Local Patch AutoAugment With Multi-Agent Collaboration

Shiqi Lin, Tao Yu®, Ruoyu Feng, Xin Li®, Student Member, IEEE, Xiaoyuan Yu, Lei Xiao, and Zhibo Chen®, Senior Member, IEEE

Abstract—Data augmentation (DA) plays a critical role in improving the generalization of deep learning models. Recent works on automatically searching for DA policies from data have achieved great success. However, existing automated DA methods generally perform the search at the image level, which limits the exploration of diversity in local regions. In this paper, we propose a more fine-grained automated DA approach, dubbed Patch AutoAugment, to divide an image into a grid of patches and search for the joint optimal augmentation policies for the patches. We formulate it as a multi-agent reinforcement learning (MARL) problem, where each agent learns an augmentation policy for each patch based on its content together with the semantics of the whole image. The agents cooperate with each other to achieve the optimal augmentation effect of the entire image by sharing a team reward. We show the effectiveness of our method on multiple benchmark datasets of image classification, fine-grained image recognition and object detection (e.g., CIFAR-10, CIFAR-100, ImageNet, CUB-200-2011, Stanford Cars, FGVC-Aircraft and Pascal VOC 2007). Extensive experiments demonstrate that our method outperforms the state-of-the-art DA methods while requiring fewer computational resources.

Index Terms—Automatic augmentation, data augmentation, multi-agent reinforcement learning, reinforcement learning.

I. INTRODUCTION

Data Augmentation (DA) has been widely used to alleviate the overfitting risk in training deep neural networks by appropriately enriching the diversity of training data [1], [2], [3], [4], [5]. Notable DA methods improve the performance and robustness of neural networks, such as rotation, Mixup [6] and Cutmix [7]. However, these approaches are typically handcrafted and require human prior knowledge, which causes weak transferability of DA across different datasets. To relieve the dependence on manual design and further explore more adaptive augmentation, AutoAugment (AA) [8], as a new DA paradigm, is proposed to automate the search of the optimal DA policies (i.e., DA operation, probability and magnitude) from the training dataset. To be specific, AA trains a proxy model with the augmentation policy generated by a controller, which is updated through reinforcement learning using validation accuracy as the reward signal. In spite of the superior performance of AA, its optimization procedure is computationally intensive due to the need to evaluate thousands of policies. Therefore, to reduce the search costs, multiple automated DA approaches [9], [10], [11], [12], [13] have been proposed. For example, [9] employs density matching as a search method to accelerate the policy search, and [12] introduces adversarial learning to organize the target network training and augmentation policy search in an online manner.

Yet the aforementioned automated DA methods all search for policies at the image level. They ignore the exploration of diversity in local regions, which may result in insufficient diversity of the dataset and limit the benefits of DA [2]. In addition, due to this coarse-grained augmentation, they may damage key semantic information and introduce ambiguity into the training process (see a simple example in Fig. 1 row 2, column 2). With those in mind, it is necessary to automatically search for the optimal augmentation policies for different regions by taking regional diversity into account. One straightforward idea is to directly

Fig. 1. Illustration of different automated augmentation policies. We show the examples processed by image-wise automated DA, e.g., AutoAugment (middle row) and processed by our proposed Patch AutoAugment (bottom row).
apply image-wise automated DA methods in different regions. However, such an intuitive solution ignores the contextual relationship between regions, which may lead to the non-globally optimal effectiveness of DA policies across the entire image. In addition, it may encounter an extremely high computational cost due to the need of optimizing multiple policies for regions respectively.

In this paper, we propose a new approach, named Patch AutoAugment (PAA) (see the last row of Fig. 1), to address the above-mentioned problems. We first divide an image into a grid of patches to increase the flexibility of representations for different regions and take “patch” as the basic control unit. Then, we model the search for the augmentation policies of patches as a fully cooperative multi-agent task, and we leverage a multi-agent reinforcement learning (MARL) algorithm where agents cooperatively learn the policies. Specifically, based on the content of each patch and the semantics of the entire image, the agent searches for a policy in terms of choosing which transformation to apply out of the pre-defined DA operations. To encourage our policy networks to adaptively learn the beneficial augmentation policies for the target network, we use the feedback of the target network as the team reward signal to guide the policy networks to learn on the fly. All agents wind up benefiting from two mechanisms (i.e., parameter sharing and centralized training with decentralized execution) in MARL. In this way, all agents collaboratively and parallely learn policies to further achieve the joint optimal DA policy across the whole image and alleviate the computational cost.

In summary, our contributions can be summarized as follows: 1) We pinpoint that exploring diversity in local regions is important for automated learned DA approaches. To the best of our knowledge, we are the first to propose a more fine-grained automated DA approach, that searches for the optimal policies for the patches according to the content of the patch and the semantics of the entire image. 2) To further achieve the joint optimal policy across the image, we model the DA policy search of patches as a fully cooperative multi-agent task, and adopt a multi-agent reinforcement learning algorithm for Patch AutoAugment by considering the contextual relationship between the patches. 3) Our visualization results provide some insights to the DA community on which augmentation operation should be chosen for patches with different content during the whole training process.

II. RELATED WORK

A. Data Augmentation

Despite the remarkable performance of deep learning models in computer vision tasks, they often suffer from overfitting. Data augmentation (DA) as an effective technique has been proved to improve the generalization ability of deep learning models [1]. Previous works [2], [14] indicate that the main benefit of DA arises from increasing the diversity of images. Popular techniques, such as rotation, flipping, color transformation, have been performed as commonly used augmentation methods. Recently, thanks to the skillful design of human experts, many DA methods (e.g., Cutout [15], Mixup [6], CutMix [7]) have been proposed and show significant performance. However, these manually designed methods require additional human prior knowledge on the dataset and sometimes they are limited to certain datasets and target tasks. Naturally, the methods automatically finding DA from data have emerged to overcome the limitations of dependence on cumbersome manual exploration. Some works use generative adversarial networks (GANs) to directly generate training data [16], [17], [18] proposes a multi-label data augmentation method based on Wasserstein generative adversarial networks to expand multi-label datasets. [19] designs a cover augmentation network, which intelligently adds noise to the original cover to generate the augmented cover.

Furthermore, recent studies aim to automate the search for augmentation policies that choose the optimal transformations for training images. AutoAugment (AA) [8] adopts a controller to generate an augmentation policy that is used to train a proxy network, then gets the validation accuracy as the reward signal to update the controller using reinforcement learning. Unfortunately, the evaluation of thousands of policies makes AA computationally expensive. Therefore, multiple automated DA approaches focus on reducing the huge complexity and have achieved great progress. For example, PBA [10] employs hyperparameter optimization, Fast AA [9] uses a density matching algorithm, Adv AA [12] proposes an adversarial framework to jointly optimize the target network and the augmentation network, and DADA [20] and DADAS [21] target relaxing the discrete DA policy selection to a differentiable problem. [22] proposes an Automatic Model Augmentation (AutoMA) approach to find a strong model augmentation policy for transferable adversarial attacks. Besides, RandAugment [14] removes the separate search on a proxy task in the offline automated DA methods (e.g., AA) while also outperforming them. However, these automated DA methods perform the search at the image level, i.e., they use the same policy on the whole image. It inevitably ignores the diversity of different regions in an image, which limits the diversity of data increased by data augmentation, and sometimes causes damage to critical semantic information. In contrast, our method takes diversity in local regions and contextual relationships into account to search for augmentation policies for multiple regions.

Our proposed method is conceptually orthogonal to most region-based DA methods where DA transformations are performed in a non-automated way. For example, RandomErasing [23], CutMix [7] perform cropping or replacement operation on a randomly selected rectangle region, Hard chips [24] extracts the object patches from an object pool and pastes them to an image to train the detector, and SECOND [25] performs DA by pasting the objects into the 3D point cloud. Some works further exploit class activation map [26] or saliency map [27] to select representative regions which are augmented by a randomly selected DA operation (e.g., SaliencyMix [28], SnapMix [29], AugMix [30], KeepAugment [31] and TrivialAugment [32]). Yet our proposed method automatically searches for augmentation transformations based on the given datasets and tasks.
B. Multi-Agent Reinforcement Learning

The most significant characteristic of MARL is the cooperation between agents [33], [34], [35] which is distinct from directly applying reinforcement learning (RL) algorithms to multi-agent systems. Due to the limited observation and action of a single agent, cooperation is necessary in the reinforced multi-agent system to achieve the common goal. Compared with independent agents, cooperative agents can improve the efficiency and robustness of the model [36], [37], [38]. Many vision tasks use MARL to interact with the public environment to make decisions, with the goal of maximizing the expected total return of all agents, such as image segmentation [39], [40], [41], image processing [42].

III. PROPOSED METHOD

As mentioned above, we aim to search augmentation policies for the patches to explore more augmentation diversity. Learning the optimal DA policies often needs to interact with the target network, which aligns with the nature of RL (i.e., RL agents learn to make decisions by interacting with the environment [43], [44], [45], [46]). In addition, it is of great necessity to take the patch content and the contextual relationship between patches into account. Therefore, we use the multi-agent reinforcement learning (MARL) algorithm. Specifically, the search for policy is based on the content of the patch together with the semantics of the image, and policies are encouraged to coordinate to achieve the joint optimal DA policy across the whole image. In this section, we first describe the preliminaries of MARL, and then elaborate on our augmentation policy formulation and modeling. Furthermore, we summarize the framework of Patch AutoAugment.

A. Preliminaries of Multi-Agent Reinforcement Learning

We first introduce the preliminaries of reinforcement learning (RL). RL models the decision-making problem as a Markov decision process (MDP) which is presented with a tuple \( \langle S, A, P, R, \gamma, T \rangle \). In the RL framework, given the state \( s \in S \), the agent takes an \( a \in A \) according to its policy \( \pi(a|s) : S \times A \to [0, 1] \) and then receives a reward \( r : S \times A \to \mathbb{R} \). The environment moves to the next state with a transition function denoted as \( P : S \times A \times S \to [0, 1] \). \( \gamma \in (0, 1) \) is a discount factor and \( T \) is a time horizon. The agent aims to maximize the long-term reward \( R \) over \( T \) steps to learn the optimal policy \( \pi^* \).

Furthermore, multi-agent reinforcement learning (MARL) considers a group of \( N \) agents, denoted as \( \mathcal{N} \), operating cooperatively in a shared environment towards a common goal. It can be formulated as a multi-agent MDP (MAMDP) [47] represented with a tuple \( \langle S, \{O_i\}_{i=1}^N, \{\mathcal{A}_i\}_{i=1}^N, P, R, \gamma, T \rangle \). Here, \( S \) describes the shared state space and \( \{O_i\}_{i=1}^N \) is a set of observations for agents. In the MARL configuration, each agent receives a private observation correlated with part of state, i.e., \( o_i : S \to O_i \). According to the global state \( s \) and its observation \( o_i \), the agent takes its action \( a_i \in \mathcal{A}_i \) based on its policy \( \pi_i(a_i|o_i, s) \), \( \mathcal{A}_i \) is the action space for the \( i \)-th agent and \( \mathcal{A} = \{\mathcal{A}_1, \ldots, \mathcal{A}_N\} \) denotes the joint action space. Then, the joint action \( a = a_1 \times \cdots \times a_N \) produces the next state according to transition function \( P : S \times \mathcal{A} \times S \to [0, 1] \) and the environment gives the team reward \( r : S \times \mathcal{A} \to \mathbb{R} \) to agents \( \mathcal{N} \).

The objective of each agent is to maximize the total cumulative reward \( R \) to cooperatively learn the globally optimal policy \( \pi^* = \{\pi_1, \ldots, \pi_N\} \) that consists of the optimal policy of each agent. In this paper, as mentioned before, we employ MARL to search for the optimal policy for each patch, and achieve the optimal effectiveness of DA across the entire image to further improve the performance of the target network as much as possible.

B. Patch AutoAugment

In our proposed Patch AutoAugment (PAA), we formulate the task of policy search for the patches as a cooperative multi-agent decision-making problem and adopt multi-agent reinforcement learning (MARL) to solve it. In the following, we clarify the detailed formulation (i) the state, observation and action modeling for the policy, (ii) an effective team reward function design and (iii) the detailed MARL algorithm for policy learning) of Patch AutoAugment.

1) Policy Modeling: As illustrated in Fig. 2, given the original input batch \( x \) and the corresponding label \( y \) (i.e., \( \{x, y\}_{i=1}^b \) and \( b \) is the batch size), we divide an image \( x_j \) into \( N \) equal-sized and non-overlapping patches, denoted as \( x_j = \{P_j\}_{i=1}^N \), where \( P_j^i \) is the \( i \)-th patch of the image \( x_j \). In our proposed method, we aim to search the augmentation policy for each patch. Therefore, in MARL formulation, the augmentation policy of the patch \( P_j^i \) is controlled by an agent \( i \in \mathcal{N} \), and we detail the state, observation and action for the augmentation policy as below.

2) State: As aforementioned, the selection of augmentation operation for a patch is closely bound up with the contextual relationship between regions. Therefore, the augmentation policy needs to perceive the image semantics and we take the deep features of the whole image extracted by a backbone (e.g., ImageNet pre-trained ResNet-18 [48]) as the global state \( s \) which is visible to all agents. We analyze the impact of feature extractor initialization in F.

3) Observation: Apart from capturing state (i.e., the global information), the agent only uses their own observation (i.e., the local information) which is invisible to other agents. The augmentation policy seeks to choose the augmentation operation based on the content of the patch, and the observation is generally part of the state. Considering all these factors, we utilize the deep features of the \( i \)-th patch as observation \( o_i \), which are extracted by the same backbone as the state extractor.

4) Action: The augmentation policy is responsible for choosing which transformations to apply from pre-defined operations. Following the previous automated DA methods [8], [9], we define fifteen operation functions (i.e., ShearX/Y, TranslateX/Y, Rotate, Invert, Equalize, Solarize, Posterize, Contrast, Color, Brightness, Sharpness, RandomErasing, Cutout, Mixup [6], Cutmix [7]) to construct the action space \( A \). The operation details are shown in Table I. Given the state \( s \) and the observation \( o_i \), according to policy \( \pi_i(a_i|o_i, s) \), each agent \( i \) determines...
Fig. 2. Framework of Patch AutoAugment (PAA). We divide an image into a grid of patches and assign an agent to each patch to select the optimal augmentation operation according to the patch content together with the whole image semantics. The agents cooperate with each other to achieve the joint optimal DA results by sharing a team reward through multi-agent reinforcement learning (MARL). Specifically, each agent outputs $O_i(\cdot; p_i; m_i)$ operation performed on patch $i$, which includes two parameters: probability of calling the operation $p_i$ and the magnitude $m_i$. Note that our proposed method PAA is co-trained with the target network, obviating the need for re-training the target network after learning the sample selection strategy.

| Operation | Description | Magnitudes |
|-----------|-------------|------------|
| Brightness | Adjust the brightness of the patch. A magnitude=0 gives a black patch, whereas magnitude=1 gives the original patch. | $\{0.05, 0.95\}$ |
| Contrast | Control the contrast of the patch. A magnitude=0 gives a gray patch, whereas magnitude=1 gives the original patch. | $\{0.5, 0.95\}$ |
| CutMix | Replace this patch with another patch (selected at random from the patches which are also performed CutMix). | - |
| Cutout | Set all pixels in this patch to the average value of the patch. | - |
| Invert | Invert the pixels of the patch. | - |
| Mixup | Linearly add the image with another image (selected at random from the patches which are also performed Mixup). $\lambda \sim \text{Beta}(1, 1)$ | - |
| Posterize | Reduce the number of bits for each pixel to magnitude bits. $\lambda=3$ | - |
| Solarize | Invert all pixels above a threshold value of magnitude. $\lambda=0.1$ | - |
| RandomErasing | Erase a random rectangle region in a patch. $\{0.09, 0.36\}$, $\{0.5, 2.0\}$ | - |
| Rotation | Rotate the patch magnitude degrees. $\lambda=30.0$ | - |
| Sharpness | Adjust the sharpness of the image. $\lambda=0.5$ | - |
| Shear(X/Y) | Shear the image along the horizontal or vertical axis with rate $\lambda$ $\lambda=-30.0$ | - |
| Translate(X/Y) | Translate the patch in the horizontal or vertical direction by absolute fraction of patch length. $\{0.4, 0.4\}$ | - |
| Color | Adjust the color balance of the image. $\text{hue}=0.3, 0.3$ | - |
| Equalize | Equalize the image histogram. | - |

An action $a_i(\cdot) \in A$, which is the operation performed on the patch. Each operation has two hyperparameters: probability $p$ and magnitude $m$. In order to dramatically and effectively reduce the action space for the augmentation policy, similar to [9], [10], [14], we take a fixed probability and magnitude schedule. Among them, the probability of applying the operation is sampled from the uniform distribution (i.e., $p_i \sim U(0, 1)$). Following [10], we employ the same linear scale as the magnitude schedule. In summary, the processed $i$-th patch in image $x_j$ is denoted as $P_i^j = a_i(P_i^j)$ with the probability $p_i$ and the magnitude of $a_i(\cdot)$ is $m_i$, otherwise $P_i^j = P_i^j$. Note that most operations are label-invariant, except Mixup [6] and CutMix [7], which are label-disturbing operations that combine different patches as well as their labels. We take Mixup as an example, then $P_i^j = \lambda P_i^j + (1-\lambda)P_i^t$ with probability $p_i$, where $P_i^j$ is a patch from another image $x_t, t \neq j, \lambda \sim \text{Beta}(\alpha, \alpha)$, for $\alpha \in (0, \infty)$, and the one-hot label is modified as $\tilde{y}_j = \frac{\lambda}{N} y_j + \frac{(1-\lambda)}{N} \tilde{y}_t$. Once all patches are processed by the corresponding operations chosen by the augmentation policies, we obtain the image augmented by our PAA, denoted as $\tilde{x}_j = \{\tilde{P}_1^j, \ldots, \tilde{P}_N^j\}$, and the final label $\tilde{y}_j$.  

5) Reward Function: The reward function is of importance to guide the agents to learn so that they follow desired behaviors. The previous work, AdvAA [12], attempts to increase the training loss of the target network to generate harder augmentation policies and explore the weakness of the target network. Inspired by AdvAA, we reformulate the reward design appropriately under our configuration. Our Patch AutoAugment only trains one target network. Specifically, we input the original batch into the target network to get $l(\phi(x), y)$ without back propagation. And
we input the augmented batch into the target network to get \( l(\phi(\tilde{x}), \tilde{y}) \) and use it to update the parameters of the target network. We take their difference as the reward to for updating the policy network. In our proposed PAA, the common objective of all agents is to improve the performance of the mainstream target task through enhancing the benefits of DA. Therefore, we compare the feedback of the target network \( \phi(\cdot) \) on the augmented data processed by our proposed PAA \( \tilde{x} \) with the original data \( x \) and take their difference on the training losses as the reward for the policy in MARL, as in (1):

\[
    r = l(\phi(\tilde{x}), y) - l(\phi(x), y),
\]

where \( x \) and \( y \) denote raw inputs and labels in supervision tasks. In our PAA model, all agents are encouraged to cooperate to achieve the common goal, thus we adopt the team reward function design (i.e., the shared reward mechanism in MARL) as (1) for all agents to make the joint augmentation policy achieve the optimal effectiveness.

6) Policy Learning: Here, we introduce the training for the augmentation policies mentioned above. Considering that the action space is discrete, we adopt the multi-agent Advantage Actor-Critic algorithm to learn the augmentation policies and encourage the coordination behaviors. In MARL, the framework of centralized training with decentralized execution [49], [50] is generally adopted. More concretely, each agent \( i \) has an actor that learns the discrete policy \( \pi_i(a_i|s) \) and shares a common critic which aims to estimate the value of global state \( V^\pi(s) \). We use the centralized critic to train decentralized actors. Here, we reformulate it appropriately for our task. We model the centralized action-value Q function that takes the actions of all agents in addition to state information \( s \) and outputs the Q-value for the team, formulated as:

\[
    Q^\pi(s, a) = \mathbb{E}_r[R_t|s, a_1, \ldots, a_N],
\]

where \( a \) is the joint action of all agents \( a = \{a_1, \ldots, a_N\} \) and \( R_t = \sum_{t=0}^\infty \gamma^t r_{t+1} \) is the long-term discounted reward. Then, the advantage function on the augmentation policy is given as follows:

\[
    A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s).
\]

The critic network is a Multi-layer Perceptron (MLP). And we use \( \varphi \) to denote its parameters. We take the square value of the advantage function \( A^\pi \) as the loss function to update \( \varphi \):

\[
    L(\varphi) = (A^\pi(s, a))^2.
\]

The global policy network is a fully convolutional network (FCN), denoted as \( \theta \). Besides, to further achieve the ability to cooperate, similar to [49], [50], the parameters of the actor networks of all agents are shared. The loss function for updating \( \theta \) is defined as:

\[
    L(\theta) = -\log \pi_\theta(a|s)A^\pi(s, a).
\]

7) Framework Summary: We summarize the overall framework of our proposed PAA. As shown in Fig. 2, PAA first divides an image into a grid of patches. Then, we use a feature extractor to obtain the deep features of the whole image as the state. Each agent draws its individual observation, i.e., the deep features of the patch. According to the global state (i.e., the semantics of the entire image) and the local observation (i.e., the content of the patch), the actor networks output the augmentation operations of patches to further construct the joint operation map performed on the whole image. The augmented images are processed by our proposed PAA and then we input them into the target task network for parameters updating. Moreover, the feedback of the target network is used as the team reward signal to update the policy network.

C. Discussion

1) How to Ensure the Convergence of \( L(\theta) \): The policy increases the log-likelihood of the action (see (5)) through the policy gradient (PG). And the function in PG is continuous and convex. Therefore, the policy in PAA can be improved through this optimization paradigm. In addition, according to the visualization of the policy (see Fig. 3), the policy has converged.

2) How Does the Reward Prevent Agents From Learning bad Augmentations: Image-wise AutoAugment (AA) adopts a control to generate an augmentation policy that is used to train a proxy network, and gets the validation accuracy as the reward signal to update the controller using reinforcement learning. AutoAugment repeatedly trains the light-weight proxy network for the evaluation of every candidate DA policies using validation accuracy, which is computationally expensive (15000 gpu hours on ImageNet). Therefore, we consider using the training loss to improve the efficiency. However, directly exploiting the loss of augmented samples does encourage all agents to corrupt key semantic information to maximize training loss. We design our reward function introducing another loss as reference.

We take the difference between the training loss of the original batch and the loss of the augmented batch as the reward, instead of using only one of two losses. The two losses reflect the training status of the target network. If all agents are encouraged to lose important semantic information to obtain high training loss \( l(\phi(\tilde{x}), \tilde{y}) \), the target network updated using the augmented images without key semantics may be misled which causes the loss of the original input batch \( l(\phi(x), y) \) is still very high and the difference between the two (i.e. reward) is actually small. For example, in extreme cases, an all-black augmented image with no information will be generated to maximize the loss of augmented images. However, the loss of original images obtained by the target network trained with such augmented images is still very large, resulting in a small difference between the two loss. MARL which aims to maximize the reward, reduces selections of bad DA to prevent the agents from bad DA and semantics loss.

This mutual influence between the two losses will alleviate the damage of key semantics and the introduction of ambiguity during training. The constraint relationship can effectively control hard mining without losing key semantic information. As shown in Fig. 3, Cutout is chosen for the relatively unimportant patches, which does not hurt the semantics of the image and reduces the impact of the unexpected features.
3) How to Achieve Cooperation Between Multi-Agents in MARL: 1) the agent takes its action $a_i \in A_i$ based on its policy $\pi_i(a_i|o_i, s)$ and $s$ describes the shared state space. That is, each agent makes a decision considering the global state and is not isolated. 2) the parameters of the actor networks of all agents are shared to further achieve the ability to cooperate. Although they share parameters, each agent’s policy is not identical. When determining an action, each agent’s policy relies on different activations of the global policy network. 3) all agents share the team reward function design to cooperate to achieve the common goal.

IV. EXPERIMENTS

A. Datasets

We evaluate Patch AutoAugment (PAA) on the following datasets: CIFAR-10, CIFAR-100, ImageNet [51] and three fine-grained image recognition datasets (CUB-200-2011 [52], Stanford Cars [53], FGVC-Aircraft [54]) and Pascal VOC. To be specific, both CIFAR-10 and CIFAR-100 have 50,000 training examples. Each image of size $32 \times 32$ belongs to one of 10 categories. ImageNet dataset has about 1.2 million training images and 50,000 validation images with 1000 classes. Original ImageNet data have different sizes and we resize them to $224 \times 224$. In addition, we evaluate the performance of our proposed PAA on three standard fine-grained object recognition datasets, CUB-200-2011 [52] consists of 6,000 train and 5,800 test bird images distributed in 200 categories. Stanford Cars [53] contains 16,185 images in 196 classes. The FGVC-Aircraft dataset contains 10,200 images of aircraft, with 100 images for each of 102 different aircraft model variants, most of which are airplanes. The image size in the above three datasets is $448 \times 448$. The Pascal VOC dataset consists of a set of images and each image has an annotation file giving a bounding box and object class label for each object in one of the twenty classes present in the image. The data has been split into 50% for training and 50% for testing. In total there are 9,963 images, containing 24,640 annotated objects.

B. Implementation Details

We detail various target models hyperparameters (e.g., batch size, learning rate and training epochs) on CIFAR-10, CIFAR-100, ImageNet, CUB-200-2011, Stanford Cars and FGVC-Aircraft in Table III. We do not specifically tune these hyperparameters, and all of these are consistent with previous works [8], [9], [55], [56], [57]. In addition, the time horizon $T$ determines the times of augmentations performed sequentially on a patch.

---

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
TABLE III
THE HYPERPARAMETERS OF VARIOUS TARGET MODELS ON CIFAR-10, CIFAR-100, ImageNet, CUB-200-2011, Stanford Cars and FGVC-Aircraft. LR REPRESENTS LEARNING RATE OF THE TARGET NETWORK, WD REPRESENTS WEIGHT DECAY, AND LD REPRESENTS LEARNING RATE DECAY METHOD. IF LD IS MULTISTEP, WE DECAY THE LEARNING RATE BY 10-FOLD AT EPOCHS 30, 60, 90 ETC. ACCORDING TO LR-STEP. LR-A2C REPRESENTS THE LEARNING RATE OF THE AUGMENTATION MODEL.

| Dataset          | Model               | Batch Size | LR   | WD  | LD    | LRstep | LR-A2C | Epoch |
|------------------|---------------------|------------|------|-----|-------|--------|--------|-------|
| CIFAR-10         | WRN-28-10           | 128        | 0.1  | 1e-4| cosine| -      | 1e-3   | 200   |
|                  | SS(26 2x32d)        | 128        | 0.2  | 1e-4| cosine| -      | 1e-4   | 600   |
|                  | SS(26 2x96d)        | 128        | 0.2  | 1e-4| cosine| -      | 1e-4   | 600   |
|                  | SS(26 2x112d)       | 128        | 0.2  | 1e-4| cosine| -      | 1e-4   | 600   |
|                  | Pyramid+SD          | 128        | 0.1  | 1e-4| cosine| -      | 1e-4   | 600   |
| CIFAR-100        | WRN-28-10           | 128        | 0.1  | 1e-4| cosine| -      | 1e-4   | 200   |
|                  | SS(26 2x32d)        | 128        | 0.1  | 1e-4| cosine| -      | 1e-4   | 1200  |
|                  | SS(26 2x96d)        | 128        | 0.1  | 1e-4| cosine| -      | 1e-4   | 1200  |
|                  | Pyramid+SD          | 128        | 0.5  | 1e-4| cosine| -      | 1e-4   | 1200  |
| ImageNet         | ResNet-50           | 2048       | 0.8  | 1e-4| multistep | [30,60,80] | 1e-4 | 90    |
|                  | ResNet-200          | 2048       | 0.8  | 1e-4| multistep | [30,60,80] | 1e-4 | 90    |
| CUB-200-2011     | ResNet-50           | 512        | 1e-3 | 1e-4| multistep | [30,60,90] | 1e-4 | 200   |
|                  | ResNet-101          | 512        | 1e-3 | 1e-4| multistep | [30,60,90] | 1e-4 | 200   |
| Stanford Cars    | ResNet-50           | 512        | 1e-3 | 1e-4| multistep | [30,60,90] | 1e-4 | 200   |
|                  | ResNet-101          | 512        | 1e-3 | 1e-4| multistep | [30,60,90] | 1e-4 | 200   |
| FGVC-Aircraft    | ResNet-50           | 512        | 1e-3 | 1e-4| multistep | [30,60,90] | 1e-4 | 200   |
|                  | ResNet-101          | 512        | 1e-3 | 1e-4| multistep | [30,60,90] | 1e-4 | 200   |

Therefore the agent can output more than one augmentation for a patch by setting $T > 1$. In our PAA model, we set time horizon $T = 1$, i.e., the augmentation policies take actions at every time step to save computational cost and avoid over-regularization. Even in the single-step decision-making process (i.e., $T = 1$), RL can still effectively search for the optimal policy as illustrated in [58], [59]. In addition, we use the SGD optimizer with an initial learning rate of 1e-4 to train the actor network and the critic network. Moreover, we default the number of patches $N$ to 16 except for CIFAR tasks $N = 4$. To ensure the reliability of our experiments, we run each experiment four times using different random seeds.

Moreover, we provide the detailed model architecture for each component in our PAA augmentation policy model, including feature extractor network, actor network and critic network. We use the pre-trained on ImageNet ResNet-18 backbone (excluding the final avgpool and softmax layer) to extract the deep features of the image and the patch, which are denoted as the state $s$ and the observation $o_i$ respectively. As for the global policy network and the critic network, the detailed model architectures are shown in Table II. The policy network is a fully convolutional neural network (FCN). The parameters of the actor networks of all agents are shared. Although they share parameters, each agent’s policy is not identical. When determining an action, each agent’s policy relies on different activations of the global policy network.

C. Comparison With State-of-the-Arts

We compare PAA with baseline pre-processing, Cutout [15], Mixup [6], Cutmix [7], Co-Mixup (Co-Mix) [60], AutoAugment (AA) [8], Fast AutoAugment (FastAA) [9], RandAugment (RA) [14], DADA [20] and Adversarial AA (AdvAA). The baseline follows, [61]: standardizing the data, horizontally flipping with 0.5 probability, zero-padding and random cropping. Our policy model architecture design mainly follows [62], and we make some adjustments according to our task scenario.

1) Classification Results on CIFAR-10 and CIFAR-100: For CIFAR-10 and CIFAR-100, we experiment on Wide-ResNet-28-10 (WRN-28-10) [63], Shake-Shake (SS) [61] and PyramidNet+ShakeDrop (Pyramid+SD), [64] models. The results are reported in Table IV, which shows the proposed approach consistently outperforms several state-of-the-art DA methods. We observe that the improvement of performance is relatively slight, due to the small image size of CIFAR which is $32 \times 32$. In the following, we further apply our proposed PAA on datasets with larger image sizes and other networks.

2) Classification Results on ImageNet: As shown in Table IV, we evaluate our method on ResNet-50 and ResNet-200 [48] backbones on ImageNet, and our PAA significantly improves the performance of the target networks. The results further demonstrate that our proposed method is an effective DA technique for consistent and expressive benefits for datasets with larger image sizes.

3) Effectiveness of Fine-Grained Classification: Furthermore, we evaluate our proposed method on fine-grained image recognition tasks. According to previous work [56], [57], we take ResNet-50 and ResNet-101 as the backbones. The results are shown in Table IV, which illustrates that the performance of PAA is consistently better than other methods and PAA has achieved remarkable performance on these challenging fine-grained tasks.

4) Effectiveness of Local Tasks: In order to further verify the effectiveness of our proposed method, we conduct experiments on the object detection task. We adopt the mainstream detector Faster R-CNN [65] (ResNet-50) on the dataset Pascal VOC 2007 and show the mAP results on the Table VIII. We observe that the proposed PAA achieves consistent improvements and PAA is indeed effective for local tasks such as object detection.

5) Results on Other Backbones: Furthermore, we show more results on other CNN models, e.g., ResNet [66] and EfficientNet (see Table IX). PAA also achieves performance on other CNN models.
TABLE IV
TEST SET ACCURACY (%) ON CIFAR-10 AND CIFAR-100, VALIDATION SET TOP-1 ACCURACY (%) ON ImageNet. TEST ACCURACY (%) ON VARIOUS FINE-GRAINED CLASSIFICATION DATASETS INCLUDING CUB-200-2011 (CUB), STANFORD CARS (CARS) AND FGVC-ACRAF (AIRCRAFT). RESULTS OF OUR PROPOSED PAA IS THE AVERAGE ACCURACY (+/− STANDARD DEVIATION) OVER FOUR RANDOM RUNS

| Dataset   | Model      | Baseline | CutOut | Mixup | CutMix | Co-Mix | AA   | FastAA | RA    | DADA | PAA     |
|-----------|------------|----------|--------|-------|--------|--------|------|--------|-------|------|---------|
| CIFAR-10  | WRN-28-10  | 96.1     | 96.9   | 97.1  | 97.2   | 97.3   | 97.4  | 97.3   | 97.1  | 97.3 | 97.5±0.1 |
|           | SS(26 2x32d) | 96.4     | 97.0   | 97.2  | 97.3   | 97.3   | 97.5  | 97.3   | 97.3  | 97.3 | 97.6±0.1 |
|           | SS(26 2x96d) | 97.1     | 97.4   | 97.7  | 97.8   | 97.8   | 97.7  | 97.7   | 97.6  | 97.8 | 98.1±0.1 |
|           | SS(26 2x112d) | 97.2     | 97.4   | 98.0  | 98.0   | 98.0   | 98.1  | 98.0   | 97.8  | 97.9 | 98.1±0.1 |
|           | Pyramid+SD | 97.3     | 97.7   | 98.0  | 98.1   | 98.0   | 98.1  | 98.0   | 98.2  | 98.3 | 98.6±0.1 |
| CIFAR-100 | WRN-28-10  | 81.2     | 81.6   | 82.1  | 82.8   | 82.5   | 82.9  | 82.7   | 82.8  | 82.5 | 83.4±0.3 |
|           | SS(26 2x96d) | 82.8     | 84.0   | 85.4  | 85.6   | 85.7   | 85.7  | 85.1   | 85.5  | 84.7 | 85.9±0.2 |
|           | Pyramid+SD | 86.0     | 87.8   | 88.5  | 88.9   | 88.9   | 89.3  | 88.1   | 88.9  | 88.8 | 89.2±0.1 |
| ImageNet  | ResNet-50  | 76.3     | 76.5   | 77.0  | 77.2   | 77.6   | 77.6  | 77.6   | 77.6  | 77.5 | 78.3±0.3 |
|           | ResNet-200 | 78.5     | 78.8   | 79.6  | 79.9   | 80.0   | 80.1  | 80.6   | 80.1  | 79.8 | 81.0±0.3 |

TABLE V
COMPARISON OF COMPUTATIONAL COST (GPU HOURS) BETWEEN OUR PROPOSED PAA AND OTHER PREVIOUS AUTOMATED DATA METHODS. WE TRAIN WIDE-RESNET-28-10 ON CIFAR-10 AND RESNET-50 ON ImageNet. SEARCH: THE TIME OF SEARCHING FOR AUGMENTATION POLICIES. TRAIN: THE TIME OF TRAINING THE TARGET NETWORK. THE TRAINING TIME OF PAA INCLUDES THE INFERENCE TIME REQUIRED TO COMPUTE THE STATE AND OBSERVATIONS. TOTAL: THE TOTAL TIME. EXCEPT PAA, ALL METRICS ARE CITED FROM [9], [12].

| Dataset   | GPU hours | Baseline | Mixup | CutMix | AA    | FastAA | AdvAA | PAA    |
|-----------|-----------|----------|-------|--------|-------|--------|-------|--------|
| CIFAR-10  |           | Search   | 0     | 0      | 5000  | 3.5    | ~0    | ~0     |
|           |           | Train    | 6     | 6.5    | 6     | 6      | 6     | 7.5    |
|           |           | Total    | 6     | 6.5    | 6.5   | 5006   | 9.5   | 7.5    |
| ImageNet  |           | Search   | 0     | 0      | 0     | 15000  | ~0    | ~0     |
|           |           | Train    | 160   | 220    | 220   | 160    | 160   | 1280   |
|           |           | Total    | 160   | 220    | 220   | 15160  | 610   | 1280   |

TABLE VI
ABLATION STUDY: PERFORMANCE COMPARISONS (%) OF PATCH RANDOM AUGMENT (PRA), PATCH AUTOMAUGMENT WITHOUT COOPERATION (PAAwOC) AND OUR PAA ON VARIOUS DATASETS WITH VARIOUS MODELS FOR A FAIR COMPARISON. OUR PAA ALSO ACHIEVES HIGHER ACCURACY. ALTHOUGH BOTH MARL AND ViT CAN LEARN THE RELATIONSHIP BETWEEN PATCHES, THEY HAVE DIFFERENT OBJECTIVES. MARL AIMS TO ALLOW PATCHES TO COOPERATE TO SELECT AUGMENTATION OPERATIONS, WHILE ViT AIMS TO EXTRACT PATCH-WISE FEATURES.

| Dataset   | Model      | Baseline | PRA | PAAwOC | PAA(ours) |
|-----------|------------|----------|-----|--------|-----------|
| CIFAR-10  | WRN-28-10  | 96.1     | 97.1 | 97.3   | 97.5      |
|           | SS(26 2x32d) | 96.4     | 97.4 | 97.4   | 97.6      |
|           | SS(26 2x96d) | 97.1     | 97.6 | 97.4   | 98.1      |
|           | SS(26 2x112d) | 97.2     | 97.8 | 97.9   | 98.1      |
|           | Pyramid+SD | 97.3     | 98.5 | 98.4   | 98.6      |
| CIFAR-100 | WRN-28-10  | 81.2     | 83.0 | 83.1   | 83.4      |
|           | SS(26 2x96d) | 82.9     | 85.7 | 85.7   | 85.9      |
|           | Pyramid+SD | 86.0     | 89.0 | 88.9   | 89.2      |
| ImageNet  | ResNet-50  | 76.3     | 77.9 | 78.0   | 78.3      |
|           | ResNet-200 | 78.5     | 80.4 | 80.6   | 81.0      |

D. Complexity Analysis
In this section, in order to further demonstrate the performance of PAA in terms of complexity, we compare the policy search time and training time of PAA with AA [8], FastAA [9] and AdvAA [12], as illustrated in Table V. The computational cost of our proposed PAA is estimated on GeForce GTX 1080 Ti while AA [8] and AdvAA [12] are on NVIDIA Tesla P100. In addition, FastAA [9] is estimated on NVIDIA Tesla V100. The computational cost is mainly used for searching policies and training the target networks. As shown in Table V, compared to previous works, PAA requires the fewest total computational resources, and the search time is almost negligible. As for the parameters, the total number of PAA model parameters (about 0.23 M) is less than 1% of the target network (e.g., ResNet50: about 25.5 M).

In summary, the main methods to reduce the computational cost lie in three aspects: 1) The augmentation policy network is jointly optimized with the target network, similar to [12], [13].
Namely, our proposed method searches for policies in an online manner, obviating thousands of policy validations on a small proxy network and the requirement for retraining the target network. Besides, we use fixed schedules for the two corresponding hyperparameters (i.e., probability and magnitude) of each transformation to effectively reduce the search space, which makes it easier to search for effective policies. By these means, PAA compresses most of the policy search time 2) As mentioned before, a MARL algorithm is adopted, in which all agents parallelly learn the augmentation policies, to reduce the training time. 3) Compared with previous DA methods using image-by-image sequential transformations, we perform parallel transformations on tensors. Specifically, we pick the patches in a batch performing the same operation to reconstruct a new tensor. And we use Kornia\(^1\) to realize tensor transformations on GPU to further reduce the processing time which is included in the training time.

E. Ablation Study

In this section, we study the effectiveness of MARL and discuss the design of patch numbers, through several ablation experiments.

1) Performance of Random Policies: We randomize the augmentation policy for each patch, dubbed Patch RandAugment and compare it with our proposed PAA. To be specific, the augmentation policy is randomly selected from predefined transformations with uniform probability. As shown in Table VII, Patch RandAugment (PRA) leads to considerable appreciable improvements. However, our Patch AutoAugment with meaningful guidance significantly surpasses the random patch policies.

\(^1\)Kornia [49] is a differentiable computer vision library for PyTorch. We use it to accelerate augmentation operation on tensors.

2) Performance of the Policies Searched Using PAA Without Cooperation Between Multiple Agents: We directly use \(N\) independent RL networks that do not share parameters but have the same network architecture, to search for each patch’s policy. We input the feature of the current patch into \(N\) policy network separately and then it outputs the augmentation operation of the current patch independently. Only the reward design is the same, it is difficult to learn a cooperative relationship based on this alone, denoted as Patch AutoAugment without communication (PAAw/oC). In short, PAAw/oC ignores the contextual relationship between patches, where the agents are independent and patches are non-cooperative. The results (see Table VI) indicate that taking the contextual information into account with cooperative RL-agents has improved the joint effectiveness of DA.

3) Impact of the Number of Patches: We explore the effect of the number of patches on the target model’s performance. As shown in Table VIII, we respectively set the number of patches \(N\) = \{4, 16, 64, 256, 1024\}. The results show that on ImageNet / CUB-200-111, when patch numbers \(N\) increase, the performance accuracy first increases and then decreases. In addition, when \(N = 16\), PAA achieves the best performance. We analyze that the small number of patches may cause PAA to be unable to effectively explore local diversity, and its advantages would be limited. In particular, when \(N = 1\), PAA degenerates into image-wise automated DA. In contrast, under too larger values (e.g., the extreme case is to search for DA policy for each pixel), the local semantic consistency is broken and the benefit brought by the consideration of contextual relationships between patches is gradually overtaken.

4) Impact of the Initialization of the Feature Extractor: We further study the impact of the initialization of the feature extractor as shown in Table VII. We first observe that the performance of the Non-pretrained ResNet-18 scheme (using non-pretrained ResNet-18 as the state/observation extractor) is still much better than the baseline. In other words, our agents do not rely on the pre-trained ResNet to produce features to be used as the state or observation. We also notice that the performance of the Non-pretrained ResNet-18 scheme is slightly lower than the Pretrained ResNet-18. This may be because at the very early

---

**TABLE VII**

| Dataset & Model | Baseline | Pretrained ResNet-18 | Non-pretrained ResNet-18 |
|----------------|----------|----------------------|--------------------------|
| CIFAR-100 (WRN-28-10) | 81.2     | 83.4                 | 83.2                     |

**TABLE VIII**

MAP (%) RESULTS ON PASCAL VOC. WE USE THE FASTER R-CNN AS THE BASELINE DETECTOR

| Method       | Baseline | CutOut | Mixup | AA   | PAA  |
|--------------|----------|--------|-------|------|------|
| mAP          | 76.0     | 77.2   | 78.3  | 78.7 | 79.0 |

**TABLE IX**

RESULTS ON SOTA MODELS (I.E., RESNEXT, EFFICIENTNET, DEiT, AND SWIN-TRANSFORMER ON IMAGENET-1 K)

| Model                | Arch.  | Baseline | AA   | Ours  |
|----------------------|--------|----------|------|-------|
| ResNext-50 (2x40d)   | CNN    | 77.0     | 77.2 | 77.4  |
| EfficientNet-B0      | CNN    | 77.3     | 77.4 | 77.7  |
| DeiT-S               | ViT    | 79.8     | 79.8 | 81.2  |
| Swin-Transformer     | ViT    | 81.2     | 81.3 | 81.5  |

**TABLE X**

ABLATION STUDY ON THE NUMBER OF PATCHES: PERFORMANCE COMPARISONS (TOP-1 ACCURACY (%)) UNDER FIVE DIFFERENT PATCH NUMBERS \(N\) ON CIFAR-10 (IMAGE SIZE: 32 × 32) WITH WRN-28-10, ON CUB-200-2011 (IMAGE SIZE: 448 × 448) WITH RESNET-50 BACKBONE AND ON IMAGENET (IMAGE SIZE: 224 × 224)

| \(N\)  | Baseline | 4 | 16 | 64 | 256 | 1024 |
|-------|----------|---|----|----|-----|------|
| CIFAR-10 (WRN-28-10) | 96.1 | 97.5 | 97.3 | 97.2 | 97.1 | -    |
| CUB-200-2011 (ResNet-50) | 85.5 | 87.2 | 87.5 | 87.3 | 87.1 | 86.8 |
| ImageNet (ResNet-50)    | 76.3 | 78.0 | 78.3 | 77.8 | 77.5 | 77.3 |
stage of the training, the features extracted by Non-pretrained ResNet-18 are not as good as Pretrained ResNet-18 in practice.

5) Impact of Time Horizon $T$: Time horizon $T$ reflects how often the target network feeds back rewards to update the policy network. We explore the effect of time horizon on Table XI, which illustrates that when $T$ increases, the performance accuracy decreases. This suggests that we should calculate the training loss of each batch to update the target network and calculate the corresponding reward to update our policy network every time step. Previous works [69], [70] indicate that value decomposition in MARL require sufficiently dense rewards to successfully learn to decompose the value function into the individual agents for further easy learning to achieve performance. Meanwhile, the augmentation policies take actions at every time step to save computational cost and avoid over-regularization, as illustrated in [58], [59]. Therefore, we set $T = 1$ to push our MARL policy network to obtain dense and frequent rewards and avoid much overhead to update the policy network once.

6) Impact of the Reward Design: We explore the effect of reward design on the target model’s performance. We take the loss of augmented input as the reward function and conduct ablation studies. The results (see Table XII) indicate that using the difference of loss is consistently better than using only the loss of augmented batches.

F. The Transferability of Learned Augmentation Policies

Here, we seek to understand the transferability of our learned policy network in the following two aspects: 1) cross-dataset: transfer augmentation policies from one dataset to another and 2) cross-network: transfer from one network to another on the same dataset. Specifically, we apply the policy network that is learned on CUB dataset with ResNet-50 on Aircraft to train ResNet-50. In addition, we transfer the policy network learned on ImageNet with ResNet50 to train ViT on ImageNet.

As can be seen from the results, the experimental results show that our learned augmentation policy network are surprisingly transferable and yield significant improvements on baseline models on these datasets. The learned augmentation policies we find are transferable between datasets. In addition, the policy learned on ImageNet with ResNet-50 transfers well to achieve significant improvements on other architecture (i.e., ViT). Despite the observed transferability, we find that the augmentation policy, which is directly applied to co-train with the target network to search for the optimal augmentation for patches, outperforms the transfer policy.

G. Visualization

1) Policy Visualization: Grad-CAM [71] is used to localize the important regions in the image. Therefore we can calculate the importance score of each patch, and we divide the importance scores into four bins, i.e., very important, important, normal and not important. Then, we categorize patches into four groups and draw four stacked area charts showing the percentages of operations selected by PAA augmentation policies over time. We take ResNet-50 backbone trained on CUB-200-2011 as an example. Since our proposed PAA searches for patch policies in an online manner, the strategies change dynamically over time as shown in Fig. 3. At the beginning of the training process, the selected actions are messy since the MARL network is in the exploratory stage. At the tail end of the training, the target network has converged, causing the percentage of all operations to be almost the same and the percentage to flatten out.

In addition, we have some interesting findings that may provide some insights to the DA community. i) The optimal augmentation strategies of patches vary by their importance levels. Therefore, it is necessary to take the image content into account when performing data augmentation. ii) In the middle of the training process, different types of patches prefer to select different augmentation operations. Concretely, as illustrated in Fig. 3, for the important patches, color transformation is mostly picked. For the unimportant patches, RandomEasing and Cutout are usually chosen by PAA. Important patches commonly take along semantic information that is related to mainstream tasks. It is better to choose the mild transformations (e.g., color) for them, which can effectively protect the semantic information from being damaged. In contrast, unimportant patches typically carry unexpected features [72] the are causally unrelated to the desired class. Severe transformations (e.g., Cutout, RandomErasing) could be chosen for them, which introduce noise and disturbance to reduce the impact of unexpected features on the target network learning.

2) Grad-CAM Visualization: Here, we adopt Grad-CAM [71] to visualize the learned features to intuitively show the impact of PAA, as shown in Fig. 4. We take the ResNet-50 as the backbone which is trained with dataset processed by 1) Baseline 2) AutoAugment and 3) Patch AutoAugment, respectively. We observe that the model trained with PAA focuses on more task-related areas rather than spurious correlations (e.g., the branch where the bird stands) or overemphasized features (e.g., bird claws).
Therefore, we leave the automated data augmentation for vision transformers to future work, which may provide interesting insights to the community.

V. Conclusion

In this paper, we propose Patch AutoAugment (PAA), a more fine-grained automated data augmentation approach. Our method adopts multi-agent reinforcement learning to automate the search for the optimal augmentation policies for patches, and encourages agents to cooperate with each other to further achieve the joint optimal policy across the entire image. Extensive experiments demonstrate that PAA improves the target network performance with low computational cost in various tasks. Meanwhile, we use visualization to show that PAA is beneficial for the targeted network to localize more class-related cues. Furthermore, we hope that our visual observations of policies will be useful for future development. In future work, we will investigate different schemes on dividing different regions. Additionally, our method is naturally aligned with the patch token mechanism in the current vision transformers [67], [68], [73] and the data augmentation specific to vision transformers has not been extensively studied. Therefore, we leave the automated data augmentation for vision transformers to future work, which may provide interesting insights to the community.

References

[1] C. Shorten and T. M. Khoshgoftaar, “A survey on image data augmentation for deep learning,” J. Big Data, vol. 6, no. 1, pp. 1–48, 2019.
[2] R. G. Lopes, S. J. Smullin, E. D. Cubuk, and E. Dyer, “Affinity and diversity: Quantifying mechanisms of data augmentation,” 2020, arXiv:2002.08973.
[3] Q. Wang et al., “Inter-domain adaptation label for data augmentation in vehicle re-identification,” IEEE Trans. Multimedia, vol. 24, pp. 1031–1041, 2022.
[4] P. Dai, Y. Li, H. Zhang, J. Li, and X. Cao, “Accurate scene text detection via scale-aware data augmentation and shape similarity constraint,” IEEE Trans. Multimedia, vol. 24, pp. 1883–1895, 2022.
[5] B. Chen, K.-H. Liu, Y. Xu, Q.-Q. Wu, and J.-F. Yao, “Block division convolutional network with implicit deep features augmentation for micro-expression recognition,” IEEE Trans. Multimedia, vol. 25, pp. 1345–1358, 2023.
[6] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, “Mixup: Beyond empirical risk minimization,” in Proc. 6th Int. Conf. Learn. Representations, 2018, pp. 1–13.
[7] S. Yun et al., “Cutmix: Regularization strategy to train strong classifiers with localizable features,” in Proc. IEEE Int. Conf. Comput. Vis., 2019, pp. 6023–6032.
[8] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, “Autoaugment: Learning augmentation policies from data,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2019, pp. 113–123.
[9] S. Lim, I. Kim, T. Kim, C. Kim, and S. Kim, “Fast autoaugment,” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 6665–6675.
[10] 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.
[11] R. Hataya, J. Zidenek, K. Yoshizoe, and H. Nakayama, “Faster autoaugment: Learning augmentation strategies using backpropagation,” in Proc. Comput. Vis. ECCV 2020: 16th Eur. Conf., Glasgow, U.K., Part XXV 16, Springer, Aug. 23–28, 2020, pp. 1–16.
[12] X. Zhang, Q. Wang, J. Zhang, and Z. Zhong, “Adversarial autoaugment,” in Proc. Int. Conf. Learn. Representations, 2019.
[13] C. Lin et al., “Online hyper-parameter learning for auto-augmentation strategy,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2019, pp. 6579–6588.
[14] 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, 2020, pp. 702–703.
[15] T. DeVries and G. W. Taylor, “Improved regularization of convolutional neural networks with cutout,” 2017, arXiv:1708.04552.
[16] A. Shrivastava et al., “Learning from simulated and unsupervised images through adversarial training,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 2107–2116.
[17] T. Tran, T. Pham, G. Carneiro, L. Palmer, and I. Reid, “A Bayesian data augmentation approach for learning deep models,” Adv. Neural Inf. Process. Syst., vol. 30, 2017.
[18] X. Ke, J. Zou, and Y. Niu, “End-to-end automatic image annotation based on deep CNN and multi-label data augmentation,” IEEE Trans. Multimedia, vol. 21, no. 8, pp. 2093–2106, Aug. 2019.
[19] J. Zhang, K. Chen, C. Qin, W. Zhang, and N.-H. Yu, “Distribution-preserving-based automatic data augmentation for deep image steganalysis,” IEEE Trans. Multimedia, vol. 24, pp. 4538–4550, 2022.
[20] Y. Li et al., “DADA: Differentiable automatic data augmentation,” 2020, arXiv:2003.03780.
[21] A. Liu, Z. Huang, Z. Huang, and N. Wang, “Direct differentiable augmentation search,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2021, pp. 12219–12228.

Table XIII

| Dataset       | Model         | Baseline | PAA (online) | PAA (offline) |
|---------------|---------------|----------|--------------|---------------|
| ImageNet      | ResNet-50     | 79.8     | 81.2         | 81.0          |
| CUB → Aircraft| ResNet-50     | 91.0     | 92.6         | 92.3          |
Shiqi Lin received the B.S. degree in electronics and information engineering from the Hefei University of Technology, Hefei, China, in 2020. She is currently working toward the M.S. degree with the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei. Her research interests include computer vision and reinforcement learning.

Tao Yu received the B.S. degree in electronics and information engineering from Anhui University, Hefei, China, in 2018. He is currently working toward the Ph.D. degree with the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei. He is also a Research Student with the National University of Singapore, Singapore. His research interests include computer vision and reinforcement learning. He has authored or coauthored papers in top-tier machine learning and computer vision conferences and journals, such as NeurIPS, ECCV, AAAI, ACM MM, and TMM. He has served as a reviewer for NeurIPS, ICML, ECCV, AAAI, ACM MM, TCSVT, and TCybernetics.

Ruoyu Feng received the B.Eng. degree from the Department of Automation, Southeast University, Nanjing, China. He is currently working toward the M.Sc. degree with the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei, China. His current research interests include image/video compression, image/video coding for machines, and multimodality pre-training.

Xin Li (Student Member, IEEE) received the B.S. degree in 2019 from the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei, China, where he is currently working toward the Ph.D. degree. He was a Research Intern with Microsoft Research Asia from 2021 to 2022. His research interests include image/video compression, image/video restoration, visual quality assessment, causal learning, and computer vision.

Lei Xiao received the M.S. degree from the Department of Information Science and Technology, University of Science and Technology of China, Hefei, China, in 2019. He is currently a Technical Engineer with Huawei Cloud Technology and Computing Company, Ltd. His research interests include pose estimation, action recognition, and representation learning.

Zhibo Chen (Senior Member, IEEE) received the B.Sc. and Ph.D. degrees from the Department of Electrical Engineering, Tsinghua University, Beijing, China, in 1998 and 2003, respectively. He is currently a Full Professor with the University of Science and Technology of China, Hefei, China. His research interests include image and video compression, visual quality of experience assessment, learning based visual signal representation, and coding. He has authored or coauthored more than 160 publications and more than 80 granted patent applications. Some of his standard proposals have been adopted in MPEG/VCEG on video coding and ITU-T P.1202 on video quality assessment. He is the Chair of IEEE Visual Signal Processing and Communications Technical Committee (VSPC-TC). He was an Associate Editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, Guest Editor for IEEE OPEN JOURNAL OF CIRCUITS AND SYSTEMS, TPC Chair of IEEE PCS 2019, Organization Committee Member of ICIP 2017 and ICME 2013, served as TPC Member in IEEE ISCAS, and IEEE VCIP.