When to Learn What: Model-Adaptive Data Augmentation Curriculum

Chengkai Hou[1], Jieyu Zhang[2], Tianyi Zhou[3]*
[1]Jilin university, [2]University of Washington, [3] University of Maryland
houck20@mails.jlu.edu.cn, jieyuz2@cs.washington.edu, tianyi@umd.edu

Abstract

*Data augmentation (DA) is widely used to improve the generalization of neural networks by enforcing the invariances and symmetries to pre-defined transformations applied to input data. However, a fixed augmentation policy may have different effects on each sample in different training stages but existing approaches cannot adjust the policy to be adaptive to each sample and the training model. In this paper, we propose "**M**odel-**A**daptive **D**ata **Aug**mentation (MADAug)" that jointly trains an augmentation policy network to teach the model "**when to learn what**". Unlike previous work, MADAug selects augmentation operators for each input image by a model-adaptive policy varying between training stages, producing a data augmentation curriculum optimized for better generalization. In MADAug, we train the policy through a bi-level optimization scheme, which aims to minimize a validation-set loss of a model trained using the policy-produced data augmentations. We conduct an extensive evaluation of MADAug on multiple image classification tasks and network architectures with thorough comparisons to existing DA approaches. MADAug outperforms or is on par with other baselines and exhibits better fairness: it brings improvement to all classes and more to the difficult ones. Moreover, MADAug learned policy shows better performance when transferred to fine-grained datasets. In addition, the auto-optimized policy in MADAug gradually introduces increasing perturbations and naturally forms an easy-to-hard curriculum. Our code is available at* `https://github.com/JackHck/MADAug`.

1. Introduction

Data augmentation is a widely used strategy to increase the diversity of training data, which improves the model generalization, especially in image recognition tasks [21, 35, 17]. Unlike previous works that apply manually-designed augmentation operations [6, 44, 47, 23, 4, 24], recent researchers have resorted to searching a data augmentation policy for a target dataset/samples. Despite the success of these learnable and dataset-dependent augmentation policies, they are fixed once learned and thus non-adaptive to either different samples or models at different training stages, resulting in biases across different data regions [2] or inefficient training.

In this paper, we study two fundamental problems towards developing a data-and-model-adaptive data augmentation policy that determines a curriculum of "when to learn what" to train a model: **(1)** *when to apply data augmentation in training?* **(2)** *what data augmentations should be applied to each training sample at different training stages?*

First, applying data augmentation does not always bring improvement over the whole course of training. For example, we observed that a model tends to learn faster during earlier training stages without using data augmentation. We hypothesize that models at the early stage of training even have no capability to recognize the original images so excessively augmented images are not conducive to the convergence of the models. Motivated by this observation, we first design a strategy called monotonic curriculum to progressively introduce more augmented data to the training. In particular, we gradually increase the probability of applying data augmentation to each sample by following the *Tanh* function (see Figure 1), so the model can be quickly improved in earlier stages without distractions from augmentations while reaching a better performance in later stages through learning from augmented data.

Secondly, a fixed augmentation policy is not optimal for learning every sample or different training stages. Although the monotonic curriculum gradually increases the augmented data as the model improves, it does not determine which augmentations applied to each sample can bring the most improvement to the model training. Intuitively, the model can learn more from diverse data augmentations. Moreover, the difficulty of augmented data also has a great impact on the training and it depends on both the augmentations and the sample they are applied to. For example, "simple" augmentation is preferred in the early stages to accelerate model convergence but more challenging aug-

*Corresponding author

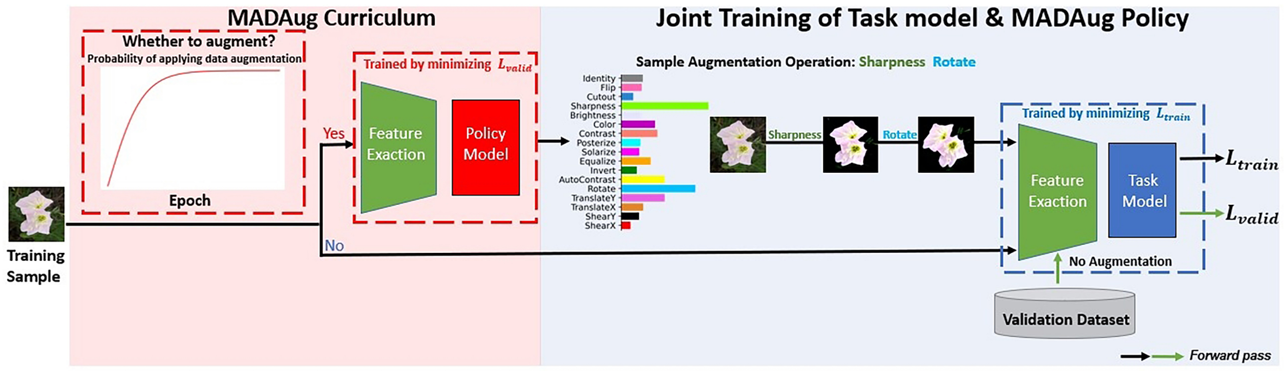

Figure 1: **MADAug** applies a monotonic curriculum to gradually introduce more data augmentations to the task model training and uses a policy network to choose augmentations for each training sample. MADAug trains the policy to minimize the validation loss of the task model, so the augmentations are model-adaptive and optimized for different training stages.

mented data provide additional information for learning more robust features for better generalization in the later stage. One plausible strategy is leveraging expert knowledge and advice to adjust the augmentation operation and their strengths [29, 47, 34, 14]. In this paper, instead of relying on human experts, we regard the evaluation of the current model on a validation set as an expert to guide the optimization of augmentation policies applied to each sample in different training stages. As illustrated in Figure 1, we utilize a policy network to produce the augmentations for each sample (*i.e*., data-adaptive) used to train the task model, while the training objective of the policy network is to minimize the validation loss of the task model (*i.e*., model-adaptive). This is a challenging bi-level optimization [5]. To address it, we train the task model on adaptive augmentations of training data and update the policy network to minimize the validation loss in an online manner. Thereby, the policy network is dynamically adapted to different training stages of the task model and generates customized augmentations for each sample. This results in a curriculum of data augmentations optimized for improving the generalization performance of the task model.

Our main contributions can be summarized as follows:

(a) A **monotonic curriculum** gradually introducing more data augmentation to the training process.

(b) **MADAug** that trains a data augmentation policy network on the fly with the task model training. The policy automatically selects augmentations for each training sample and for different training stages.

(c) Experiments on CIFAR-10/100, SVHN, and ImageNet demonstrate that MADAug consistently brings greater improvement to task models than existing data augmentation methods in terms of test-set performance.

(d) The augmentation policy network learned by MADAug on one dataset is **transferable to unseen datasets and downstream tasks**, producing better models than other baselines.

2. Related Work

Random crop and horizontal flip operations are commonly employed as standard data augmentation techniques for images in deep learning. Recently, there are significant advancements in advanced data augmentation techniques that have significantly increased the accuracy of image recognition tasks [46, 41, 43, 38, 9, 16, 17]. However, data augmentations may only be applicable to certain domains, and heuristically selected transformations, such as transplanting transformations that are effective in one domain into another, could have the opposite effect [2]. Thus, the exploration of optimal data augmentation policies necessitates specialized domain knowledge.

AutoAugment [6] adopts reinforcement learning to automatically find an available augmentation policy. However, AutoAugment requires thousands of GPU hours to find the policies on a reduced setting and limits randomness on the augmentation policies. To tackle these challenges, searching the optimal data augmentation strategies has become a prominent research topic and many methods have been proposed [46, 36, 13, 24, 14, 23, 25, 22, 44, 18, 45, 4].

These methods can be broadly classified into two distinct categories: fixed augmentation policies and online augmentation policies. The first category of methods [13, 23, 24, 47, 6, 45, 4] employs subsets of the training data and/or smaller models to efficiently discover fixed augmentation policies. However, the limited randomness in these policies makes it challenging to generate suitable samples for various stages of training. Thus, the fixed augmentation policies are suboptimal. The second category of methods [7, 36, 26, 22, 30, 44, 25] focuses on directly finding dynamic augmentation policies on the task model. This strategy is increasingly recognized as the primary choice for data augmentation search.

RandAugment [7] and TrivialAugment [30] are typically the second type of methods for finding online augmentations. They randomly select the augmentation parameters without relying on any external knowledge or prior information. Other methods, such as Adversarial AutoAugment [44], generate the adversarial augmentations by maximizing the training loss. However, the inherent instability of adversarial augmentations, without appropriate constraints, poses a risk of distorting the intrinsic meanings of images. To avoid this collapse, TeachAugment [36] utilizes the "teacher knowledge" to effectively restrict adversarial augmentations. However, Adversarial AutoAugment [44] and TeachAugment [36] both offer "hard" augmentations rather than "adoptive" augmentations, which are not effective to enhance the model generalization at the early training stage, because models at the early training even do not recognize the primitive images. "Hard" augmentations are reluctant to converge the model. Thus, in our paper, we gradually apply the data augmentations for samples and track the model performance on the validation set to adjust the policies through the original bi-level optimization during the model training.

3. Method

In this section, we first propose monotonic curriculum which progressively introduces more augmented samples as the training epoch increases. We then introduce the policy network that generates model-adaptive data augmentations and study how to train it through bi-level optimization with the task model.

3.1. When to Augment: Monotonic Curriculum

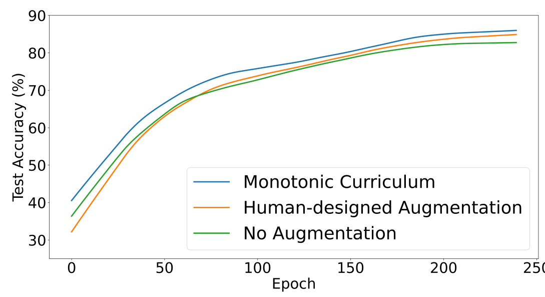

Figure 2: Test accuracy on Reduced CIFAR-10. **No Augmentation** does not apply any augmentations. **Human-designed Augmentation** always applies human pre-defined augmentations. **Monotonic Curriculum** gradually increases the probability of applying human-designed augmentations.

Previous studies [6, 23, 24, 13] have adopted the data augmentations for the whole model training process. However, at the early stage of model training, the model doesn't even recognize the original images. In this case, is data augmentation effective? In Figure 2, the test accuracy of a model trained on the Reduced CIFAR-10 dataset drops in the first ~ 70 epochs if applying human-designed data augmentations. To address this problem, at the beginning of model training, we only apply augmentations to a randomly sampled subset of training images while keeping the rest as original. In the later training stages, we apply a monotonic curriculum that gradually increases the proportion of images to be augmented or the probability of applying augmentation. Specifically, the proportion/probability $p(t)$ increases with the number of epochs by following a schedule defined by $\tanh$, i.e.,

$$p(t) = \tanh(t/\tau) \tag{1}$$

where t is the current training epoch number and τ is a manually adjustable hyperparameter that controls the change of proportion. Therefore, the early-stage model is mainly trained on the original images without augmentations, which helps the premature model converge quickly. However, as training proceeds, the model fully learned the original images and its training can benefit more from the augmented images. To validate the efficiency of our strategy, compare with the images without augmentation policy or with the fixed human-design augmentation policies, our method can effectively boost model performance during various training stages (see Figure 2).

3.2. What Augmentations to Apply: Model-Adaptive Data Augmentation

Instead of constantly applying the same data augmentation policies to all samples over the whole training process, adjusting the policy for each sample and model in different training stages can provide better guidance to the task model and thus accelerate its training towards better validation accuracy.

Following AdaAug [4], we assign an augmentation probability p and magnitude λ to each sample. The augmentation probability vector p contains the possibility p_i of applying each augmentation-i, *i.e.*, $\sum_{i=1}^{n} p_i = 1$, where there are n possible augmentation operations. The augmentation magnitude vector λ contains the associated augmentation strengths such that $\lambda_i \in [0, 1]$. In the training process, for every training image x, we draw k operations without replacement according to p and build an augmentation policy based on them and their magnitude in λ. In particular, each sampled augmentation operator-j is applied to the image x with magnitude λ_j, resulting in an augmented image $\Gamma^j(x) \triangleq \tau_j(x; \lambda_j)$. By applying the k sampled augmentations, the final augmented image $\gamma(x)$ can be written as:

$$\begin{aligned}\Gamma^t(x) &= \tau_j(x;\lambda_j), \quad j \sim p \\ \gamma(x) &= \Gamma^k \circ \cdots \circ \Gamma^1(x),\end{aligned} \tag{2}$$

where $\circ$ is the compositional operator.

An arbitrary augmentation policy is not guaranteed to improve the performance of a task model but a brute-force search is not practically feasible. Hence, we optimize a policy model producing the optimal augmentation probability vector p and magnitude vector λ for each image at different training stages. For image x, we define $f(x;w)$ as the task model with parameter w and $g_w(x)$ as the intermediate-layer representation of image x extracted from the task model $f(x;w)$. The policy model $p(\cdot;\theta)$ with parameters θ takes the extracted feature $g_w(x)$ as input and outputs the probability vector p and magnitude vector λ for the image x. The parameter w of the task model is optimized by minimizing the following training loss on the training set $\mathcal{D}^{tr} = \{x_i, y_i\}_{i=1}^{N^{tr}}$:

$$w = \arg\min_w \mathcal{L}^{tr}(w;\theta) = \frac{1}{N^{tr}} \sum_{i=1}^{N^{tr}} \mathcal{L}_{CE}(f(\gamma(x_i);w), y_i), \tag{3}$$

where the augmented training image $\gamma(x_i)$ is generated by the policy network $p(g_w(x_i);\theta)$ and $\mathcal{L}_{CE}(\cdot,\cdot)$ is the cross-entropy loss. The policy model is to produce augmentation policies applied to the training of the task model and its optimization objective is to minimize the trained task model's loss on a validation set, i.e., $\mathcal{D}^{val} = \{x_i^{val}, y_i^{val}\}_{i=1}^{N^{val}}$. The above problem can be formulated as the bi-level optimization [5] below:

$$\begin{aligned}\min_\theta \quad & \mathcal{L}^{val}(w^*(\theta)) = \frac{1}{N^{val}} \sum_{i=1}^{N^{val}} \mathcal{L}_i^{val}(w^*(\theta)) \\ s.t. \quad & w^*(\theta) = \arg\min_w \mathcal{L}^{tr}(w;\theta)\end{aligned} \tag{4}$$

where $\mathcal{L}_i^{val}(w^*(\theta)) = \mathcal{L}_{CE}(f(x_i^{val}; w^*(\theta)), y)$. Bi-level optimization is challenging because the lower-level optimization (i.e., the optimization of w) does not have a closed-form solution that can be substituted into the higher-level optimization (i.e., the optimization of θ). Recent work [33, 34, 47, 14, 29] address this problem (4) by alternating minimization. In this paper, we employ the same strategy as [34, 47, 1].

3.3. Joint Training of Task model & MADAug Policy

To address the bi-level optimization, we alternately update θ and w by first optimizing the policy network θ for a task model $\hat{w}$ achieved by one-step training and then update w using the augmentations produced by the new policy network θ.

We split the original training set into two disjoint sets, *i.e.*, a training set and a validation set. Each iteration trains the model on a mini-batch of n^{tr} images $\mathcal{D}_{m_i}^{tr} = \{x_i, y_i\}_{i=1}^{n^{tr}}$ drawn from the training set. Let $\mathcal{L}^{tr}(w_t;\theta_t) = \mathcal{L}_{CE}(f(\gamma(x_i);w_t), y_i)$ denote the lower-level objective for optimizing w_t. We apply one-step gradient descent on w_t to achieve a closed-form surrogate of the lower-level problem solution, i.e.,

$$\hat{w}_t = w_t - \alpha \frac{1}{n^{tr}} \sum_{i=1}^{n^{tr}} \nabla_w \mathcal{L}^{tr}(w_t;\theta_t) \tag{5}$$

where α is a learning rate. However, we cannot use back-propagation to optimize θ_t for the high-level optimization because the sampling process of the k augmentation operations in $\gamma(x_i)$ is non-differentiable. Hence, back-propagation cannot compute the partial derivative w.r.t. the augmentation probability p and magnitude λ. To address this problem, we relax the non-differentiable $\gamma(x_i)$ to be a differentiable operator. Since the augmentation policy in most previous work [6, 18] only consists of two operations, for $k = 2$, $\gamma(x_i)$ can be relaxed as

$$\gamma(x_i) \approx \sum_{j_1=1}^{n} \sum_{j_2=1}^{n} p_{ij_1} \cdot p_{ij_2} \Gamma_{ij_2}^2(\Gamma_{ij_1}^1(x_i)))) \quad j_1 \neq j_2 \tag{6}$$

where $\Gamma_{ij_k}^t(x_i) = \tau_{j_k}(x_i;\lambda_{j_k})$ applies augmentation-j_k (with magnitude λ_{j_k}) to x_i in the t-th augmentation operation. The relaxed $\gamma(x_i)$ is differentiable by combining different augmentations according to weights as their probabilities, so we can estimate the partial derivatives w.r.t. p via back-propagation through Eq. 6. In our approach, the forward pass still uses the sampling-based $\gamma(x_i)$, whereas the backward pass uses its differentiable relaxation in Eq. 6.

For back-propagation through the augmentation magnitude vector λ, we apply the straight-through gradient estimator [3, 39] because the magnitudes of some operations such as "Posterize" and "Solarize" are discrete variables that only have finite choices. In previous approaches [4, 13], the loss's gradient w.r.t. λ_m is estimated by applying the chain-rule to each pixel value $\gamma(x_{h,w})$ in the augmented image $\gamma(x)$, *i.e.* $\frac{\partial \gamma(x_{h,w})}{\partial \lambda_j} = 1$. Hence, the gradient of loss $\mathcal{L}$ w.r.t. λ_m can be computed as:

$$\frac{\partial \mathcal{L}}{\partial \lambda_m} = \sum_{h,w} \frac{\partial \mathcal{L}}{\partial \gamma(x_{h,w})} \frac{\partial \gamma(x_{h,w})}{\partial \lambda_m} = \sum_{h,w} \frac{\partial \mathcal{L}}{\partial \gamma(x_{h,w})} \tag{7}$$

Then, the policy network parameters θ_t can be updated by minimizing the validation loss computed by the current meta task model $\hat{w}_t$ on a mini-batch of validation set $\mathcal{D}^{val} = \{x_i^{val}, y_i^{val}\}_{i=1}^{n^{val}}$ with batch size n^{val}. Therefore, the outer loop updates of θ_t is formulated by:

$$\theta_{t+1} = \theta_t - \beta \frac{1}{n^{val}} \sum_{i=1}^{n^{val}} \nabla_\theta \mathcal{L}_i^{val}(\hat{w}_t(\theta_t)) \tag{8}$$

where β is a learning rate. The third step is to update the parameter w_t based on the parameter θ_{t+1} of the policy model in the outer loop of iteration $t+1$:

$$w_{t+1} = w_t - \alpha \frac{1}{n_{tr}} \sum_{i=1}^{n_{tr}} \nabla_w \mathcal{L}^{tr}(w_t; \theta_{t+1}) \tag{9}$$

With these updating rules, the policy and task networks can be alternatively trained. Our proposed algorithm is summarized in Algorithm 1.

Algorithm 1 Model-Adaptive Data Augmentation

```
Require: Training set $\mathcal{D}_{train} = \{x_i, y_i\}_{i\in[N_{train}]}$; Validation set $\mathcal{D}_{vaild} = \{x_i, y_i\}_{i\in[N_{valid}]}$; Batch sizes $n_{tr}, n_{val}$; Learning rate $\alpha, \beta$; Iteration number $T$;
Ensure: Task model $w_T$; policy network $\theta_T$
 1: Initialize $w_0$, $\theta_0$
 2: for $t = 0$ to $T$ do
 3:     Sample a training set mini-batch $d_{train} \in \mathcal{D}_{train}$.
 4:     Draw Augment$\sim P(t)$ in Eq. 1.
 5:     if Augment then
 6:         Apply policy network $\theta_t$ to achieve augmentations $\gamma(x)$ for each sample $x \in d_{train}$.
 7:     end if
 8:     Update $\hat{w}_t$ on the augmented $d_{train}$ (Eq.5).
 9:     Sample a validation set mini-batch $d_{valid} \in \mathcal{D}_{valid}$.
10:     Update policy network $\theta_{t+1}$ on $d_{valid}$ (Eq. 8).
11:     Apply policy network $\theta_{t+1}$ to achieve new augmentations $\gamma(x)$ for each sample $x \in d_{train}$.
12:     Update task model $w_{t+1}$ on the newly augmented $d_{train}$ (Eq. 9).
13: end for
```

4. Experiments

In this section, following AutoAugment [6], we examine the performance of MADAug on two experiments: MADAug-direct and MADAug-transfer. In the first experiment, we directly explore the performance of the MADAug on the benchmark datasets: CIFAR-10 [20], CIFAR-100 [20], SVHN [31], and ImageNet [8]. For CIFAR-10, CIFAR-100, and SVHN, we equally select 1,000 images from the dataset as the validation set to train the policy model. Plus, for SVHN, we apply both the training images and additional "extra" training images as the training set. For ImageNet, the validation set consists of 1,200 examples from a randomly selected 120 classes. We compare the average test set error of our method with previous state-of-the-art methods, AutoAugment (AA) [6], Population Based Augmentation (PBA) [18], Fast AutoAugment (Fast AA) [24], DADA [23], Faster AutoAugment (Fasrer AA) [13], RandAugment (RA) [7], TrivialAugmen (TA) [30], Deep AutoAugment (Deep AA) [45], TeachAugment (Teach) [36], OnlineAug [37], and AdaAug [4].

Our experiment results demonstrate that MADAug-direct considerably improves the accuracy of baselines and achieves state-of-the-art performance on these benchmark datasets. In the second experiment, we investigate the transferability of MADAug-learned policy network to unseen fine-grained datasets. To verify its effectiveness, we apply the augmentation policies learned by MADAug on the CIFAR-100 dataset to fine-grained classification datasets such as Oxford 102 Flowers [32], Oxford-IIIT Pets [10], FGVC Aircraft [28], and Stanford Cars [19]. Our findings demonstrate the remarkable transferability of MADAug-learned policy, which significantly outperforms the robust baseline models on fine-grained classification datasets.

4.1. Augmentation Operations

We follow the augmentation actions taken by AutoAugment [6]. We adopt the 16 augmentation operations (ShearX, ShearY, TranslateX, TranslateY, Rotate, AutoContrast, Invert, Equalize, Solarize, Posterize, Contrast, Color, Brightness, Sharpness, and Cutout) that are previously suggested to build the augmentation policies. Meanwhile, we add the Identity operation, which does not apply augmentation on images. For the sample baseline, we employ random horizontal flip, color jittering, color normalization, and Cutout with a 16×16 patch size as basic augmentations. The found policies learned by MADAug and other baselines are applied on top of these basic augmentations.

4.2. Implementation Details

In our experiments, the policy network of MADAug refers to a fully-connected layer that takes the representations produced by the penultimate layer of the task model as its inputs and outputs p and λ. Following AdaAug [4], the update of policy projection network parameters uses the Adam optimizer with a learning rate of 0.001. For the CIFAR-10, CIFAR-100, and SVHN, we evaluate our method on four models: Wide-ResNet- 40-2 [42], Wide-ResNet-28-10 [42], Shake-Shake (26 2x96d) [11], and PyramidNet with ShakeDrop [40, 12]. We train all models using a batch size of 128 except for PyramidNet with Shake-Drop, which is trained with a batch size of 64. We train the Wide-ResNet for 200 epochs and Shake-Shake/PyramidNet for 1,800 epochs. For Wide-ResNet models trained on SVHN, we follow PBA [18] to use the step learning rate schedule [9] and all other models use a cosine learning rate scheduler with one annealing cycle [27]. To align our re-

Table 1: **Test error (%, average of 5 random trials) on CIFAR-10, CIFAR-100, SVHN and ImageNet.** Lower value is better. "Simple" applies regular random crop, random horizontal flip, and Cutout. All other methods apply "Simple" on top of their proposed augmentations. We report the accuracy of our re-implemented AdaAug†, while other baselines' results are adapted from Zheng *et al*. [45], Cheung *et al*. [4], Tang *et al*. [37], and Suzuki *et al*. [36]. The best performance is highlighted in **Bold**.

Dataset	Backbone	Simple	AA	PBA	Fast AA	DADA	Faster AA	RA	TA	DeepAA	Teach	OnlineAug	AdaAug	AdaAug†	MADAug
Reduced CIFAR-10	Wide-ResNet-28-10	18.9	14.1	12.8	14.6	15.6	-	15.1	-	-	-	14.3	13.6	15.0	**12.5**
	Shake-Shake (26 2x96d)	17.1	10.1	10.6	-	-	-	-	-	-	-	-	10.9	11.8	**10.0**
CIFAR-10	Wide-ResNet-40-2	5.3	3.7	3.9	3.6	3.6	3.7	4.1	-	-	-	-	3.6	-	**3.3**
	Wide-ResNet-28-10	3.9	2.6	2.6	2.7	2.7	2.6	2.7	2.5	2.4	2.5	2.4	2.6	-	**2.1**
	Shake-Shake (26 2x96d)	2.9	2.0	2.0	2.0	2.0	2.0	2.0	1.9	1.9	2.0	-	-	-	**1.8**
	PyramidNet (ShakeDrop)	2.7	1.5	1.5	1.8	1.7	-	1.5	-	-	1.5	-	-	-	**1.4**
CIFAR-100	Wide-ResNet-40-2	26.0	20.6	22.3	20.7	20.9	21.4	-	19.4	-	-	-	19.8	-	**19.3**
	Wide-ResNet-28-10	18.8	17.1	16.7	17.3	17.5	17.3	16.7	16.5	16.1	16.8	16.6	17.1	-	**16.1**
	Shake-Shake (26 2x96d)	17.1	14.3	15.3	14.9	15.3	15.6	-	-	14.8	14.5	-	-	-	**14.1**
	PyramidNet (ShakeDrop)	14.0	10.7	10.9	11.9	11.2	-	-	-	-	11.8	-	-	-	**10.5**
Reduced SVHN	Wide-ResNet-28-10	13.2	8.2	7.8	8.1	7.6	-	9.4	-	-	-	**6.7**	8.2	9.1	8.4
	Shake-Shake (26 2x96d)	13.3	**5.9**	6.5	-	-	-	-	-	-	-	-	6.4	6.9	6.3
SVHN	Wide-ResNet-28-10	1.5	1.1	1.2	1.1	1.2	1.2	1.0	-	-	-	-	-	-	**1.0**
	Shake-Shake (26 2x96d)	1.4	1.1	1.1	1.1	1.1	-	-	-	-	-	-	-	-	**1.0**
ImageNet	ResNet-50	23.7	22.4	-	22.4	22.5	23.5	22.4	22.1	21.7	22.2	22.5	22.8	-	**21.5**

Table 2: **Transferability of MADAug learned policy network.** Test set error (%) of fine-tuning a pertrained ResNet-50 using the augmentations produced by the policy network on downstream tasks. Baseline results are adapted from Cheung *et al*. [4].

Dataset	# of classes	Train number	Simple	AA	Fast AA	RA	AdaAug	MADAug
Oxford 102 Flowers	102	2,040	5.0	6.1	4.8	3.9	2.8	**2.5**
Oxford-IIIT Pets	37	3,680	19.5	18.8	23.0	16.8	16.1	**15.3**
FGVC Aircraft	100	6,667	18.4	16.6	17.0	17.4	16.0	**15.4**
Stanford Cars	196	8,144	11.9	9.2	10.7	10.3	8.8	**8.3**

sults with other baselines, we train the ResNet-50 [15] from scratch on the full ImageNet using the hyperparameters in AutoAugment [6] on ImageNet. For all models, we use gradient clipping with magnitude 5. We provide specific details about the learning rate and weight decay values on the supplementary materials.

4.3. Main Results

Table 1 shows that the learned policies through bi-level optimization achieve the best performance than the baselines for different models on the Reduced CIFAR-10, CIFAR-10, CIFAR 100, Reduced SVHN, SVHN, and ImageNet. The Reduced CIFAR-10(SVHN) dataset randomly selects 4,000(1,000) images for CIFAR-10(SVHN) as the training set and sets the remaining images as the validation set. MADAug achieves state-of-the-art performance on this dataset. On the Reduced SVHN dataset, compared to AdaAug, we achieve 0.7% and 0.6% improvement on Wide-ResNet-28-10 and Shake-Shake (26 2x96d), respectively. On ImageNet, compare with other baselines, our method performs the best on a large and complex dataset. Different from the prior work (AutoAugment, PBA, and Fast AutoAugment) which constructs the fixed augmentation policies for the enter dataset, our method can find the dynamic and model-adoptive policies for each image, which enhances the model's generalization. We provide the average and variance of the experiment results in Section 4.7.

4.4. Transferability of MADAug-Learned Policy

Following AdaAug [4], we apply the augmentation policies learned from the CIFAR-100 directly on the fine-grained datasets (MADAug-direct). To evaluate the transferability of the policies found on CIFAR-100, we compare the test error rate with AutoAugment (AA) [6], Fast AutoAugment (Fast AA) [24], RandAugment (RA) [7], and AdaAug [4] using their published policies on CIFAR-100. For all the fine-grained datasets, we compare the transfer results by training the ResNet-50 model [15] pretrained on ImageNet. Following the experiment setting of AdaAug, we use the cosine learning rate decay with one annealing cycle [27] and train the model for 100 epochs. According to the validation performance, we adjust the learning rate for different fine-grained datasets. The weight decay is set as 1e-4 and the gradient clipping parameter is 5.

Table 2 shows that our method outperforms the other baselines when training the pertrained ResNet-50 model on these fine-grained datasets. Previous methods (AutoAugment [6], Fast Augmentation [24], and RandAugment [7]) apply the fixed augmentation policies. This strategy does not help the fine-grained sample to distinguish from each other, which makes the model easy to recognize their differences. In contrast, AdaAug and MADAug adapt the augmentation policies for the entire dataset to instance-level grade. Because our method gradually augments the images and provides more optimal augmentation policies to unseen fine-grained images according to their relationship to the classes that have been learned on the CIFAR-100 dataset, the model achieves better performance than AdaAug, especially for the Pet dataset. The Pet dataset only contains the "Cat" and "Dog" images. In Figure 3, we also show that MADAug can improve the model's ability to recognize "Cat" and "Dog" classes significantly on the Reduced CIFAR-10 dataset.

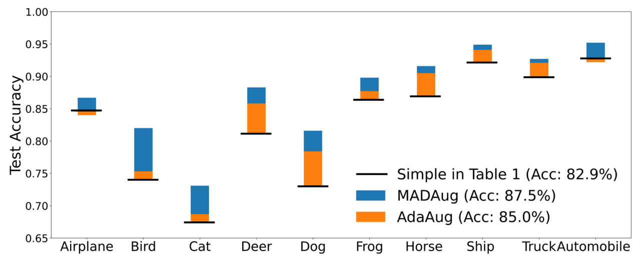

Figure 3: **Improvement that MADAug and AdaAug bring to different classes.** MADAug consistently improves the test accuracy over all classes and brings greater improvements to more difficult classes (fairness), *e.g.*, "Bird", "Cat", "Deer", and "Dog". In contrast, AdaAug has a negative impact on "Airplane" and "Automobile".

4.5. Analysis of MADAug Augmentations

We compare the pre-class accuracy from MADAug with AdaAug on the Reduced CIFAR-10 dataset, which only has 4,000 training images. Figure 3 shows the pre-class accuracy of a model trained on MADAug is higher than that trained on AdaAug and basic baseline, especially in "Bird", "Deer", "Cat" and "Dog" classes. Moreover, compared with the basic baseline, we can see that the augmentation policies trained by AdaAug play a negative impact on the "Airplane" and "Automobile" classes.

In Figure 4, we display some augmented samples with AdaAug and MADAug, which are randomly selected from "Bird", "Cat", "Deer", "Dog", "Airplane", and "Automobile" classes. For AdaAug, some augmented images in their classes have lost semantic information caused by the translation, because augmentation operations like TranslateY with unreasonable magnitude collapse the main information of the image. For example, in the Figure 4, the selected "Cat" augmented image loses its face and only leaves its legs. And the "Dog" augmented image also discards a part of key information. We think that these unreasonable data augmentation strategies lead to an imbalance in the number of samples containing sufficient information about their true label across different categories, although the original dataset is balanced [2]. This phenomenon leads to a reduction in the classification accuracy of some categories. However, for our method, Figure 5 shows augmentation policies generated by MADAug can produce more "hard" samples for the model with the training process. This strategy would improve the model generalization because at the early phase, "simple" samples can help models converge quickly and when the models have the capability to recognize the original samples, the "hard" samples can make them learn more robust features. Figure 4 shows augmented images on the different training phases. The model gradually receives more adversarial augmented images. And, these augmented policies learned by MADAug increase the diversity of training data and highlight the key information of data.

The same analysis of the Reduced SVHN dataset is presented in Appendix A. Through analyses of Reduced SVHN and Reduced CIFAR-10 dataset, from an experimental perspective, we illustrate that MADAug consistently provides higher-quality data augmentation strategies for samples, leading to improve more accuracy of the task model across different categories than AdaAug. From a methodological perspective, we also provide a detailed account of the advantages of MADAug over AdaAug in Appendix A.

4.6. Computing cost of MADAug

To demonstrate the effectiveness of MADAug, we present a comparison of GPU hours needed to search the augmentation policy and train the task model across different baselines. The results are showed in Table 3. The searching time of our method is regarded as the time to optimize Eq. 4. Thus, we do not need extra time to find data augmentation policies. Our approach is more effective than AutoAugment [6], PBA [18], and AdaAug [4].

Table 3: **Time consumption.** Comparison of computing cost (GPU hours) in training Wide-ResNet-28-10 on Reduced CIFAR-10 datasets between AutoAugment, PBA, AdaAug, and MADAug.

Method	Computing Cost		
	Searching	Training	Total
AutoAugment	5000	1.2	5001.2
PBA	5	1.2	6.2
AdaAug	2.9	1.4	4.3
MADAug	$\sim$0	1.8	**1.8**

4.7. Mean and variance of the experiment results

Table 4 represents the average values and variances of these experimental results obtained from multiply trials on different benchmark datasets.

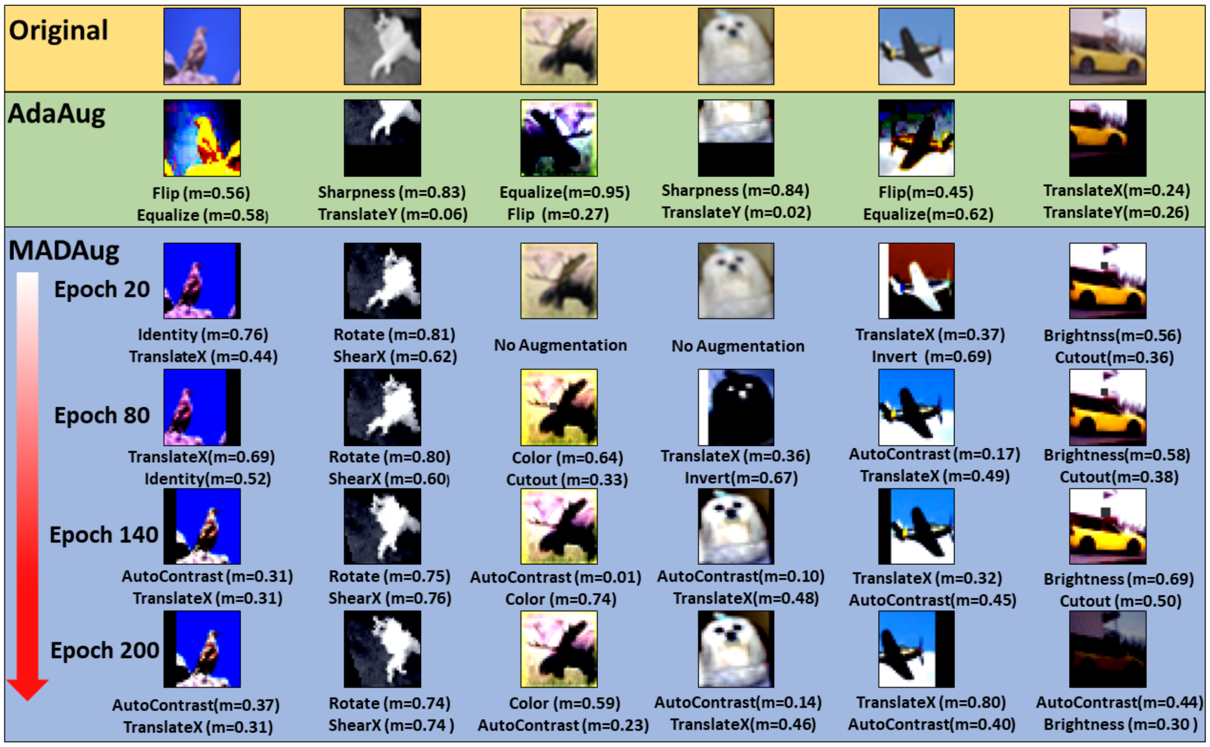

Figure 4: **Augmentations of AdaAug and MADAug for different classes of images (operations and associated strengths).** AdaAug only produces specific augmentations for different images, while MADAug adjusts the augmentations for each image to be adaptive to different training epochs. MADAug introduces less distortion than AdaAug.

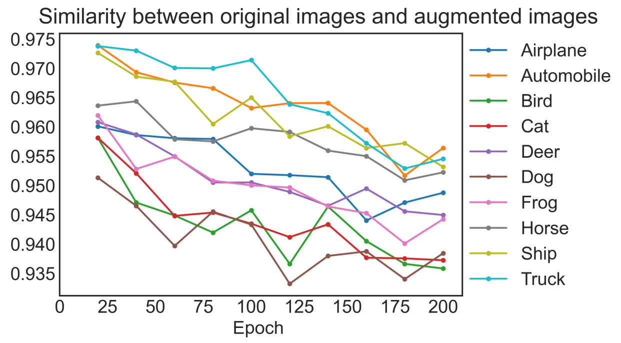

Figure 5: **Similarity between the original images and MADAug-augmented images at different training epochs.** MADAug starts from less perturbed images but generates more challenging augmentations in later training.

5. Ablation study

Magnitude perturbation. Following the AdaAug [4], we also add the magnitude perturbation δ for the augmentation policy. From Table 5, when the magnitude perturbation of operation is set as 0.3, the performance of the model on the test dataset is best. And we can conclude that the magnitude perturbation plays a positive effect in improving the generalization of the model.

Table 4: **Mean and variance of experiment results.** Test error and variance (%) of MADAug on different benchmark datasets with Wide-ResNet-28-10 and ResNet-50.

Dataset	Reduced CIFAR-10	CIFAR-10	CIFAR-100
	12.5±0.05	2.1±0.11	16.1±0.10
Dataset	Reduced SVHN	SVHN	ImageNet
	8.4±0.09	1.0±0.10	21.5±0.15

Number of augmentation operations. The number of operations k is arbitrary. Table 5 shows the relationship between the number of operations and the final test error on the Reduced CIFAR-10 with Wide-ResNet-28-10 model. The number of operations ranges from 1 to 5. When the augmentation operation is chosen as 2, we have the lowest error rate on the dataset. Policies learned by other methods (AutoAugment [6], PBA [18], and AdaAug [4]) also formulates two augmentation operations. This phenomenon indicates two augmentation operations' policies not only increase the diversity and amount of images but also do not make the task model unable to recognize the images due to excessive data augmentations.

Structure of policy network. Does the use of a nonlinear projection deliver better performance? We would add the policy model with the multiple hidden layers and the ReLU activation. Table 5 shows the influence of different number h of hidden layers on model performance. A single linear layer is sufficient for the policy model, without adding extra hidden layers.

Hyperparameter of τ. The hyperparameter τ controls the relationship between the epoch and the number of augmented samples. As is shown in Table 5, the performance of the task model is quite robust to hyperparameter $\tau \in \{10, 20, 30, 40, 50\}$. For the Reduced CIFAR-10, τ is optimally set as 40.

Analysis of optimization steps. Table 5 illustrates the impact of optimizing data augmentation strategies through different steps s on the experiment results using the Reduced CIFAR-10. The task model exhibits its highest accuracy when the step size is configured to 1.

Table 5: **Ablation study.** Sensitivity analysis of hyperparameter δ, k, h τ, and s on Reduced CIFAR-10 (Wide-ResNet-28-10).

δ	0	0.1	0.2	0.3	0.4
ACC(%)	86.7	87.0	87.3	**87.5**	87.2
k	1	2	3	4	5
ACC(%)	86.6	**87.5**	87.3	86.8	86.2
h	0	1	2	3	4
ACC(%)	**87.5**	87.2	87.1	86.9	86.7
τ	10	20	30	40	50
ACC(%)	87.0	87.3	87.4	**87.5**	87.3
s	1	2	5	10	30
ACC(%)	**87.5**	87.0	86.6	86.3	85.9

Effect of monotonic curriculum. We investigate the effect of monotonic curriculum which is introduced in Section 3.1. We train the Wide-ResNet-28-10 on the Reduced CIFAR-10 and Reduced SVHN datasets without/with this trick across different baselines. The results are shown in Table 6. For MADAug, monotonic curriculum contributes to the improvement of accuracy in these datasets. For other baselines, whether AutoAugment [6] method that applies the same data augmentation policy for the entire dataset, or AdaAug approach that offers different data augmentation policies to different samples, monotonic curriculum has been found effectively.

Table 6: **Effect of monotonic curriculum.** Test error (%) of MADAug and other baselines without/with monotonic curriculum.

Method	Monotonic curriculum	Reduced CIFAR-10	Reduced SVHN
AA		14.1	8.2
	✓	**13.7**	**7.8**
AdaAug		15.0	9.1
	✓	**14.4**	**8.7**
MADAug		13.1	8.9
	✓	**12.5**	**8.4**

Strategy of MADAug. MADAug not only dynamically adjusts the augmentation strategies to minimize the loss of the task model on the validation set which is named a model-adaptive strategy but also provides different data augmentation policies for each sample called data-adaptive strategy. In order to verify the effectiveness of MADAug, we use one of these two strategies to find the augmentation policies and train the task model to classify the dataset. Table 7 shows MADAug combines these two training strategies well and offers the higher quality of data augmentation policies for the dataset.

Table 7: **Effect of model/data-adaptive augmentation strategy.** Test error (%) of model-adaptive/data-adaptive only MADAug on two datasets.

Model-adaptive	Data-adaptive	Reduced CIFAR-10	Reduced SVHN
✓		14.5	9.1
	✓	14.0	9.6
✓	✓	**12.5**	**8.3**

6. Conclusion

In this paper, we propose a novel and general data augmentation method, MADAug, which is able to produce instance-adaptive augmentations adaptive to different training stages. Compared to previous methods, MADAug is featured by a monotonic curriculum that progressively increases augmented data and a policy network that generates augmentations optimized to minimize the validation loss of a task model. MADAug achieves SOTA performance on several benchmark datasets and its learned augmentation policy network is transferable to unseen tasks and brings more improvement than other augmentations. We show that MADAug-augmentations preserve the key information of images and change with the task model in different training stages accordingly. Due to its data-and-model-adaptive property, MADAug has a great potential to improve a rich class of machine learning tasks in different domains.

References

[1] Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your maml. *arXiv preprint arXiv:1810.09502*, 2018. 4

[2] Randall Balestriero, Leon Bottou, and Yann LeCun. The effects of regularization and data augmentation are class dependent. *arXiv preprint arXiv:2204.03632*, 2022. 1, 2, 7

[3] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 4

[4] Tsz-Him Cheung and Dit-Yan Yeung. Adaaug: Learning class-and instance-adaptive data augmentation policies. In *International Conference on Learning Representations*, 2021. 1, 2, 3, 4, 5, 6, 7, 8, 13

[5] Benoît Colson, Patrice Marcotte, and Gilles Savard. An overview of bilevel optimization. *Annals of operations research*, 153:235–256, 2007. 2, 4

[6] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018. 1, 2, 3, 4, 5, 6, 7, 8, 9

[7] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 702–703, 2020. 2, 3, 5, 6

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 5

[9] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017. 2, 5

[10] Yan Em, Feng Gag, Yihang Lou, Shiqi Wang, Tiejun Huang, and Ling-Yu Duan. Incorporating intra-class variance to fine-grained visual recognition. In *2017 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1452–1457. IEEE, 2017. 5

[11] Xavier Gastaldi. Shake-shake regularization. *arXiv preprint arXiv:1705.07485*, 2017. 5

[12] Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep pyramidal residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5927–5935, 2017. 5

[13] Ryuichiro Hataya, Jan Zdenek, Kazuki Yoshizoe, and Hideki Nakayama. Faster autoaugment: Learning augmentation strategies using backpropagation. In *European Conference on Computer Vision*, pages 1–16. Springer, 2020. 2, 3, 4, 5

[14] Ryuichiro Hataya, Jan Zdenek, Kazuki Yoshizoe, and Hideki Nakayama. Meta approach to data augmentation optimization. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2574–2583, 2022. 2, 4

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6

[16] Dan Hendrycks, Norman Mu, Ekin D Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. *arXiv preprint arXiv:1912.02781*, 2019. 2

[17] Alex Hernández-García and Peter König. Data augmentation instead of explicit regularization. *arXiv preprint arXiv:1806.03852*, 2018. 1, 2

[18] Daniel Ho, Eric Liang, Xi Chen, Ion Stoica, and Pieter Abbeel. Population based augmentation: Efficient learning of augmentation policy schedules. In *International Conference on Machine Learning*, pages 2731–2741. PMLR, 2019. 2, 4, 5, 7, 8

[19] Jonathan Krause, Jia Deng, Michael Stark, and Li Fei-Fei. Collecting a large-scale dataset of fine-grained cars. 2013. 5

[20] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 5

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 1

[22] Ruihui Li, Xianzhi Li, Pheng-Ann Heng, and Chi-Wing Fu. Pointaugment: an auto-augmentation framework for point cloud classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6378–6387, 2020. 2

[23] Yonggang Li, Guosheng Hu, Yongtao Wang, Timothy Hospedales, Neil M Robertson, and Yongxin Yang. Dada: Differentiable automatic data augmentation. *arXiv preprint arXiv:2003.03780*, 2020. 1, 2, 3, 5

[24] Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast autoaugment. *Advances in Neural Information Processing Systems*, 32, 2019. 1, 2, 3, 5, 6

[25] Chen Lin, Minghao Guo, Chuming Li, Xin Yuan, Wei Wu, Junjie Yan, Dahua Lin, and Wanli Ouyang. Online hyper-parameter learning for auto-augmentation strategy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6579–6588, 2019. 2

[26] Tom Ching LingChen, Ava Khonsari, Amirreza Lashkari, Mina Rafi Nazari, Jaspreet Singh Sambee, and Mario A Nascimento. Uniformaugment: A search-free probabilistic data augmentation approach. *arXiv preprint arXiv:2003.14348*, 2020. 2

[27] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 5, 6

[28] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013. 5

[29] Saypraseuth Mounsaveng, Issam Laradji, Ismail Ben Ayed, David Vazquez, and Marco Pedersoli. Learning data augmentation with online bilevel optimization for image classification. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1691–1700, 2021. 2, 4

[30] Samuel G Müller and Frank Hutter. Trivialaugment: Tuning-free yet state-of-the-art data augmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 774–782, 2021. 2, 3, 5

[31] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011. 5

[32] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729. IEEE, 2008. 5

[33] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *International conference on machine learning*, pages 4334–4343. PMLR, 2018. 4

[34] Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. Meta-weight-net: Learning an explicit mapping for sample weighting. *Advances in neural information processing systems*, 32, 2019. 2, 4

[35] Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. *Advances in neural information processing systems*, 28, 2015. 1

[36] Teppei Suzuki. Teachaugment: Data augmentation optimization using teacher knowledge. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10904–10914, 2022. 2, 3, 5, 6

[37] Zhiqiang Tang, Yunhe Gao, Leonid Karlinsky, Prasanna Sattigeri, Rogerio Feris, and Dimitris Metaxas. Onlineaugment: Online data augmentation with less domain knowledge. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16*, pages 313–329. Springer, 2020. 5, 6

[38] Yuji Tokozume, Yoshitaka Ushiku, and Tatsuya Harada. Between-class learning for image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5486–5494, 2018. 2

[39] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017. 4

[40] Yoshihiro Yamada, Masakazu Iwamura, Takuya Akiba, and Koichi Kise. Shakedrop regularization for deep residual learning. *IEEE Access*, 7:186126–186136, 2019. 5

[41] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019. 2

[42] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 5

[43] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 2

[44] Xinyu Zhang, Qiang Wang, Jian Zhang, and Zhao Zhong. Adversarial autoaugment. *arXiv preprint arXiv:1912.11188*, 2019. 1, 2, 3

[45] Yu Zheng, Zhi Zhang, Shen Yan, and Mi Zhang. Deep autoaugment. *arXiv preprint arXiv:2203.06172*, 2022. 2, 5, 6

[46] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13001–13008, 2020. 2

[47] Fengwei Zhou, Jiawei Li, Chuanlong Xie, Fei Chen, Lanqing Hong, Rui Sun, and Zhenguo Li. Metaaugment: Sample-aware data augmentation policy learning. In *Proceedings of the AAAI Conference on Ar-*

tificial Intelligence, volume 35, pages 11097–11105, 2021. 1, 2, 4