**A²-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 (\(\sim 5\) vs 100 of RandAugment) to alleviate these issues. Specifically, we propose **Adaptive Automated Augmentation** (A²-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, A²-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, A²-Aug obtains state-of-the-art performance with less training budget. In experiments, our A²-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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Total cost (GPU-h)</th>
<th>CIFAR-10</th>
<th>CIFAR-100</th>
<th>ImageNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>384</td>
<td>97.3</td>
<td>81.2</td>
<td>76.3</td>
</tr>
<tr>
<td>AA</td>
<td>15,640</td>
<td>98.5</td>
<td>82.9</td>
<td>77.6</td>
</tr>
<tr>
<td>Fast AA</td>
<td>1,026</td>
<td>98.3</td>
<td>82.7</td>
<td>77.6</td>
</tr>
<tr>
<td>PBA</td>
<td>N/A</td>
<td>98.3</td>
<td>83.3</td>
<td>N/A</td>
</tr>
<tr>
<td>RA</td>
<td>10,368</td>
<td>98.5</td>
<td>83.3</td>
<td>77.6</td>
</tr>
<tr>
<td>A²-Aug</td>
<td>624</td>
<td>98.9</td>
<td>85.2</td>
<td>79.2</td>
</tr>
</tbody>
</table>

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...
We find that misclassified input samples under one augmentation are more likely to be correctly classified by other augmentations. Therefore, Adaptive AugNet can improve the top-1 accuracy of AutoAugNet by 1.6% on CIFAR-10 and 4.0% on CIFAR-100. On the challenging ImageNet dataset, Adaptive AugNet 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 A²-Aug, a new proxy-free automated augmentation framework that only needs a limited number of searches. A²-Aug enjoys a better trade-off between performance and computation than conventional proxy-free methods, significantly boosting its application.

- A²-Aug maximizes the ensemble logit of multiple augmentations to improve the adaptability of each candidate augmentation.
data 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] employs 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 task [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 ensembles [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.

**Knowledge 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 \( \theta \) 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 \( \mathcal{X}_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{N_{\text{train}}} \), and validation dataset \( \mathcal{X}_{\text{val}} = \{(x_i^*, y_i^*)\}_{i=1}^{N_{\text{val}}} \), the purpose of automated augmentation is to find the optimal policy \( \theta \). 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:

\[
\theta^* = \arg \max_{\theta \in \mathcal{A}} \text{ACC}_{\text{val}}(w^*_\theta),
\]

where \( \text{ACC}_{\text{val}}(\cdot) \) refers to the accuracy on the validation set. For different policy \( \theta \) 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 (\( 10^3 \)) 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 \( \theta \) 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 \( \mathcal{A}(\theta) \).
where $\tilde{\theta}$ denotes the continuous policies that represent the distribution of the $\theta$. In such a continuous search space, augmentation policy $\tilde{\theta}$ can be flexibly sampled from $A(\tilde{\theta})$ and optimized together with weights, as

$$
(\tilde{\theta}^*, W_{\tilde{\theta}^*}) = \arg \min_{\tilde{\theta}, W} L_{\text{train}}(F(\tilde{\theta}(x), W), y), \quad (3)
$$

After optimization, the best policy $\tilde{\theta}^*$ is sampled from $A(\tilde{\theta})$ 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 $\theta$ between the proxy and the actual tasks. Because the weights of different policy $\theta$ 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 ($\theta_1, \theta_2, \ldots, \theta_N$) from the search space $A(\theta)$ and use them for augmenting training images to train the model $F(\cdot, w)$. Thus, there are multiple individual logits $\{Q_i = \}$.
\( F(\theta_i(x), w_i) \mid i \in \{1, 2, \cdots, N\} \). Next, we obtain the adaptive ensemble logit \( Q_{\text{ens}} = \sum_{i=1}^{N} \mu_i Q_i \) for the \( N \) logits using the weight factors \( \mu = \{\mu_1, \mu_2, \cdots, \mu_N\} \). The differentiable \( \mu \) is first normalized via softmax transformation and then optimized by label as:

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

(4)

Note that the gradients of the logits \( Q_i \) are detached when optimizing \( \mu \). The weight factor \( \mu \) 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 \( \theta_{\text{ens}} \) by label under independent training. As shown in Figure 3, \( \mathcal{A} \) RandAugment. In the validation set as the Equation (2).

For the distillation part, each logit \( Q_i \) of different augmentations is allowed to mimic the ensemble logit \( Q_{\text{ens}} \) by optimizing the KL loss as \( \mathcal{L}_{\text{U}}(Q_{\text{ens}}, Q_i) \).

For the overall optimization objective, \( \mathcal{A}^2 \)-Aug mimics the classification loss of each logit \( Q_i \), the ensemble loss of \( Q_{\text{ens}} \) and the distillation loss, as:

\[
(\hat{\mu}^*, W^*) = \arg \min_{\hat{\mu}, W} \sum_{i=1}^{S} \left( \mathcal{L}_{\text{aux}}(F(\theta_i(x), w_i), y) \right. \\
+ \mathcal{L}_{\text{train}}( \sum_{i=1}^{S} \mu_i F(\theta_i(x), w_i), y) \\
+ \mathcal{L}_{\text{train}}( \sum_{i=1}^{S} \mu_i F(\theta_i(x), w_i), F(\theta_i(x), w_i) \right),
\]

(5)

where \( W = [w_1, w_2, \cdots, w_N] \) is the weight of each network, \( \hat{\mu}^* \) represents the optimal weight factor \( \mu \) of multiple augmentations. From the Equation (5), the optimization of \( \mu \) means that the implicit ensemble augmentation \( \theta_{\text{ens}} \) 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 \( \theta_{\text{best}} \) 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 \( \mathcal{A}^2 \)-Aug and RandAugment. In \( \mathcal{A}^2 \)-Aug, different policies are distilled by the adaptive ensemble augmentation \( \theta_{\text{ens}} \) 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 \( \theta_{\text{ens}} \) performs stable improvements compared with each augmentation. With the distillation of \( \theta_{\text{ens}} \), the performance of the same policies under

\[
\hat{\theta}_i(x) = \gamma_i \frac{\theta_i(x) - E[\theta_i(X)]}{\sqrt{\text{var}[\theta_i(X)]}} + \beta_i,
\]

(6)

where \( \theta_i(X) \) is the batch of samples; \( \gamma_i \) and \( \beta_i \) 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...
We use an adaptive ensemble to optimize online ensemble where we derive the adaptive ensemble logit \( Q \). Adaptive ensemble.

The distillation loss \( L \) is calculated as a summation of KL divergence, which reduces the gap before performing KL divergence, which reduces the gap. Inspired by the previous work \([43]\), we normalize the logits predicting probabilities. Therefore, the overconfidence of ensemble logit \( Q \) may become label noise and harms the accuracy of 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 \( \mu = \{\mu_1, \mu_2, \cdots, \mu_N\} \) with softmax, we derive the adaptive ensemble logit \( Q_{ens} \) as:

\[
Q_{ens} = \sum_{i=1}^{N} \frac{e^{\mu_i}}{\sum_{i=1}^{N} e^{\mu_i}} Q_i.
\]  

(7)

The Equation (7) shows that \( \mu \) acts as a dynamic controller and involves each augmentation strategy adaptively. For example, \( \mu = 1/N \) means that the ensemble logit \( Q_{ens} \) leverages the average of each logit. The \( \mu \) 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 \( Q_{ens} \) as the teacher model, which preforms distillation for each logit \( Q_i \) (i.e., the student model) using Kullback Leibler (KL) divergence as \( D_{kl}(Q_{ens}, Q_i) \). However, the ensemble logit \( Q_{ens} \) is strong and may produce too large predicting probabilities. Therefore, the overconfidence of \( Q_{ens} \) may become label noise and harms the accuracy of \( Q_i \).

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 \( L_{kd} \) is calculated as a summation of KL divergences:

\[
L_{kd} = \sum_{s=1}^{S} D_{kl}(\frac{Q_{ens}}{\|Q_{ens}\|_2}, \frac{Q_i}{\|Q_i\|_2}).
\]  

(8)

where \( \frac{Q_{ens}}{\|Q_{ens}\|_2}, \frac{Q_i}{\|Q_i\|_2} \) refer to the normalized distributions of \( Q_{ens} \) and \( Q_i \), 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 \( 32 \times 32 \). 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% ∼ 1.1% higher than other augmentation methods. For Shake-Shake with different widths/depths and PyramidNet, our method obtains 1.4% ∼ 1.6% absolute accuracy gains and outperforms other methods with 0.3% ∼ 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.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-50</td>
<td>76.3</td>
<td>77.6</td>
<td>77.6</td>
<td>77.6</td>
<td>78.9</td>
<td>77.5</td>
<td>79.2±0.4</td>
<td>2.9</td>
</tr>
<tr>
<td>ResNet-200</td>
<td>78.5</td>
<td>80.0</td>
<td>80.6</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>81.5±0.5</td>
<td>3.0</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Search cost (GPU-h)</td>
<td>0</td>
<td>15,000</td>
<td>450</td>
<td>625</td>
<td>1.3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10,368</td>
<td>10,368</td>
<td>-</td>
<td>0</td>
<td>10,368</td>
<td>10,368</td>
</tr>
<tr>
<td>Training cost (GPU-h)</td>
<td>384</td>
<td>640</td>
<td>576</td>
<td>400</td>
<td>576</td>
<td>10,368</td>
<td>624</td>
<td>400</td>
<td>577.3</td>
<td>10,368</td>
<td>-</td>
<td>10,368</td>
<td>624</td>
<td></td>
</tr>
<tr>
<td>Total cost (GPU-h)</td>
<td>384</td>
<td>15,640</td>
<td>1,026</td>
<td>1,025</td>
<td>577.3</td>
<td>10,368</td>
<td>624</td>
<td>400</td>
<td>577.3</td>
<td>10,368</td>
<td>-</td>
<td>10,368</td>
<td>624</td>
<td></td>
</tr>
</tbody>
</table>

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 ×96 d), $A^2$-Aug achieves 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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Top-1</th>
<th>Total cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A^2$-Aug (ind.)</td>
<td>79.0%</td>
<td>1.136</td>
</tr>
<tr>
<td>$A^2$-Aug w/o S-BN</td>
<td>74.7%</td>
<td>616</td>
</tr>
<tr>
<td>$A^2$-Aug</td>
<td>79.2%</td>
<td>624</td>
</tr>
</tbody>
</table>

Importance of adaptive ensemble method. As the adaptive ensemble with weight factor $\mu$ is essential to $A^2$-Aug, we first study the learned value of $\mu$ 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 $\mu$ and top-1 accuracy (%) for each augmentation in the ImageNet.

<table>
<thead>
<tr>
<th>Model</th>
<th>Learned values of $\mu$</th>
<th>Top-1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\mu_1$</td>
<td>$\mu_2$</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>0.36</td>
<td>0.49</td>
</tr>
<tr>
<td>ResNet-200</td>
<td>0.26</td>
<td>0.43</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Top-1</th>
<th>Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>76.3%</td>
<td></td>
</tr>
<tr>
<td>Average ensemble</td>
<td>78.1%</td>
<td>1.8%</td>
</tr>
<tr>
<td>Adaptive ensemble</td>
<td>79.2%</td>
<td>2.9%</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Method</th>
<th>KL-divergence</th>
<th>Logit Norm.</th>
<th>Top-1</th>
<th>Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>×</td>
<td>×</td>
<td>76.3%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>×</td>
<td>✓</td>
<td>77.5%</td>
<td>1.2%</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>×</td>
<td>78.8%</td>
<td>2.5%</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>✓</td>
<td>79.2%</td>
<td>2.9%</td>
</tr>
</tbody>
</table>

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

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


[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.


