shown in Table IV, DRA also achieves the best performance on ImageNet compared with the basic and RA, with 0.16% Top-1 performance improvement. We also notice a significant performance improvement is achieved in our reimplemented version compared with the original one in [17], which may arise from more candidate operators in DRA. We also evaluate the performance of RA without the random applying probability to be consistent with the setting in [17] and the implementation without standard deviation as the original design of RA, where the Top-1 accuracy is only 73.21% and 72.84% (not shown in Table IV), respectively. The results demonstrate the generalization of DRA to be integrated into more types of neural network structures, indicating bright prospect for wide applications.

D. Transfer Learning on Object Detection

Using ImageNet pre-trained backbones is a common strategy in object detection for fast convergence [58]. To evaluate

TABLE V
OBJECT DETECTION RESULTS EVALUATED ON COCO MINIVAL. DIFFERENT AUGMENTATION PRE-TRAINED BACKBONES ARE USED FOR TRANSFER LEARNING. RESULTS OF AP (%) ARE REPORTED ON THE AVERAGE OF THREE RUNS. BEST RESULTS ARE IN BOLD

Augment	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
RetinaNet						
Basic	37.5	56.5	40.2	19.9	41.8	50.3
RA	37.7	56.8	40.3	**21.0**	41.9	**50.7**
DRA	**37.9**	**57.0**	**40.6**	20.9	**42.1**	**50.7**
GFLV2						
Basic	45.5	62.9	49.5	28.1	49.5	58.7
RA	45.6	62.4	**49.8**	**28.7**	49.7	**59.2**
DRA	**45.7**	**63.1**	49.7	28.6	**50.0**	**59.2**

whether the backbone network trained with our DRA has a stronger ability to help extract features in object detection, we conduct experiments on the COCO dataset with pre-trained backbones under three different settings, including the basic, RA, and DRA. We test two single-stage detection networks, RetinaNet and GFLV2, to better understand the influence of pre-trained backbones on the performance where detection heads have either weak or strong feature extraction capability. We compare the mean bounding box Average Precision (AP) to evaluate the performance. The results are shown in Table V on the average of three runs. Models using backbones pre-trained with DRA generally outperform the basic and RA. In particular, DRA mainly helps to capture features of medium-sized objects, while slightly sacrificing the ability to capture those of smaller-sized objects. DRA also helps to detect the coarse location of objects. These phenomena are shown in both simplified settings and augmented settings, which may explain how DRA helps models to extract semantic features in the images.

E. Visualization

1) Visualize the Search Process: To better understand how DRA works during the search, we visualize the changes of the mean magnitude μ and selecting weights w of different augmentation layers during the search process on CIFAR-10 and ImageNet using WRN-28-10 and ResNet-50, respectively. The results are shown in Fig. 2.

As shown in Fig. 2 (a), DRA prefers a larger transform intensity of geometric transformations while less for color blending operators on CIFAR-10. Changes of μ of the same operator in different layers are almost consistent, while they are more severe for geometric transform operators in the deeper augmentation layer.

For ImageNet shown in Fig. 2 (b), μ generally sharply changes compared with CIFAR-10, indicating that ImageNet requires stronger augmentations to yield better results. Geometric transformation operators require larger magnitudes, while color blending operators require smaller ones. Since μ of some operators quickly reaches the upper bound of DRA, we hypothesize that extending the range of these operators can yield better performance.

We also notice that the changes of w on both CIFAR-10 and ImageNet are not obvious compared with the initial values. These small changes bring slight disturbance to uniform sampling, allowing operators that are more favorable to the dataset to be sampled at a higher frequency, while ensuring increased diversity of augmented inputs. However, the small disturbance may also explain why the improvement of only learning the selecting weight of RA is not obvious on complex models and datasets, as reported by both Cubuk et al. [5] and in our experiments in the discussion section.

2) Visualize the Searched Results: We also display the final searched magnitude parameters of both the mean and standard deviation of magnitudes on CIFAR-10 and ImageNet in Fig. 3 with WRN-28-10 and ResNet-50, respectively. Note that the sampled magnitudes of DRA should be within range $[0, 1]$ despite the distributions, thus the values out of the range should be clipped. We show the complete ranges of $\mu \pm \sigma$ learned by DRA to give a more accurate description of the sampled results of magnitudes after clipping, especially when the learned range has a part out of $[0, 1]$. Both CIFAR-10 and ImageNet prefer large mean magnitudes for geometric transformations, while the mean magnitudes for color transformations vary from different operators. The learned mean magnitudes of ImageNet are generally larger than CIFAR-10 on the 1^{st} augmentation layer, indicating a stronger ability of models to learn features from images with heavier transformations on ImageNet. However, it is not the case for the 2^{nd} augmentation layer. We find that the difference may be due to the influence of the operator gradient during the search, which only affects the estimation of policy parameters in augmentation layers except for the deepest one. This is because the operator gradient is passed through adjacent augmentation layers, thus the policy parameters in the deepest layer have no operator gradient passed. We also notice that the operator Cutout prefers to have a larger standard deviation, which allows the generated images to have different sizes of masked blocks. The learned policy parameters are reasonable to understand. In general, CIFAR-10 and ImageNet prefer different distributions of the policy parameters, while the policy parameters in different augmentation layers of the same dataset do not vary severely. Detailed values for all searched policy parameters are listed in Supplementary Materials Tables SI and SII.

3) Visualize the Loss Curves: To reveal the impact of DRA on training the target model, we further analyzed whether DRA makes training easier or harder. The loss curves during training on ImageNet are used to measure the difficulty, which is shown in Fig. 4. DRA has a larger augmented image space compared with RA, which is expected to increase the difficulty of training. Surprisingly, the results show that DRA has both a smaller training and validation loss at the end of the training. Meanwhile, the loss gap of DRA between training and validation is also smaller. These results indicate that DRA makes samples easier to be trained compared with RA. We analyze that the reasons are from two aspects. For one thing, DRA has a smaller average magnitude compared with RA, which makes models easier to extract features from variants of the original image. For another, since DRA has a standard deviation that makes the sampled magnitudes

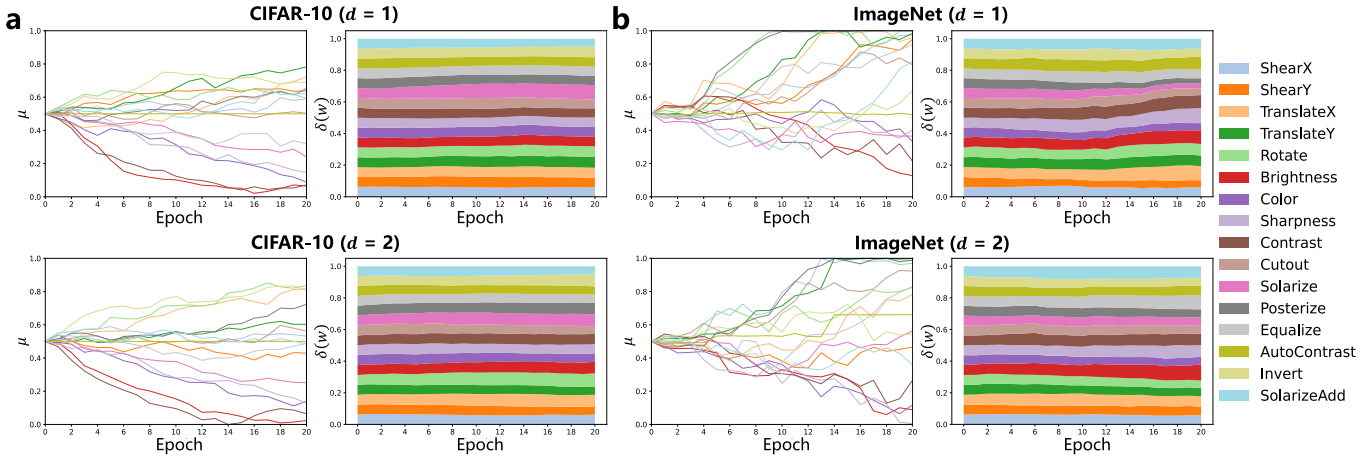

Fig. 2. Visualization of the search process for mean magnitudes μ and selecting weights w (after Softmax δ) in different augmentation layers. a) CIFAR-10 using WRN-28-10. b) ImageNet using ResNet-50.

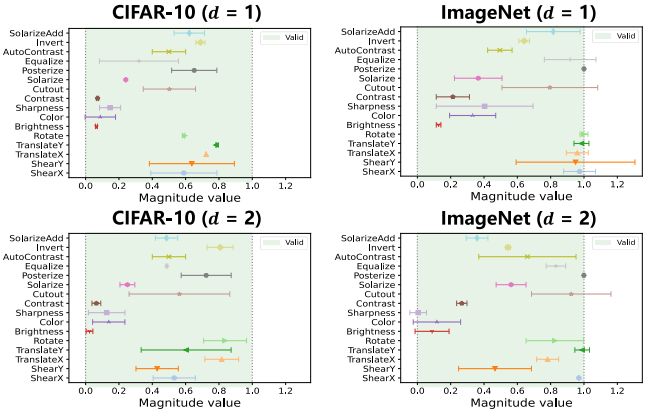

Fig. 3. Visualization of the searched results of mean magnitudes μ and standard deviations σ on CIFAR-10 (WRN-28-10) and ImageNet (ResNet-50). The colored area shows the valid range for sampled magnitudes.

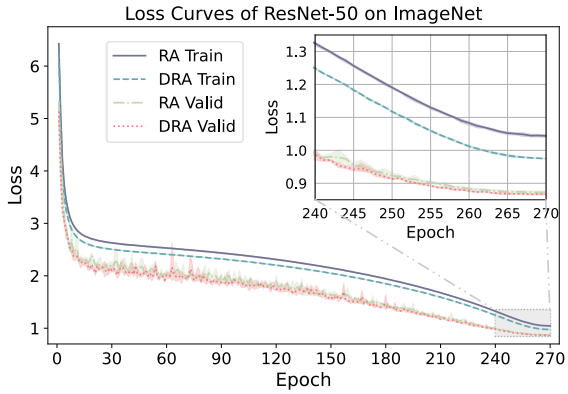

Fig. 4. Training and validation loss curves of ResNet-50 on ImageNet, which are evaluated on three runs. Colored areas show the range of the loss while lines show the average results.

different for the same operator, the same image has many variants that show gradual deformations. These gradual deformations fill the missing points in the original image space, making the space smoother to be learned for the target model. The easier training may explain why DRA has a better performance compared with RA.

DRA slightly increases the performance while reducing the training difficulty, which achieves a good balance between the optimization of accuracy and diversity. It demonstrates a balanced augmentation design between good performance and randomness may exist in an adaptive policy to the target task. We hope our DRA can enlighten future works for better design of data augmentation that can further promote the accuracy, with tricks such as augmentation customization or hard sample mining.

V. Discussion

DRA achieved admirable results compared with RA with only a small search cost. To explore how DRA improves classification performance, we conduct ablation and hyperparameter studies on ImageNet with ResNet-50. We also discuss the influence of KL divergence and proxy task in our DRA in this section. Hyperparameter settings are the same as the ones in Section IV-A if not mentioned.

A. Components of DRA

The components of our DRA can be summarized as four parts: learnable magnitude distributions, learnable selecting weights, usage of operator gradients, and modification of the outer optimization objective in bi-level optimization. The latter two provide a more accurate estimation of the gradient for the learnable policy parameters. To further understand how these components affect the performance, we explore the gradual impact of these components and list the ablation results in Table VI, focusing on the improvement of Top-1 accuracy on ImageNet with ResNet-50. Results are evaluated on the average of three runs.

We find that the proposed components can gradually improve performance, especially for the learnable magnitude distributions and usage of KL divergence. The joint usage of the learnable policy parameters and optimization for gradient estimation obviously contribute to performance improvement.

TABLE VI
ABLATION STUDY OF DRA ON IMAGENET

Component	RA				DRA
Learn μ & σ		√	√	√	√
Learn w			√	√	√
OP Gradient				√	√
KL Loss					√
Top-1	77.91	77.98	78.01	78.09	78.19

TABLE VII
IMPACT OF MAGNITUDE PARAMETERS OF DRA ON IMAGENET

Param						DRA
μ	1.0	Learn	1.0	Learn	1.0	Learn
σ	0.0	0.0	0.1	0.1	Learn	Learn
Top-1	77.97	78.04	77.99	78.04	77.96	78.19

Besides, accurate gradient estimations for differentiable methods also contribute to good results.

B. Magnitude Distribution

We further explore the impact of magnitude distributions that improve performance of DRA. We compare the performance of six cases to explore the detailed impact of magnitude parameters, including fixed magnitudes, learnable magnitudes, magnitudes following fixed distributions, learnable magnitude distributions with a fixed standard deviation, learnable magnitude distributions with a fixed mean, and learnable magnitude distributions. The results are shown in Table VII, which are evaluated on three runs.

Learning both the mean magnitudes μ and the standard deviation of the magnitude distribution σ yields the best performance compared with other settings. Different values of learnable μ indicate the preferences of different operators on the dataset. Meanwhile, different values of learnable σ provide more chances for operators to sample magnitudes m within a range, where the range depends on both the character of the operator and the dataset. Even learning μ alone can slightly improve performance compared with using a fixed μ. However, magnitude following distribution alone does not obviously improve model performance. These results demonstrate that learnable distributions, rather than using magnitudes following distributions, account for major improvement in performance. Another interesting finding is that learnable magnitude distributions with fixed μ and learnable σ even slightly hurt performance compared to the ones with both fixed μ and σ. This is due to random factor ϵ in the reparameterization trick (as shown in Equation (13)) that severely influences σ. The influence becomes obvious when only σ is learnable.

C. KL Divergence

KL divergence is used to measure the differences between the two distributions. It has been adopted in automatic data augmentation to avoid outliers caused by the removal of part of the semantic information when applying heavy augmentations [39]. Inspired by the idea, we also introduce KL divergence to refine the searched policy parameters in our DRA to avoid generating various outliers.

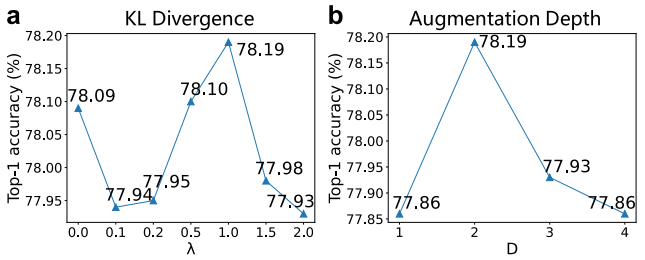

Fig. 5. Impact of some hyperparameters on ImageNet evaluated with ResNet-50. a) The ratio of KL divergence during searching. b) The total number of augmentation layers.

To explore the impact of KL divergence in DRA, we choose a group of λ and evaluate the performance of the models with different λ. Results are evaluated on three runs. As shown in Fig. 5 (a), with the increase of λ, the performance first decreases, then increases, and then decreases, which reaches the peak when $\lambda = 1.0$. We analyze that DRA benefits from the suitable ratio of KL loss, which not only helps the model to learn policy parameters that can train models to deviate from the local optimum, but also avoids severe degradation of augmented images to original ones.

Although more values of λ is worth trying and a careful selection might further increase model performance, we choose $\lambda = 1.0$ in our experiments for simplicity.

D. Augmentation Depth

With more augmentation layers, the transformation of the input image is expected to be more obvious and severe, which will increase the diversity of the input dataset. However, more augmentation layers may also introduce outliers that lose important semantic information and hurt model performance. To explore the impact of augmentation depth for DRA, we conduct experiments with different total augmentation depths D ranging from 1 to 4. Results are illustrated in Fig. 5 (b), which are evaluated on three runs. The performance first increases and then gradually decreases with the increase in D. We analyze that the increase is due to inadequate transformations when the augmentation depth is too shallow ($D = 1$). However, when D goes up to 3, the augmented data will be placed into a complex image space for the model to learn from, which increases the difficulty of the model to converge. Besides, semantic information of the original images might be severely corrupted due to the combo of different operators. Moreover, we also notice that our finding is different from the one reported by Hataya et al. [10], because we find the number of sub-policies in Faster AA is 10, which only has a total of $2^D \times 10$ variants after augmentation. This augmented space is too small for ImageNet, which limits the model to reach better performance.

In our experiments, we select $D = 2$ that not only achieves the best performance compared with other augmentation depth

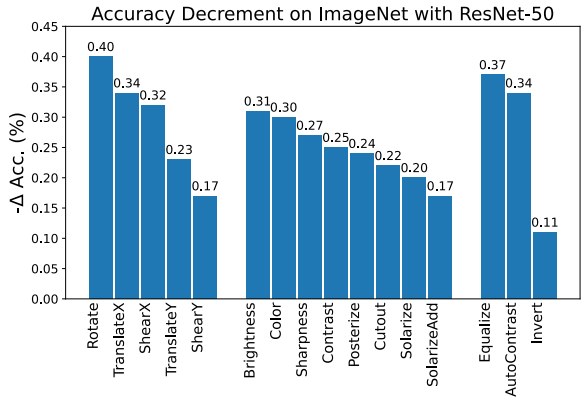

Fig. 6. Decrement of performance on ImageNet using ResNet-50 with each operator removed from the candidate operator set. Results are reported on the average of three runs. The policy parameters are searched from scratch after the removal of operators.

settings, but is also consistent with other differentiable methods for a fair comparison.

E. Operators

Operators transform images into different variants that provide deformations to the input dataset to improve the robustness of the trained models. However, the improper use of operators may hurt performance. As reported by Cubuk et al. [5], not all operators are beneficial to performance. To get a sense of the influence of each operator, we separately remove each operator in the candidate operator set of DRA and evaluated the performance.

In particular, we remove one operator from the candidate operator set and search for new policy parameters on the remaining ones to evaluate the performance decrement. The results are shown in Fig. 6 on the average of three runs. We group the operators into three groups, including geometric transformations (left), color transformations with magnitudes (middle), and color transformations without magnitudes (right). Removing any of the operators causes performance to decrease. We also find *rotation* is the most important to the classification on ImageNet. Besides, transformations in the x-axis contribute more to performance compared with those in the y-axis in geometric transformations. Color transformations with magnitudes contribute similarly to performance. For color transformations without magnitudes, the contributions vary significantly, with *invert* showing poor contribution compared with contrast changing operators.

The results show that ImageNet prefers variants generated from both geometric and color transformations, demonstrating the complexity of the dataset that requires abundant transformations during training for better generalization. Experiments with more operators in the candidate set are worth evaluation in the future, which we believe will further improve performance of DRA on ImageNet.

F. Proxy Task

1) Reasons to Use the Proxy Task: Differentiable automatic augmentations generally use the proxy task to reduce the search cost, which sacrifices the precision of gradient estimation and slightly decreases the performance as reported by Lim et al. [31]. To better estimate the loss in the outer loop of bi-level optimization, RA searches the policy parameters directly on the target task [5]. However, it also introduces heavy constraints to reduce the size of the search space, thus the flexibility is also constrained. The search cost of RA is also limited by the scale of the target dataset, which is difficult to reduce further.

Considering both the advantages and disadvantages, our DRA adopts the proxy task with several tricks for better estimation of the gradient, which strikes a balance between the search cost and performance. The search strategy based on the proxy task works well, thus we believe DRA is practical and worthwhile for wide applications. In the future, we will try to apply DRA directly on the target dataset, which is in theory more precise for gradient optimization.

TABLE VIII
IMPACT OF THE DESIGN OF PROXY TASK FOR DRA EVALUATED ON CIFAR-100

Target Model	Proxy Model	Proxy Data	Acc. (%)
WRN-28-10	WRN-28-10	Random	83.85
		Balanced	83.56
	PyramidNet	Random	83.27
	WRN-40-2		83.56
PyramidNet	PyramidNet	Random	89.17
		Balanced	89.13
	WRN-40-2	Random	89.09

2) Impact of the Proxy Model: DRA adopts the target model as the proxy model to search the optimal augmentation strategy. The idea is intuitive because the search on the target model is expected to have a smaller gap to the target task compared with the search on a smaller proxy model. To explore whether the intuition is correct, we evaluate the impact of proxy models on CIFAR-100. We compare the performance of both WRN-28-10 and PyramidNet+ShakeDrop trained on policies found on themselves and the smaller proxy model WRN-40-2. We also compare the performance of WRN-28-10 trained on the policy found on PyramidNet+ShakeDrop that can be viewed as a larger proxy model. The results are shown in Table VIII.

The usage of both smaller and larger proxy models decreases performance of the target tasks, indicating the selection of the proper proxy model is important for proxy tasks. The intuitive idea that directly using the target model as the proxy model can reduce the gap between proxy tasks and target tasks and promote the final performance.

Besides, the performance gaps between PyramidNet+ShakeDrop with policies searched on different proxy models are smaller than WRN-28-10, indicating that the impact of proxy models on larger models is smaller. This finding may support that DRA is more suitable for smaller models that require more regularization to achieve better performance.

3) Impact of the Balanced Proxy Dataset: Following previous proxy-based automatic data augmentation methods, DRA

randomly samples part of the images from the training dataset of the class-balanced dataset CIFAR as the proxy dataset. For CIFAR-100 that has many classes, the randomly sampled small-scale proxy dataset may be unbalanced. We note that previous works neglect the analysis of whether a randomly sampled dataset or a balanced proxy dataset is better for the proxy task on the balanced target dataset. To further evaluate the impact, we compare the performance of DRA trained on a randomly sampled proxy dataset and a stratified one from the CIFAR-100 training dataset, respectively. Note that the separation of the stratified proxy dataset to two halves for proxy search is random, which is the same as in previous experiments. The results show that a balanced proxy dataset does not benefit the target task on the balanced target dataset. Randomly sampling the proxy dataset is enough to yield satisfactory performance.

G. Influence to Future Works

DRA increases the performance compared with RA. Meanwhile, it reduces the training difficulty, which achieves a good balance between the optimization of accuracy and randomness. It demonstrates that a balanced policy design may exist in adapting the policy to the specific target task. We hope the idea presented in DRA can enlighten future works for better design of data augmentation that can further promote the accuracy.

A specific way that is worthwhile trying is online DRA with techniques such as adversarial learning [59], hard sample mining, or knowledge distillation [39]. With the improvement in controlling the difficulty of generated samples or transforming samples with adjustive policies, DRA may achieve better performance. These prospections are also our future efforts.

Due to the simple design and offline characteristics, transferring DRA to existing training pipelines is easy to achieve with almost no extra training budget, especially on applications that use RA as an augmentation strategy. Besides, DRA may promote model performance in cases with limited calculation sources or large cost for new data collection, such as mobile computing and medical image analysis [60]. It shows wide prospectives to the community.

VI. Conclusion

In this work, we focus on the inflexibility and large search cost of RandAugment (RA), and propose Differentiable RandAugment (DRA), a method that can automatically learn the selecting weights and magnitude distributions of different transformations. DRA generally outperforms RA with a small search cost. It adopts the search space of RA and models the magnitude of each transformation following a learnable normal distribution, and uses relaxation and approximation to differentiate learnable policy parameters. We also introduce operator gradients and KL divergence to reduce the bias in gradient estimation. Experiments on several datasets and tasks demonstrate the efficiency and effectiveness of DRA, especially on the classification of ImageNet. Our DRA is one of the few to outperform RA on ImageNet under a similar training budget. We believe this framework can be integrated into more computer vision tasks, serving as a baseline for subsequent research.

Acknowledgment

The authors would like to thank Weichen Yu, Qinglin Zhu, Jiyang Guan, and Xinjian Wu for their help.

References

[1] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," 2017, *arXiv:1708.04552*.

[2] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," 2017, *arXiv:1710.09412*.

[3] D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan, "AugMix: A simple data processing method to improve robustness and uncertainty," 2019, *arXiv:1912.02781*.

[4] S. Yun, D. Han, S. Chun, S. J. Oh, Y. Yoo, and J. Choe, "CutMix: Regularization strategy to train strong classifiers with localizable features," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6023–6032.

[5] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2020, pp. 702–703.

[6] J. Qin, J. Fang, Q. Zhang, W. Liu, X. Wang, and X. Wang, "ResizeMix: Mixing data with preserved object information and true labels," 2020, *arXiv:2012.11101*.

[7] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "AutoAugment: Learning augmentation strategies from data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 113–123.

[8] Y. Li, G. Hu, Y. Wang, T. Hospedales, N. M. Robertson, and Y. Yang, "DADA: Differentiable automatic data augmentation," 2020, *arXiv:2003.03780*.

[9] A. Liu, Z. Huang, Z. Huang, and N. Wang, "Direct differentiable augmentation search," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 12219–12228.

[10] R. Hataya, J. Zdenek, K. Yoshizoe, and H. Nakayama, "Faster AutoAugment: Learning augmentation strategies using backpropagation," in *Proc. Eur. Conf. Comput. Vis.* Glasgow, U.K.: Springer, 2020, pp. 1–16.

[11] X. Zhang, Q. Wang, J. Zhang, and Z. Zhong, "Adversarial AutoAugment," 2019, *arXiv:1912.11188*.

[12] C. Lin et al., "Online hyper-parameter learning for auto-augmentation strategy," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6579–6588.

[13] S. G. Müller and F. Hutter, "TrivialAugment: Tuning-free yet state-of-the-art data augmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 774–782.

[14] T. C. LingChen, A. Khonsari, A. Lashkari, M. R. Nazari, J. S. Sambee, and M. A. Nascimento, "UniformAugment: A search-free probabilistic data augmentation approach," 2020, *arXiv:2003.14348*.

[15] R. Hataya, J. Zdenek, K. Yoshizoe, and H. Nakayama, "Meta approach to data augmentation optimization," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2022, pp. 2574–2583.

[16] S. Mounsaveng, I. Laradji, I. B. Ayed, D. Vazquez, and M. Pedersoli, "Learning data augmentation with online bilevel optimization for image classification," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2021, pp. 1691–1700.

[17] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jegou, "Training data-efficient image transformers & distillation through attention," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 10347–10357.

[18] Z. Liu et al., "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 10012–10022.

[19] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[20] O. Russakovsky et al., "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[21] T.-Y. Lin et al., "MicroSoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.* Zürich, Switzerland: Springer, 2014, pp. 740–755.

[22] P. Chen, S. Liu, H. Zhao, and J. Jia, "GridMask data augmentation," 2020, *arXiv:2001.04086*.

[23] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 7, pp. 13001–13008.

[24] H. Zhang, Z. Xu, X. Han, and W. Sun, "Data augmentation using bit-plane information recombination model," *IEEE Trans. Image Process.*, vol. 31, pp. 3713–3725, 2022.
[25] J.-B. Grill et al., "Bootstrap your own latent—A new approach to self-supervised learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 21271–21284.
[26] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 9729–9738.
[27] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 1597–1607.
[28] I. Kostrikov, D. Yarats, and R. Fergus, "Image augmentation is all you need: Regularizing deep reinforcement learning from pixels," 2020, *arXiv:2004.13649*.
[29] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," Presented at the NIPS Workshop Deep Learn. Unsupervised Feature Learn., 2011.
[30] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2016, *arXiv:1611.01578*.
[31] S. Lim, I. Kim, T. Kim, C. Kim, and S. Kim, "Fast AutoAugment," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–11.
[32] D. Ho, E. Liang, X. Chen, I. Stoica, and P. Abbeel, "Population based augmentation: Efficient learning of augmentation policy schedules," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2731–2741.
[33] X. Wang, X. Chu, J. Yan, and X. Yang, "DAAS: Differentiable architecture and augmentation policy search," 2021, *arXiv:2109.15273*.
[34] K. Zhou et al., "DHA: End-to-end joint optimization of data augmentation policy, hyper-parameter and architecture," 2021, *arXiv:2109.05765*.
[35] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," 2018, *arXiv:1806.09055*.
[36] X. Dong and Y. Yang, "Searching for a robust neural architecture in four GPU hours," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1761–1770.
[37] W. Grathwohl, D. Choi, Y. Wu, G. Roeder, and D. Duvenaud, "Back-propagation through the void: Optimizing control variates for black-box gradient estimation," 2017, *arXiv:1711.00123*.
[38] R. Wightman, H. Touvron, and H. Jégou, "ResNet strikes back: An improved training procedure in timm," 2021, *arXiv:2110.00476*.
[39] L. Wei, A. Xiao, L. Xie, X. Zhang, X. Chen, and Q. Tian, "Circumventing outliers of AutoAugment with knowledge distillation," in *Proc. Eur. Conf. Comput. Vis.* Glasgow, U.K.: Springer, 2020, pp. 608–625.
[40] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with Gumbel–Softmax," 2016, *arXiv:1611.01144*.
[41] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," 2016, *arXiv:1611.00712*.
[42] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," 2013, *arXiv:1308.3432*.
[43] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, *arXiv:1312.6114*.
[44] E. Riba, D. Mishkin, D. Ponsa, E. Rublee, and G. Bradski, "Kornia: An open source differentiable computer vision library for PyTorch," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2020, pp. 3674–3683.
[45] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–12.
[46] M. Abadi et al., "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Oper. Syst. Design Implement. (OSDI)*, 2016, pp. 265–283.
[47] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.
[48] J. Song, Y. Chen, J. Ye, and M. Song, "Spot-adaptive knowledge distillation," *IEEE Trans. Image Process.*, vol. 31, pp. 3359–3370, 2022.
[49] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2980–2988.
[50] P. Ren et al., "A comprehensive survey of neural architecture search: Challenges and solutions," *ACM Comput. Surv.*, vol. 54, no. 4, p. 76, 2021.
[51] S. Zagoruyko and N. Komodakis, "Wide residual networks," 2016, *arXiv:1605.07146*.
[52] D. Han, J. Kim, and J. Kim, "Deep pyramidal residual networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5927–5935.
[53] Y. Yamada, M. Iwamura, T. Akiba, and K. Kise, "Shakedrop regularization for deep residual learning," *IEEE Access*, vol. 7, pp. 186126–186136, 2019.
[54] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[55] S. R. Bulo, L. Porzi, and P. Kontschieder, "In-place activated BatchNorm for memory-optimized training of DNNs," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5639–5647.
[56] M. Tan and Q. Le, "EfficientNetV2: Smaller models and faster training," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 10096–10106.
[57] I. Bello et al., "Revisiting ResNets: Improved training and scaling strategies," Presented at the Adv. Neural Inf. Process. Syst., 2021.
[58] K. He, R. Girshick, and P. Dollár, "Rethinking ImageNet pre-training," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 4918–4927.
[59] C. Xie, M. Tan, B. Gong, J. Wang, A. L. Yuille, and Q. V. Le, "Adversarial examples improve image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 819–828.
[60] J. Ying et al., "Two fully automated data-driven 3D whole-breast segmentation strategies in MRI for MR-based breast density using image registration and U-Net with a focus on reproducibility," *Vis. Comput. Ind., Biomed.*, vol. 5, no. 1, p. 25, Oct. 2022.

Anqi Xiao is currently pursuing the Ph.D. degree with the CAS Key Laboratory of Molecular Imaging, Beijing Key Laboratory of Molecular Imaging, Institute of Automation, Chinese Academy of Sciences, under the supervision of Prof. Zhenhua Hu. Her research interests include deep learning and its application in medical images.

Biluo Shen received the master's degree from the Beijing Key Laboratory of Molecular Imaging, Institute of Automation, Chinese Academy of Sciences. His research interests include deep learning, computer vision, and neural architecture search.

Jie Tian (Fellow, IEEE) received the Ph.D. degree from the Institute of Automation, Chinese Academy of Sciences, Beijing, China. He is currently the Chief Scientist of the CAS Key Laboratory of Molecular Imaging, Beijing Key Laboratory of Molecular Imaging, Institute of Automation, Chinese Academy of Sciences, and also with the Beijing Advanced Innovation Center for Big Data-Based Precision Medicine, School of Engineering Medicine, Beihang University. His research interests include optical multimodality molecular imaging technology development and artificial intelligence (AI) in radiomics.

Zhenhua Hu (Senior Member, IEEE) received the Ph.D. degree from Xidian University, Xi'an, Shaanxi, China. She is currently a Professor with the CAS Key Laboratory of Molecular Imaging, Beijing Key Laboratory of Molecular Imaging, Institute of Automation, Chinese Academy of Sciences. Her research interests include Cerenkov luminescence imaging and tomography, and fluorescence-guided surgeries.