A^2-Aug: Adaptive Automated Data Augmentation

Lujun Li[1,2], Zheng Zhu[3], Guan Huang[4], Dalong Du[4], Jiwen Lu[3], Jie Zhou[3],Qingyi Gu[1*]
[1]Institute of Automation, Chinese Academy of Sciences
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences
[3]Tsinghua University [4]XForwardAI
{lilujun2019,qingyi.gu}@ia.ac.cn, {zhengzhu,lujiwen,jzhou}@tsinghua.edu.cn,
{guan.huang,dalong.du}@xforwardai.com

Abstract

*Data augmentation is a promising way to enhance the generalization ability of deep learning models. Many proxy-free and proxy-based automated augmentation methods are proposed to search for the best augmentation for target datasets. However, the proxy-free methods require lots of searching overhead, while the proxy-based methods introduce optimization gaps with the actual task. In this paper, we explore a new proxy-free approach that only needs a small number of searches (~ 5 vs 100 of RandAugment) to alleviate these issues. Specifically, we propose **A**daptive **A**utomated **Aug**mentation (A^2-Aug), a simple and effective proxy-free framework, which seeks to mine the adaptive ensemble knowledge of multiple augmentations to further improve the adaptability of each candidate augmentation. Firstly, A^2-Aug automatically learns the ensemble logit from multiple candidate augmentations, which is jointly optimized and adaptive to target tasks. Secondly, the adaptive ensemble logit is used to distill each logit of input augmentation via KL divergence. In this way, these a few candidate augmentations can implicitly learn strong adaptability for the target datasets, which enjoy similar effects with many searches of RandAugment. Finally, equipped with joint training via separate BatchNorm and normalized distillation, A^2-Aug obtains state-of-the-art performance with less training budget. In experiments, our A^2-Aug achieves 4% performance gain on CIFAR-100, which substantially outperforms other methods. On ImageNet, we obtain a top-1 accuracy of 79.2% for ResNet-50, a 1.6% boosting over the AutoAugment with at least $25\times$ faster training speed. Code will be available at: https://github.com/lilujunai/A2-Aug-Series.*

* Corresponding author

Table 1: Accuracy results for AA [5], Fast AA [30], PBA [16], RA [6] and our A^2-Aug on CIFAR-10 [22], CIFAR-100 [22] and ImageNet [7] classification tasks, respectively. N/A means no published result is available. The total cost is estimated from the report of the original paper (see the Table 4 for more details).

Method	Total cost (GPU-h)	CIFAR-10 PyramidNet	CIFAR-100 WRN-28-10	ImageNet ResNet-50
Baseline	384	97.3	81.2	76.3
AA	15,640	98.5	82.9	77.6
Fast AA	1,026	98.3	82.7	77.6
PBA	N/A	98.3	83.3	N/A
RA	10,368	98.5	83.3	77.6
A^2-Aug	**624**	**98.9**	**85.2**	**79.2**

1. Introduction

Convolutional neural network has become a state-of-the-art technique for computer vision, but it always suffers from the over-fitting problem without sufficient labelled images. Data augmentation [40, 3, 4], such as cropping , flipping, and color jittering, is an effective training technique, which is widely used for many computer vision tasks (*e.g.*, classification [24, 48], detection [11] , and segmentation [33]). Data augmentation has been shown as a useful regularizer that reduces over-fitting by transforming images to increase the quantity and diversity of training data. Notably, applying well-designed augmentations rather than naive random transformations in training improves the generalization ability [37]. However, in most cases, designing such augmentations requires human experts with prior knowledge of the dataset.

With the recent advancement of AutoML [60, 38, 55], there are a series of methods to search for an automated augmentation policy, including proxy-based and proxy-free automated augmentation methods. The pioneering proxy-based method, AutoAugment (AA) [5] uses reinforcement learning to search for the best policy from the huge search

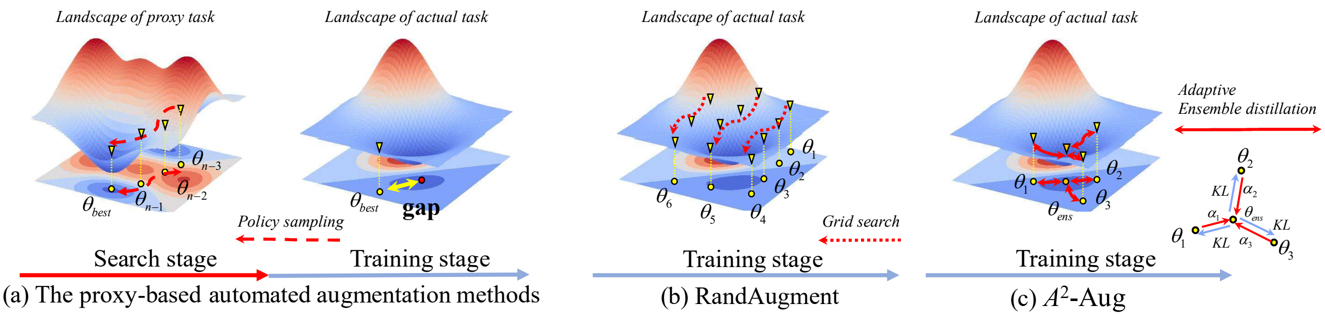

Figure 1: Comparison between the proxy-based automated augmentation methods, RandAugment and our approach. (a) The proxy-based automated augmentation methods first search for the optimal augmentation θ_{best} on the proxy task and then apply θ_{best} for augmented training on the actual task. (b) RandAugment implements a lot of grid search on the actual task. (c) Our A^2-Aug only samples a small amount of augmentation and jointly optimizes their adaptive ensemble augmentation θ_{ens} on actual task with weight factor $\boldsymbol{\mu}$ to get further improvement.

space. Although AutoAugment achieves excellent performance, it requires lots of training cost (see the Table 1). The latter proxy-based methods use a smaller proxy task with weight sharing for each policy to reduce costs, such as Fast AutoAugment (Fast AA) [30] and Population Based Augmentation (PBA) [16]. As shown in the Figure 1 (a), these proxy-based automated augmentation methods include the separate searching and training stage. However, the weight sharing leads to a serious gap between proxy tasks and actual tasks, which is a common challenge in most areas of AutoML [19, 18]. To address this issue, the proxy-free method (*e.g.*, RandAugment (RA) [6]) reduces the search space dramatically, allowing it to be trained on the target task with no need for a separate proxy task (see the Figure 1 (b)). However, RandAugment still requires lots of training costs for the grid search.

In this paper, we propose A^2-Aug, an **A**daptive proxy-free **A**utomated **Aug**mentation method that only needs a few searches. As shown in the Figure 1 (c), A^2-Aug does not require a separate search phase and a large number of grid searches, which can dramatically reduce the training budget. We find that misclassified input samples under one augmentation may be correctly classified by other augmentations. Thus, the adaptive ensemble of multiple augmentations can obtain stronger performance. Therefore, A^2-Aug jointly optimizes multiple augmentations and uses their adaptive ensemble logit to distill each candidate augmentation. This allows A^2-Aug to achieve significant performance gains without lots of searches. Specifically, we first randomly sample multiple augmentations and perform joint training with a single model with separate BatchNorm parameters for different augmentations. Then each logit of multiple augmentations is ensembled by an adaptive weight factor, which is co-optimized on the fly with the target task. Finally, we force the strong adaptive ensemble logit (*i.e.*, teacher) to perform distillation for each logit (*i.e.*, student). In this way, each candidate augmentation obtains similar adaptability to the best searched augmentation from many searches for the target dataset. We normalize each logit before mimicking the KL divergence, effectively improving the distillation performance.

In principle, our A^2-Aug is different from other automated augmentation methods because it maximizes the ensemble knowledge of multiple augmentations to further adapt to the target dataset. The merits of A^2-Aug lie in three-fold. First, it does not require a separate search stage compared to proxy-based methods, so there are no optimization gaps for A^2-Aug training. Second, each augmentation of A^2-Aug can be effectively improved and implicit adaptively optimized for the target dataset under the distillation training. Finally, A^2-Aug only needs a small number of searches to achieve significant performance improvements. Furthermore, A^2-Aug is easy to implement using joint training and does not require domain knowledge.

We conduct extensive experiments on different deep models and datasets to verify the superiority of the proposed method. As shown in the Table 1, A^2-Aug achieves a consistent and significant accuracy boost in various neural networks and datasets, which outperforms other methods by a large margin with less training overhead. For example, A^2-Aug obtains 1.6% accuracy gains on CIFAR-10 and 4.0% accuracy gains on CIFAR-100. On the challenging ImageNet dataset, A^2-Aug can improve the top-1 accuracy of ResNet-50 from 76.3% to 79.2%, which is a state-of-the-art performance among automated augmentation methods.

The contributions of this work are three-folds:

- We propose A^2-Aug, a new proxy-free automated augmentation framework that only needs a limited number of searches. A^2-Aug enjoys a better trade-off between performance and computation than conventional proxy-free methods, significantly boosting its application.

- A^2-Aug maximizes the ensemble logit of multiple augmentations to improve the adaptability of each candidate augmentation for the target dataset. With joint training and normalized distillation, A^2-Aug can efficiently and significantly improve the performance with less training budget.

- We perform a thorough evaluation of CIFAR-10, CIFAR-100, and ImageNet. A^2-Aug achieves state-of-the-art performances in various neural networks and datasets. Specifically, we achieve 79.2% top-1 accuracy for ResNet-50 on ImageNet, which outperforms AutoAugment with 1.6% significant margin and 25× faster training speed.

2. Related work

Our A^2-Aug is a new data augmentation framework, which ensembles multiple augmentations and performs distillation. In this section, we introduce related work about data augmentation, model ensemble, and knowledge distillation.

Data augmentation. Data augmentation [27, 4, 47, 44, 26] is a prevailing regularization method to curb overfitting and improve network generalization. It increases the amount and diversity of training data using linear or non-linear transformations over the original data. Some specially-designed augmentation methods like Mixup [59], CutOut [8], and CutMix [56] have been shown to improve the performance of the trained model. However, such augmentation strategies always require domain knowledge and manual design. Inspired by neural architecture search [1, 17], many data augmentation methods automatically search for optimal strategies on a specific dataset. AutoAugment [5] samples the best policies with reinforce learning. Although AutoAugment achieves excellent performance, its search process is computationally expensive. To improve the search efficiency, several proxy-based search algorithms [30, 16, 29], have been proposed. For example, Population Based Augmentation [16] employes an efficient population-based optimization to search data augmentation schedules. However, these methods all implement the search on the proxy task, which does not always achieve a significant improvement in the actual taskt [6]. Therefore, the proxy-free method, such as RandAugment [6], implements a simple search on the actual task and achieves considerable performance. However, RandAugment still brings huge training costs for its grid search. Our A^2-Aug is a proxy-free automated augmentation method, which does not rely on proxy tasks. Different from RandAugment, A^2-Aug jointly optimizes various augmentations and uses their ensemble knowledge to further improve the performance of each augmentation. This allows A^2-Aug to achieve significant performance gains without lots of training cost.

Model ensemble. Model ensemble [32, 21] is an effective machine training method, which combines several well-trained models to boost performance. Most existing ensemble methods [34, 28, 13] average the output of each model, which neglects the diversity. Different from these methods, A^2-Aug performs an adaptive ensemble using a learnable weight factor, which strengthens the weight of powerful augmentation and can be updated during training.

Konwlegde distillation. Knowledge distillation is a simple yet effective training strategy, which works by transferring the knowledge (*e.g.*, outputted logits [15, 12], intermediate feature [39, 52, 58, 49], and relational information [41, 35]) from the teacher model to student model. In our A^2-Aug, each logit of different augmentations acts as a student model, while their adaptive ensemble logit acts as a teacher model. We use ensemble logit to distill each model to learn from the implicit adaptive ensemble augmentation to fit the target dataset.

3. Review of automated augmentation

In this section, we review the conventional automated augmentation methods. Similar to the AutoML methods [60, 38], these methods (*e.g.*, proxy-free methods and proxy-based methods) evaluate and search for the best augmentation policy θ within a huge search space $\mathcal{A}(\theta)$ to further fit the target dataset. For the neural network $\mathcal{F}(\cdot, w)$ with weights w, train dataset $\mathcal{X}_{train} = \{(x_i, y_i)\}_{i=1}^{N_{train}}$, and validation dataset $\mathcal{X}_{val} = \{(\hat{x_i}, \hat{y_i})\}_{i=1}^{N_{val}}$, the purpose of automated augmentation is to find the optimal policy θ. Such that when the weights are optimized on the training set, the validation loss is minimized. Generally, the proxy-free automated augmentation needs to optimize the network weight w first, as:

$$w_\theta = \arg\min_w \mathcal{L}_{\text{train}}(\mathcal{F}(\theta(x), w), y), \tag{1}$$

where $\mathcal{L}_{\text{train}}(\cdot)$ refers to the loss function on the training set. Then, it searches the best augmentation policy θ on the validation set, as:

$$\theta^* = \arg\max_{\theta \in \mathcal{A}} \text{ACC}_{\text{val}}\left(w_\theta\right), \tag{2}$$

where $\text{ACC}_{\text{val}}(\cdot)$ refers to the accuracy on the validation set. For different policy θ sampled from the search space $\mathcal{A}(\theta)$, its corresponding optimal weights need to be trained independently from scratch. When $\mathcal{A}(\theta)$ is very large, the total training cost is expensive. Only small datasets and small search space are affordable. For example, RandAugment [6] implements a grid search on the small search space (10^2) and small dataset (CIFAR-10).

To avoid the substantial cost of individual training, most proxy-based automated augmentation methods [29, 30] make various augmentation policies θ to inherit their weights directly from a proxy task $\mathcal{F}(\cdot, W)$, where W is the shared weight. To facilitate optimization, these methods relax the discrete search space $\mathcal{A}(\theta)$ into a continuous one $\mathcal{A}(\widetilde{\theta})$,

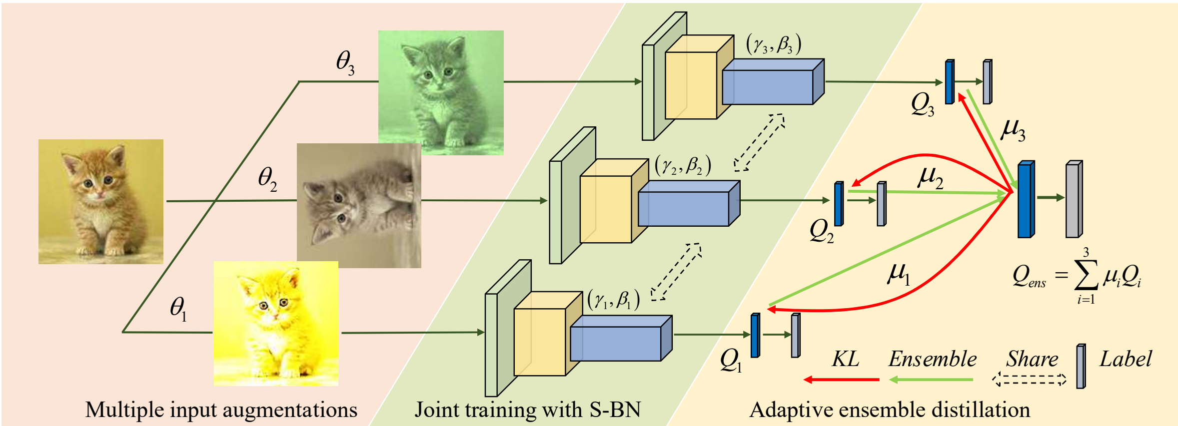

Figure 2: The overview of A^2-Aug. During training, we first perform joint training for each input augmentation θ with a single model, which has separate BatchNorm parameters (γ_i, β_i) for different augmentations. Then, we obtain ensemble logit $\boldsymbol{Q_{ens}}$ with adaptive weight factor $\boldsymbol{\mu}$ and apply it to distill each logit $\boldsymbol{Q_i}$ with KL divergence. After training, the model of the best augmentation can be employed separately in model inference.

where $\widetilde{\theta}$ denotes the continuous policies that represent the distribution of the θ. In such a continuous search space, augmentation policy $\widetilde{\theta}$ can be flexibly sampled from $\mathcal{A}(\widetilde{\theta})$ and optimized together with weights, as

$$(\widetilde{\theta}^*, W_{\widetilde{\theta}^*}) = \arg\min_{\widetilde{\theta}, w} \mathcal{L}_{\text{train}}(\mathcal{F}(\widetilde{\theta}(x), W), y), \tag{3}$$

After optimization, the best policy $\widetilde{\theta}^*$ is sampled from $\mathcal{A}(\widetilde{\theta})$ and performs augmentation training on the actual tasks.

Although these proxy-based methods are fast and save many training costs, the weight sharing leads to a performance gap for the augmentation policy θ between the proxy and the actual tasks. Because the weights of different policy θ in the proxy task depend on each other and become deeply coupled. Recent studies [6] show that the gap will limit the performance improvement of these proxy-based methods.

Different from the proxy-based method using the proxy model to speed up the training speed, A^2-Aug avoids the massive overhead of the proxy-free method by significantly reducing the number of searches. Instead of searching on the proxy task, reducing the number of searches is another way to speed up the search process. Most automated augmentation methods ignore this because the current augmentation methods rely on many searches to fit the target dataset in the augmentation space. And a few explorations will lead to instability and limited performance improvement. The recent multi-scale methods [50, 51, 46, 42] have been shown robust regular effects via simultaneous multi-resolution input. This indicates that a small number of augmentations can also further fit the target dataset under the joint training paradigm. Therefore, A^2-Aug jointly optimizes multiple augmentations and uses their ensemble logit to enhance the adaptability of each augmentation for the target dataset, which leads to a significant improvement and less training overhead.

4. Adaptive automated data augmentation

In this section, we first introduce the A^2-Aug framework during training and inference in § 4.1. Then, we present the formulation and insights of our A^2-Aug in § 4.2. Next, we elaborate on the efficient training methods in A^2-Aug, including joint training with separate BatchNorm in § 4.3 and adaptive ensemble distillation in § 4.4.

4.1. Overview of A^2-Aug

The overview of the framework is shown in the Figure 2, including joint training with separate BatchNorm (S-BN) and adaptive ensemble distillation. During training, the images with multiple augmentations are trained with shared convolution/classifiers and separate BatchNorm. The adaptive ensemble logit is learned on the fly from each logit of input augmentation. Then the ensemble logit distills each output using KL divergence. After training, the ensemble distillation of multiple augmentations can be discarded, and the trained model of best augmentation can be used separately in model inference.

4.2. Formulation of A^2-Aug

As § 3 mentioned, A^2-Aug employs the proxy-free training paradigm, which is formulated in the Equation (1) and (2). Different from the other proxy-free method (*e.g.*, RandAugment [6]), we jointly optimize multiple candidate augmentations and use their adaptive ensemble logit to distill each augmentation for more performance gain, which mainly includes ensemble and distillation parts.

For the ensemble part, we first randomly sample N augmentations $(\theta_1, \theta_2, \cdots, \theta_N)$ from the search space $\mathcal{A}(\theta)$ and use them for augmenting training images to train the model $F(\cdot, w)$. Thus, there are multiple individual logits $\{\boldsymbol{Q}_i =$

$F(\theta_i(x), w_i) | i \in \{1, 2, \cdots, N\}\}$. Next, we obtain the adaptive ensemble logit $\boldsymbol{Q}_{ens} = \sum_{i=1}^{N} \mu_i \boldsymbol{Q}_i$ for the N logits using the weight factors $\boldsymbol{\mu} = \{\mu_1, \mu_2, \cdots, \mu_N\}$. The differentiable $\boldsymbol{\mu}$ is first normalized via softmax transformation and then optimized by label as:

$$\mu = \arg\min_{\mu} \mathcal{L}_{\text{train}}(\sum_{i=1}^{N} \mu_i F(\theta_i(x), w_i), y), \tag{4}$$

Note that the gradients of the logits $\boldsymbol{Q}_i$ are detached when optimizing $\boldsymbol{\mu}$. The weight factor $\boldsymbol{\mu}$ can reduce the weight of weak augmentation and strengthen the influence of beneficial augmentation. From the perspective of input augmentation, we obtain the implicit ensemble augmentation θ_{ens} in this process and θ_{ens} is optimized together with the target dataset in an online manner.

For the distillation part, each logit $\boldsymbol{Q}_i$ of different augmentations is allowed to mimic the ensemble logit $\boldsymbol{Q}_{ens}$ by optimizing the KL loss as $\mathcal{L}_{\text{kl}}\big(\boldsymbol{Q}_{ens}, \boldsymbol{Q}_i\big)$.

For the overall optimization objective, A^2-Aug mimics the classification loss of each logit $\boldsymbol{Q}_i$, the ensemble loss of $\boldsymbol{Q}_{ens}$ and the distillation loss, as:

$$\begin{aligned}(\widetilde{\mu}^*, W_{\widetilde{\mu}^*}) = \arg\min_{\widetilde{\mu}, w} \sum_{i=1}^{S} \{ & \underbrace{\mathcal{L}_{\text{train}}(\mathcal{F}(\theta_i(x), w_i), y)}_{\text{cross-entropy of each logit } \boldsymbol{Q}_i} \\ & \underbrace{+ \mathcal{L}_{\text{train}}((\sum_{i=1}^{S} \mu_i \mathcal{F}(\theta_i(x), w_i), y)}_{\text{cross-entropy of ensemble logit } \boldsymbol{Q}_{ens}} \\ & \underbrace{+ \mathcal{L}_{\text{train}}((\sum_{i=1}^{S} \mu_i \mathcal{F}(\theta_i(x), w_i), \mathcal{F}(\theta_i(x), w_i)\},}_{\text{KL divergence between ensemble } \boldsymbol{Q}_{ens} \text{ and each logit } \boldsymbol{Q}_i}\end{aligned} \tag{5}$$

where $W = [w_1, w_2, \cdots, w_N]$ is the weight of each network, μ^* represents the optimal weight factor $\boldsymbol{\mu}$ of multiple augmentations. From the Equation (5), the optimization of $\boldsymbol{\mu}$ means that the implicit ensemble augmentation θ_{ens} is dynamically updated to better adapt to the target dataset, which is similar to policy sampling in the Equation (3). The ensemble and distillation of multiple augmentations exist during training and disappear in the inference, which does not increase any inference overhead. After training, we choose the best policy θ_{best} for inference according to the accuracy on the validation set as the Equation (2).

Comparison with RandAugment. The Equation (1) and (5) demonstrate the difference between A^2-Aug and RandAugment. In A^2-Aug, different policies are distilled by the adaptive ensemble augmentation θ_{ens} defined in Equation (4). While each policy in RandAugment is only updated by label under independent training. As shown in Figure 3, the ensemble augmentation θ_{ens} performs stable improvements compared with each augmentation. With the distillation of θ_{ens}, the performance of the same policies under

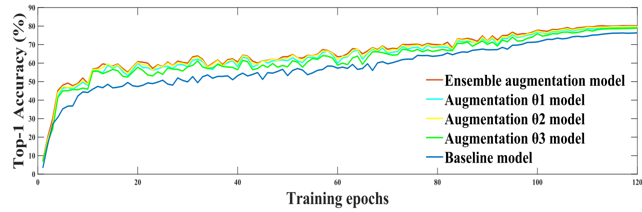

Figure 3: Training curves of ResNet-50 on ImageNet.

A^2-Aug training achieve significant gain compared to the baseline model. Although the best policy θ_{best} of A^2-Aug is searched from a small number of input augmentations, it enjoys similar adaptability with the best augmentation from a large number of searches for the target dataset under distillation training. This indicates that A^2-Aug does not require a grid search like RandAugment, which reduces costs and expands its application. Moreover, A^2-Aug is easy to implement with some efficient training approaches introduced in the following section.

4.3. Joint training with separate BatchNorm

To enhance training efficiency, each model $W = [w_1\, w_2, \cdots w_N]$ shares the weights of the convolution and classifiers, yet employs separate BatchNorm parameters. Therefore, the training for multiple augmentations can be conducted in a complete model, significantly speeding up the training process while having few affect on model accuracy. Note that A^2-Aug does not employ the same weight sharing strategy as the proxy-based augmentation method because the BN parameter in A^2-Aug is different for input augmentations. After training, we leverage the model with BatchNorm parameters of the best augmentation for model inference.

Separate BatchNorm. Batch Normalization (BN) is a widely used training technique that normalizes internal activation with the statistics of training batches. For many visual tasks (*e.g.*, domain adaptation [45], style transfer [20]), different data sources or data transformations have large gaps in BN parameters. Separate BatchNorm (S-BN) [53, 54, 46] is proved to be effective when parameter sharing for multiple data transformation. A^2-Aug uses S-BN to make the parameters of BN layers non-shareable between different augmentation strategies of inputs. The feature map of an image sample x at augmentation θ_i is normalized by:

$$\hat{\theta}_i(x) = \gamma_i \frac{\theta_i(x) - E[\theta_i(X)]}{\sqrt{Var[\theta_i(X)] + \epsilon}} + \beta_i, \tag{6}$$

where $\theta_i(X)$ is the batch of samples; γ_i and β_i are learnable scale and bias. Therefore, we have N disjoint set of parameters for each S-BN layer where N is the total number of augmentations. In the experiment section, S-BN can significantly improve the performance under multiple data

Table 2: The top-1 accuracy of different augmentation methods on CIFAR-10 and CIFAR-100. SS (26 $2 \times 32d$) refers to Shake-Shake (*i.e.*, the network has a depth of 26, 2 residual branches and the first residual block has a width of 32). **Gain** refers to the performance gain compared to the baseline. We report top-1 mean (std) accuracy (%) over 3 runs.

Dataset	Model	Baseline	CutOut [8]	AA [5]	PBA [16]	Fast AA [30]	RA [6]	DADA [29]	A^2-Aug	Gain
CIFAR-10	WRN-40-2	94.4	95.9	96.3	N/A	96.4	N/A	96.4	**96.7±0.2**	**2.3**
	WRN-28-10	96.1	96.9	97.4	97.4	97.3	97.3	97.3	**98.0±0.3**	**1.9**
	SS (26 $2 \times 32d$)	96.4	97.0	97.5	97.5	97.5	N/A	97.3	**97.7±0.1**	**1.3**
	SS (26 $2 \times 96d$)	97.1	97.4	98.0	98.0	98.0	98.0	98.0	**98.5±0.1**	**1.4**
	PyramidNet	97.3	NA	98.5	98.3	98.3	98.5	98.3	**98.9±0.2**	**1.6**
CIFAR-100	WRN-40-2	74.0	74.8	79.3	N/A	79.4	N/A	79.1	**81.3±0.3**	**7.3**
	WRN-28-10	81.2	81.6	82.9	83.3	82.7	83.3	82.5	**85.2±0.4**	**4.0**
	SS (26 $2 \times 96d$)	82.9	84.0	85.7	84.7	85.4	N/A	84.7	**86.8±0.3**	**3.9**

augmentation.

4.4. Adaptive ensemble distillation

Ensemble distillation [2, 25, 9] is a key part of A^2-Aug. We use an adaptive ensemble to optimize online ensemble augmentation and logit normalization to improve the performance of distillation.

Adaptive ensemble. Using the scaled sum of the N logits by the weight factors $\boldsymbol{\mu} = \{\mu_1, \mu_2, \cdots, \mu_N\}$ with softmax, we derive the adaptive ensemble logit $\boldsymbol{Q}_{ens}$ as:

$$\boldsymbol{Q}_{ens} = \sum_{i=1}^{N} \frac{e^{\mu_i}}{\sum_i e^{\mu_i}} \boldsymbol{Q}_i, \tag{7}$$

The Equation (7) shows that $\boldsymbol{\mu}$ acts as a dynamic controller and involves each augmentation strategy adaptively. For example, $\boldsymbol{\mu} = 1/N$ means that the ensemble logit $\boldsymbol{Q}_{ens}$ leverages the average of each logit. The $\boldsymbol{\mu}$ can be optimized on the fly with the gradient descent method on the training dataset. Moreover, we compare the average ensemble and adaptive ensemble in the experiment.

Logit normalization for distillation. We use ensemble logit $\boldsymbol{Q}_{ens}$ as the teacher model, which preforms distillation for each logit $\boldsymbol{Q}_i$ (*i.e.*, the student model) using Kullback Leibler (KL) divergence as $\mathcal{D}_{kl}(\boldsymbol{Q}_{ens}, \boldsymbol{Q}_i)$. However, the ensemble logit $\boldsymbol{Q}_{ens}$ is strong and may produce too large predicting probabilities. Therefore, the overconfidence of $\boldsymbol{Q}_{ens}$ may become label noise and harms the accuracy of $\boldsymbol{Q}_i$. Inspired by the previous work [43], we normalize the logits before performing KL divergence, which reduces the gap between each logit to get more performance improvements. The distillation loss $\mathcal{L}_{kd}$ is calculated as a summation of KL divergences:

$$\mathcal{L}_{kd} = \sum_{s=1}^{S} \mathcal{D}_{kl}\left(\frac{\boldsymbol{Q}_{ens}}{\|\boldsymbol{Q}_{ens}\|_2}, \frac{\boldsymbol{Q}_i}{\|\boldsymbol{Q}_i\|_2}\right). \tag{8}$$

where $\frac{\boldsymbol{Q}_{ens}}{\|\boldsymbol{Q}_{ens}\|_2}, \frac{\boldsymbol{Q}_i}{\|\boldsymbol{Q}_i\|_2}$ refer to the normalized distributions of $\boldsymbol{Q}_{ens}$ and $\boldsymbol{Q}_i$, which are helpful to improve the distillation effect.

5. Experiments

In this section, we first evaluate the proposed A^2-Aug on CIFAR-10 [23] and CIFAR-100 [23] in § 5.1 and ImageNet [7] in § 5.2, and compare the performance against existing data augmentation methods. For fair comparisons, we adopt the same training setting as AA [5], Fast AA [30], PBA [16] and RA [6] throughout the experiments. Then we isolate the influence of each element of A^2-Aug in § 5.3. All experiments are performed with PyTorch [36]. Full implementation details are referred to supplementary materials.

5.1. Experiments on CIFAR

Dataset. CIFAR-10 dataset consists of natural images with a size of 32×32. There are totally 60,000 images in 10 classes. Moreover, CIFAR-100 contains 50,000 training images and 10,000 test images with 100 classes, respectively.

Implementation. Following RA [6] and AA [5], we use the same training settings for different models, including weight decay, learning rate, batch size and total training epochs. For A^2-Aug, we randomly sample 4 augmentations from the same search space as RA. More implementation details, including detailed training and augmentation settings, are available in supplementary materials.

CIFAR-10 results. Table 2 shows the results of our experiments using A^2-Aug on Wide-ResNet-40-2 (WRN-40-2), Wide-ResNet-28-10 (WRN-28-10) [57] and Shake-Shake [10] models. Specifically, for WRN-40-2, our method obtains 2.3% absolute accuracy gain and outperforms AA with 0.4% obvious margins. Note that AA and CutOut are strong baselines for CIFAR-10. For WRN-28-10 with a wider channel, the accuracy of A^2-Aug is between 1.9% better than the baseline and $0.6\% \sim 1.1\%$ higher than other augmentation methods. For Shake-Shake with different widths/depths and PyramidNet, our method obtains $1.4\% \sim 1.6\%$ absolute accuracy gains and outperforms other methods with $0.3\% \sim 0.6\%$ obvious margins.

CIFAR-100 results. Different from CIFAR-100, CIFAR-100 is more challenging for more categories, and our method

Table 3: Top-1 accuracy of different augmentation methods on ImageNet for ResNet-50 and ResNet-200. Note that these results refer to the published report of the original papers. We report top-1 mean (std) accuracy (%) over 3 runs.

Model	Baseline	AA [5]	Fast AA [30]	RA [6]	OHL AA [31]	DADA [29]	A^2-Aug	Gain
ResNet-50	76.3	77.6	77.6	77.6	78.9	77.5	**79.2±0.4**	**2.9**
ResNet-200	78.5	80.0	80.6	N/A	N/A	N/A	**81.5±0.5**	**3.0**

Table 4: Search and training cost of different augmentation methods on ImageNet for ResNet-50. Search cost is the result of the original paper report. The training cost is estimated according to the training epoch reported in the original paper. The training cost of our method is measured on an 8× 2080Ti GPU server with a batch size of 1,024.

Method	Baseline	AA [5]	Fast AA [30]	OHL AA [31]	DADA [29]	RA [6]	A^2-Aug
Search cost (GPU-h)	0	15,000	450	625	1.3	0	**0**
Training cost (GPU-h)	384	640	576	400	576	10,368	**624**
Total cost (GPU-h)	384	15,640	1,026	1,025	577.3	10,368	**624**
Compared baseline	-	40.7×	2.7×	2.7×	1.5×	27×	**1.6**×

obtains more obvious performance improvements. As shown in Table 2, A^2-Aug achieves 7.3% absolute accuracy gain and outperforms AA with 2.0% obvious margins for WRN-40-2. For the high-capacity models such as WRN-28-10 and Shake-Shake (26 $2{\times}96d$), A^2-Aug achieve 4.0% and 3.9% accuracy gain compared to baseline.

5.2. Experiments on ImageNet

Dataset. We also perform experiments on the ImageNet dataset, known as the most challenging image classification dataset. It contains about 1.2 million training images and 50 thousand validation images, and each image belongs to one of 1,000 categories.

Implementation. Experiments are conducted on standard ResNet-50 and ResNet-200 with 120 training epochs, less than AA (200 epochs) and RA (180 epochs). A^2-Aug randomly samples 3 augmentations from the same search space as RA, which only introduces the training overhead of 1.6× compared to the baseline. Please refer to the supplementary materials for more details.

Results. Table 3 shows the top-1 accuracy of A^2-Aug compared to other augmentation methods for ResNet-50 [14] and ResNet-200 [14] models on ImageNet. For ResNet-50, our method achieves 2.9% absolute accuracy gains, which outperforms AA with 1.6% obvious margins. For ResNet-200, our method achieves 81.5% accuracy, which is the state-of-the-art performer in data augmentation methods.

Comparison on training efficiency. Besides the significant performance gains, A^2-Aug also enjoys considerable training efficiency. As shown in Table 4, A^2-Aug only introduces 1.6× training overhead compared to baseline and achieves 16× reduction in cost than RandAugment. Moreover, the A^2-Aug without search phase also presents less total cost than some proxy-based methods (*e.g.*, AA, FastAA and OHL AA) and obtains a 1.7% accuracy gain compared to DADA with a similar cost. Note that DADA is the fastest automated augmentation method so far.

5.3. Ablation study

In this section, we analyze the impact of the key parts of A^2-Aug, including the joint training, ensemble distillation method and multiple augmentations.

Efficiency of joint training with separate BatchNorm. To improve training efficiency, each model in A^2-Aug utilizes the same convolution/classifier weights and Separate BatchNorm (S-BN), preventing mutual interference. As shown in Table 5, the parameter sharing with S-BN improves the training speed by 1.8× without introducing loss of accuracy.

Table 5: Top-1 accuracy (%) and training cost (GPU-h) on ImageNet using ResNet-50 for A^2-Aug under independent training (ind.), A^2-Aug without S-BN and complete A^2-Aug. The training cost is measured on an 8× 2080Ti GPU server with a batch size of 1,024.

Method	Top-1	Total cost
A^2-Aug (ind.)	79.0%	1,136
A^2-Aug w/o S-BN	74.7%	616
A^2-Aug	79.2%	624

Importance of adaptive ensemble method. As the adaptive ensemble with weight factor μ is essential to A^2-Aug, we first study the learned value of μ and accuracy for each augmentation in our ImageNet experiment. As shown in the Table 6, the stronger augmentation possesses a larger weight factor, which verifies the effectiveness of the adaptive ensemble. Then we compare the average ensemble and adaptive ensemble in the Table 7. For A^2-Aug, the accuracy gain of the adaptive ensemble outperforms the average ensemble on the ImageNet dataset for ResNet-50.

Table 6: The learned value of weight factor $\boldsymbol{\mu}$ and top-1 accuracy (%) for each augmentation in the ImageNet.

Model	Learned values of $\boldsymbol{\mu}$			Top-1			
	μ_1	μ_2	μ_3	w_1	w_2	w_3	Ens
ResNet-50	0.36	0.49	0.15	79.0	79.2	78.6	79.5
ResNet-200	0.26	0.43	0.31	80.6	81.5	81.0	81.7

Table 7: Comparison of average ensemble method and adaptive ensemble method of ResNet-50 on ImageNet.

Method	Top-1	Gain
Baseline	76.3%	
Average ensemble	78.1%	1.8%
Adaptive ensemble	79.2%	2.9%

Effect of normalized distillation. In A^2-Aug, the ensemble logit distils each logit of multiple input augmentations to perform the augmentation training. As shown in Table 8, A^2-Aug without KL-divergence only outperforms baseline with 1.2%. The logit normalization can achieve 0.4% performance improvements for distillation.

Table 8: Contribution of KL-divergence and logit normalization on ImageNet for ResNet-50.

Method		Top-1	Gain
KL-divergence	Logit Norm.		
×	×	76.3%	
×	✓	77.5%	1.2%
✓	×	78.8%	2.5%
✓	✓	79.2%	2.9%

Sensitivity study on the number of multiple augmentations. We evaluate the different number of initial augmentations for A^2-Aug for 10 different seeds. In Figure 4, we observe that the number $4 \sim 5$ seems to be a considerable choice on CIFAR-100 . Furthermore, the performances under different seeds (see error bar in Figure 4) is small when there are more than 5 augmentations.

Compare to the multi-scale method. In the field of model design and compression, multi-scale methods (*e.g.*, MutualNet [51]) use multiple input resolutions to achieve dynamic accuracy-efficiency trade-offs at runtime. **However, our approach focuses on improving the performance of random candidates in automated data augmentation and has clear differences with these methods in the motivation and contribution**. Furthermore, we also compare A^2-Aug with the same multi-resolution and training settings with MutualNet in the Table 9. The result shows that A^2-Aug obtains 2.8% gains and outperforms MutualNet.

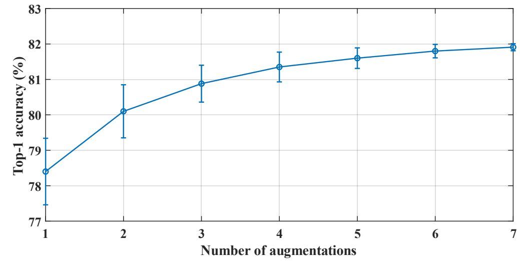

Figure 4: Effect of number of multiple augmentations for WRN-40-2 on CIFAR-100 with 10 different seeds.

Table 9: The results of A^2-Aug with multi-resolution inputs for ResNet-50 on ImageNet.

Method	Top-1	Gain
Baseline	76.3%	
MutualNet [51]	78.6%	2.3%
A^2-Aug (Multi-scale)	79.1%	2.8%

6. Conclusions

In this paper, we propose A^2-Aug, a simple and effective proxy-free framework. A^2-Aug can significantly improve performance and only need to search for a few augmentations. With the adaptive ensemble distillation of multiple augmentations, A^2-Aug can obtain similar effects compared to the grid search of RandAugment. Our method achieves state-of-the-art performance on CIFAR-10, CIFAR-100, and ImageNet via less training overhead. These improvements and perspectives show a novel and potential method. We hope this elegant and practical approach would facilitate the research for data augmentation.

Acknowledgment

This work is supported by the National Key R&D Program of China 2019YFB1311100, the Scientific Instrument Developing Project of the Chinese Academy of Sciences (No. YJKYYQ20200045), the National Key Research and Development Program of China under Grant 2017YFA0700802, in part by the National Natural Science Foundation of China under Grant 61822603, Grant U1813218, and Grant U1713214, in part by a grant from the Beijing Academy of Artificial Intelligence (BAAI), and in part by a grant from the Institute for Guo Qiang, Tsinghua University. We thank Aojun Zhou, Yikai Wang, Anbang Yao and Yiming Hu for valuable help.

References

[1] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *ICLR*, 2018.

[2] Defang Chen, Jian-Ping Mei, Can Wang, Yan Feng, and Chun Chen. Online knowledge distillation with diverse peers. In *AAAI*, pages 3430–3437, 2020.

[3] Pengguang Chen, Shu Liu, Hengshuang Zhao, and Jiaya Jia. Gridmask data augmentation. *ArXiv*, abs/2001.04086, 2020.

[4] Jaehoon Choi, Tae-Kyung Kim, and Changick Kim. Self-ensembling with gan-based data augmentation for domain adaptation in semantic segmentation. *ICCV*, 2019.

[5] E. Cubuk, Barret Zoph, Dandelion Mané, V. Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation strategies from data. *CVPR*, 2019.

[6] Ekin. D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space. *CVPR*, 2020.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[8] Terrance Devries and Graham. W. Taylor. Improved regularization of convolutional neural networks with cutout. *ArXiv*, abs/1708.04552, 2017.

[9] Shangchen Du, Shan You, Xiaojie Li, Jianlong Wu, Fei Wang, Chen Qian, and Changshui Zhang. Agree to disagree: Adaptive ensemble knowledge distillation in gradient space. In *NeurIPS*, 2020.

[10] Xavier Gastaldi. Shake-shake regularization. *ArXiv*, abs/1705.07485, 2017.

[11] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[12] Jia Guo, Minghao Chen, Yao Hu, Chen Zhu, X. He, and Deng Cai. Reducing the teacher-student gap via spherical knowledge disitllation. *arXiv: 2010.07485*, 2020.

[13] Himel Das Gupta, Kun Zhang, and V. Sheng. A simple ensemble learning knowledge distillation. In *MLIS*, 2020.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[16] Daniel Ho, Eric Liang, Ion Stoica, Pieter Abbeel, and Xi Chen. Population based augmentation: Efficient learning of augmentation policy schedules. In *ICML*, 2019.

[17] Shoukang Hu, Si rui Xie, Hehui Zheng, Chunxiao Liu, Jianping Shi, Xunying Liu, and Dahua Lin. Dsnas: Direct neural architecture search without parameter retraining. *ArXiv*, abs/2002.09128, 2020.

[18] Shou-Yong Hu, Sirui Xie, Hehui Zheng, Chunxiao Liu, Jianping Shi, Xunying Liu, and Dahua Lin. Dsnas: Direct neural architecture search without parameter retraining. 2020.

[19] Yiming Hu, Xingang Wang, Lujun Li, and Qingyi Gu. Improving one-shot nas with shrinking-and-expanding supernet. *Pattern Recognition*, 118:108025, 2021.

[20] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. *ICCV*, 2017.

[21] Alex Kantchelian, J. Tygar, and A. Joseph. Evasion and hardening of tree ensemble classifiers. In *ICML*, 2016.

[22] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[23] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Tech Report*, 2009.

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

[25] Xu Lan, Xiatian Zhu, and Shaogang Gong. Knowledge distillation by on-the-fly native ensemble. In *NeurIPS*, 2018.

[26] Hankook Lee, Sung Ju Hwang, and Jinwoo Shin. Rethinking data augmentation: Self-supervision and self-distillation. In *ICML*, 2020.

[27] Bo-Yi Li, Felix Wu, Ser-Nam Lim, Serge J. Belongie, and Kilian Q. Weinberger. On feature normalization and data augmentation. *ArXiv*, abs/2002.11102, 2020.

[28] Hanhan Li, Joe Yue-Hei Ng, and Paul Natsev. Ensemblenet: End-to-end optimization of multi-headed models. *ArXiv*, abs/1905.09979, 2019.

[29] Yonggang Li, Guosheng Hu, Yongtao Wang, Timothy Hospedales, Neil Robertson, and Yongxin Yang. Dada: Differentiable automatic data augmentation. *ArXiv*, abs/2003.03780, 2020.

[30] Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast autoaugment. In *NeurIPS*, 2019.

[31] Chen Lin, Minghao Guo, Chuming Li, Wei Wu, Dahua Lin, Wanli Ouyang, and Junjie Yan. Online hyper-parameter learning for auto-augmentation strategy. *ICCV*, 2019.

[32] Xuanqing Liu, Minhao Cheng, Huan Zhang, and C. Hsieh. Towards robust neural networks via random self-ensemble. In *ECCV*, 2018.

[33] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[34] Andrey Malinin, Bruno Mlodozeniec, and Mark Gales. Ensemble distribution distillation. *ArXiv*, abs/1905.00076, 2020.

[35] Wonpyo Park, Yan Lu, Minsu Cho, and Dongju Kim. Relational knowledge distillation. In *CVPR*, 2019.

[36] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NeurIPS Workshop*, 2017.

[37] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *ArXiv*, abs/1712.04621, 2017.

[38] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *ICML*, 2018.

[39] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *ICLR*, 2015.

[40] Ryo Takahashi, Takashi Matsubara, and K. Uehara. Data augmentation using random image cropping and patching for deep cnns. *TCSVT*, 2020.

[41] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. In *ICLR*, 2020.

[42] Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Herve Jegou. Fixing the train-test resolution discrepancy. In *NeurIPS*, 2019.

[43] Feng Wang, Xiang Xiang, Jian Cheng, and A. Yuille. Normface: L2 hypersphere embedding for face verification. *ACM MM*, 2017.

[44] Huan Wang, Suhas Lohit, Michael Jones, and Yun Fu. Knowledge distillation thrives on data augmentation. *ArXiv*, abs/2012.02909, 2020.

[45] Ximei Wang, Y. Jin, Mingsheng Long, J. Wang, and Michael I. Jordan. Transferable normalization: Towards improving transferability of deep neural networks. In *NeurIPS*, 2019.

[46] Yikai Wang, Fuchun Sun, Duo Li, and Anbang Yao. Resolution switchable networks for runtime efficient image recognition. In *ECCV*, 2020.

[47] Qizhe Xie, Zihang Dai, E. Hovy, Minh-Thang Luong, and Quoc V. Le. Unsupervised data augmentation. *ArXiv*, abs/1904.12848, 2019.

[48] Qizhe Xie, Le Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. Self-training with noisy student improves imagenet classification. *ArXiv*, abs/1911.04252, 2019.

[49] Kunran Xu, Lai Rui, Yishi Li, and Lin Gu. Feature normalized knowledge distillation for image classification. In *ECCV*, 2020.

[50] Taojiannan Yang, Sijie Zhu, and Chen Chen. Gradaug: A new regularization method for deep neural networks. *ArXiv*, abs/2006.07989, 2020.

[51] Taojiannan Yang, S. Zhu, Chen Chen, Shen Yan, Mi Zhang, and A. Willis. Mutualnet: Adaptive convnet via mutual learning from network width and resolution. In *ECCV*, 2020.

[52] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *CVPR*, 2017.

[53] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. *ArXiv*, abs/1812.08928, 2019.

[54] Zhuoran Yu, Aojun Zhou, Yukun Ma, Yudian Li, Xiaohan Zhang, and P. Luo. Scale calibrated training: Improving generalization of deep networks via scale-specific normalization. *arXiv:1909.00182*, 2019.

[55] Kaiyu Yue, Jiangfan Deng, and Feng Zhou. Matching guided distillation. In *ECCV*, 2020.

[56] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. *ICCV*, 2019.

[57] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016.

[58] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *ICLR*, 2017.

[59] Hongyi Zhang, Moustapha Cisse, Yann Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *ArXiv*, abs/1710.09412, 2018.

[60] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *CVPR*, 2018.