Augmentation Invariant Training

Weicong Chen
Tsinghua University
chenwc18@mails.tsinghua.edu.cn

Lu Tian
Tsinghua University
Xilinx
tl.september@gmail.com

Liwen Fan
Zhihu
vincentfan.china@gmail.com

Yu Wang
Tsinghua University
yu-wang@mail.tsinghua.edu.cn

Abstract

Data augmentation is acknowledged to help deep neural networks generalize better, while their augmentation transfer ability is far from satisfactory. The networks perform worse when tested with augmentations not used during training, which is also a manifestation of insufficient generalization ability. To address this problem, we carefully design a novel Augmentation invariant Loss (AiLoss) to assist networks to learn augmentation invariant by minimizing intra-augmentation variation. Based on AiLoss, we propose a simple yet efficient training strategy, Augmentation Invariant Training (AIT), to enhance the generalization ability of networks. Extensive experiments show that AIT can be applied to a variety of network architectures, and consistently improve their performance on CIFAR-10, CIFAR-100 and ImageNet without increasing computational cost. Further extending AIT to multiple networks, we propose multi-AIT to learn inter-network augmentation invariant, which achieves better performance in enhancing generalization ability. Moreover, further experiments present that networks trained with our strategy do obtain better augmentation transfer ability and learn features that are invariant to augmentations. Our source code is available at Github[1].

1. Introduction

Data augmentation is widely used to reduce the generalization error of neural networks in many machine learning tasks. Commonly used data augmentation approaches in computer vision include horizontally flipping, random crop, grayscale, etc. Through these ways, new variants can be synthesized from original training data samples.

[1]https://github.com/juicecwc/Augmentation-Invariant-Training

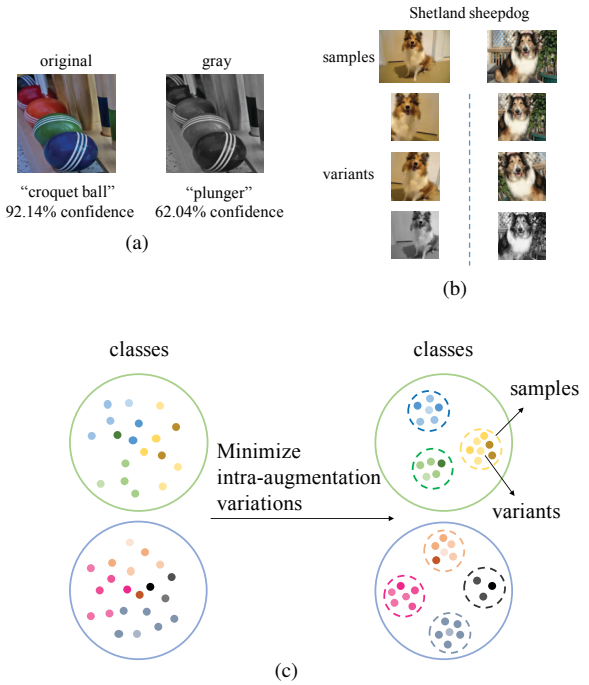


Figure 1. (a) presents an example of the poor augmentation transfer ability of neural network. We select an image from the 522-th class "croquet ball" of ImageNet, and obtain the left image after applying random crop to it. Further implementing grayscale delivers the right one. The results below the images are the predictions of the same network trained with only random crop augmentation. (b) shows the gap between variants generated from different samples. The samples are chosen from the 230-th class "Shetland sheepdog" of ImageNet, and we use horizontally flipping, random crop, and color jittering to generate the variants. (c) compares the feature distribution before and after minimizing intra-augmentation variation. The big circle means a class, and the small solid ones of the same color but different shades represent variants generated from the identical sample, while the dotted circles in the right are distinct samples.

Conventional training strategy for data augmentation is merely adding the variants to training data and optimizing the network to predict the same label as their original samples. However, the network does not obtain good augmentation transfer ability through this way [8]. For example, a network trained with random crop augmentation may perform well on a test set that mainly contains spatial distortion compared to the training set. However, if we add other augmentations to the test set such as color jittering, the performance may drop a lot. As shown in Figure 1(a), the Resnet50 [4] model trained on ImageNet [18] using only random crop augmentation predicts the right label for the original image, while concludes a wrong result with high confidence for the grayscale image. This gap of augmentation between training data and test data is quite common, including different lighting conditions, color space, spatial shape, backgrounds and so on. Therefore, poor augmentation transfer ability can harm the performance in the real-world application heavily.

A natural idea to improve augmentation transfer ability is applying the kinds of augmentations as various as possible during training, then the network may generalize to all augmentations. However, [2] presents that a proper combination of augmentations rather than a large number of them is better. Then how can network improve augmentation transfer ability? This is a key challenge in deep learning.

Before we formally discuss how to address this problem, let us scrutinize an observation first. Compared to other samples of the same class, variants are much more similar to their original samples. As shown in Figure 1(b), there is an apparent gap between the left three variants and the right ones, while it is totally ignored by the conventional training strategy. This gap implies that there is some invariant in the variants which distinguishes variants generated from different samples, and we call it augmentation invariant. Augmentation invariant is an important property of data distribution, since it describes the association between variants and their original samples. Augmentation invariant mainly includes invariance related to data augmentation, such as spatial invariance and color invariance. Intuitively, the networks which have learned augmentation invariant are more robust to the variation of augmentations, and thus have better augmentation transfer ability.

To this end, we introduce a novel loss function called Augmentation invariant Loss (AiLoss) to help networks learn augmentation invariant. This loss function is designed based on the intuition that the key to learning augmentation invariant is minimizing intra-augmentation variation. And the intuition is explained in Figure 1(c). The left big circle illustrates the feature distribution of variants before minimizing intra-augmentation variation, where the variants are mixed together, while the right one is the desired distribution after applying AiLoss. The variants belonging to the same sample are clustered together, and there is an apparent gap between different samples.

Based on AiLoss, we propose an efficient training strategy to help the network learn augmentation invariant. As presented in Figure 2, the variants are firstly inputted into the network in a pairwise way. Then AiLoss is applied to jointly supervise the training together with the original Softmax Loss. Through this way, the network is optimized to not only predict the right label, but also minimize intra-augmentation variation, and thus the augmentation transfer ability is enhanced. We call this strategy Augmentation Invariant Training (AIT). The implementation of AIT is straightforward. It requires no changes to the data or network, and introduces no additional computational overhead.

Our contributions can be summarized as follows:

(1) We propose to enhance the augmentation transfer ability of networks by learning augmentation invariant. Augmentation transfer ability represents the performance of generalizing to augmentations not seen during training. Augmentation invariant defines the association among different variants, which are generated by the same sample using random data augmentations. Networks that have learned augmentation invariant are more robust to the variation of augmentations, and thus can obtain better augmentation transfer ability.

(2) A novel AiLoss is designed to learn augmentation invariant by minimizing intra-augmentation variation. Based on AiLoss, we propose a simple yet efficient training strategy named AIT. Extensive experiments carried out on CIFAR-10, CIFAR-100 and ImageNet show that AIT can improve the generalization ability of networks significantly. Further extending to multiple networks, we propose multi-AIT to learn inter-network augmentation invariant. Multi-AIT shows better performance in enhancing generalization ability and is also promising to train small student networks.

(3) Further experiments demonstrate that networks trained with our algorithm do obtain better augmentation transfer ability. AIT outperforms baseline when tested with augmentations not seen during training. The visualization of feature distributions also shows AiLoss can help networks learn features that are more discriminative and invariant to augmentations.

2. Related Work

Invariant Learning. Other related works on invariant learning include [11] [7] [3] [6] [24] [23] [20] [16]. Pooling [11] down-samples an input representation, and makes the output feature invariant to scale or orientation changes. STN [7] proposes a new module to learn spatial invariant, such as pose, scale and warping. Deformable convolution [3] is also introduced to learn dense spatial transformation, which samples the feature map in a local and dense manner rather than global transformation. Dropout [20]

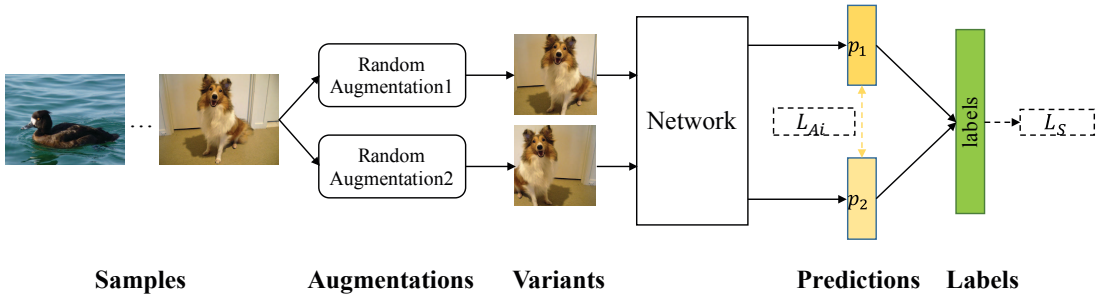


Figure 2. Augmentation Invariant Training (AIT) schmatic. Each sample generates two variants after random augmentations, and their probability distributions predicted by the network are p_1 and p_2 respectively. AiLoss (L_{Ai}) measures the similarity between p_1 and p_2, and jointly supervises the training of the network with Softmax Loss (L_S).

randomly drops some values during training and thus can reduce the effects of sample noise. Spectral Norm [24], IN [23] and IBN-Net [16] can capture and eliminate appearance variance, such as colors and styles. Different from the above works, this paper does not design any complicated module or architecture, but proposes to learn augmentation invariant from data distribution itself by carefully designing loss function and training strategy.

Generalization Ability. Generalization is the ability of the network to fit unseen samples. According to the data distribution difference of training data and test data, generalization can be divided into intra-domain generalization and cross-domain generalization. Previous works typically improve the intra-domain generalization ability by reducing overfitting. Data augmentation [11] [15] [27], regularization [12] [6] and noise injection [20] [1] are common techniques. For cross-domain generalization, transfer learning methods such as finetuning, domain adaption [21] [14] and domain generalization [9] are mainly studied. Our algorithm belongs to the former, since augmentations will not change the domain of data. And AiLoss is actually a regularization term, which can help reduce overfitting.

3. Approaches

3.1. Augmentation invariant Loss (AiLoss)

3.1.1 Formulation

We firstly formulate the conventional training process with data augmentation. Given a sample x from class y, with $y \in \{1, 2, ..., c\}$, a variant $v = F(x, seed)$ is obtained by applying random augmentation $F(\cdot, seed)$ to x, where $seed$ means the random seed. Then the variant is fed into the network $\Theta(\cdot, \theta)$ with trainable parameter θ to output the prediction as $p = \Theta(F(x, seed), \theta)$. Adopting Softmax Loss as the classification loss function, the final loss can be calculated as $l = softmax(y, p)$. And the optimization of this network is to minimize l, in other words, it maximizes posterior probability of the groundtruth class.

For two variants v_1 and v_2 generated from the same sample, they share the same label y, whose predictions are p_1 and p_2 respectively. Under the supervision of Softmax Loss, $\arg\max_k(p_1^k)$ and $\arg\max_k(p_2^k)$ are optimized to be equal to y. In this case, $\arg\max_k(p^k)$ is invariant for variants generated from the same sample, where p^k denotes the k-th value of p. However, for other variants from another sample with the same label, they are also optimized to predict class y. In other words, Softmax Loss treats the variants from the identical class equally and does not take advantage of the association between the variants and their original samples.

In order to minimize intra-augmentation variation, it is reasonable to intend to optimize $p1$ and $p2$ to be more similar. To this end, we adopt symmetric KL divergence to measure the similarity between two probability distributions:

$$\begin{aligned} D(p_1, p_2) &= KL(p_1 \parallel p_2) + KL(p_2 \parallel p_1) \\ &= \sum_{k=1}^{c}(p_1^k log\frac{p_1^k}{p_2^k} + p_2^k log\frac{p_2^k}{p_1^k}). \end{aligned} \tag{1}$$

$D(p_1, p_2)$ measures the logarithmic difference between p_1 and p_2, and it is always non-negative, with $D(p_1, p_2) = 0$ if and only if $p_1 = p_2$. Since this difference can be seen as an error in classification, regarding minimizing it as training target is reasonable. So we directly adopt this formulation as our AiLoss

$$L_{Ai} = D(p_1, p_2). \tag{2}$$

In this way, each variant is trained to match the probability distribution with its peer throughout the whole training process.

We adopt the joint supervision of Softmax Loss and AiLoss to train deep neural networks:

$$L = L_S + \lambda L_{Ai}, \tag{3}$$

where $L_s = softmax(y, p_1) + softmax(y, p_2)$ and λ is the loss weight to balance the two loss functions.

3.1.2 Discussion

Actually, in addition to minimizing the variation between p_1 and p_2, many other optional methods are also able to achieve our purpose. For instance, the outputs of the penultimate layer, known as features, can be optimized to be more similar as well. Denoting f_1 and f_2 as the features of v_1 and v_2, the similarity of f_1 and f_2 can be measured by Euclidean distance, formulated as $D(f_1, f_2)$. Theoretically, minimizing $D(f_1, f_2)$ can also reduce intra-augmentation variation, and AiLoss is changed to $D(f_1, f_2)$ in this case. Moreover, the outputs of other layers are also worth trying, as well as other similarity metrics.

All the situations above have been considered, and extensive experiments are conducted to find out the best formulation. According to the experiment results in Section 4.2.1, AiLoss is finally formulated as the KL divergence of predictions.

3.2. Augmentation Invariant Training (AIT)

3.2.1 Basic Structure

The structure of AIT is presented in Figure 2. During training, the variants generated by augmentations $F_1(x, seed_1)$ and $F_2(x, seed_2)$ are fed into the network in a pairwise way. Under the joint supervision of Softmax Loss and AiLoss, the network tends to predict the correct label for each variant and minimize the variation between the two predictions at the same time. If not specified, F_1 and F_2 are set to be identical.

3.2.2 Extension to more networks

Inspired by Deep Mutual Learning (DML) [28], we adopt a similar structure to extend AIT to multiple networks, denoted as multi-AIT. In DML, a cohort of networks are trained collaboratively and learn from each other during the training phrase. Similarly, multi-AIT trains a group of networks together with their inputs being different variants generated from the same sample.

The key distinction between AIT and multi-AIT is that variants in the former share network, while those in the latter don't. Following the formulation in Section 3.1.1, the variants v_1 and v_2 are fed into two networks $\Theta_1(x, \theta_1)$ and $\Theta_2(x, \theta_2)$ respectively in multi-AIT. And the predictions are reformulated as $p_1 = \Theta_1(F_1(x, seed_1), \theta_1)$ and $p_2 = \Theta_2(F_2(x, seed_2), \theta_2)$. Subsequent loss function formulation remains unchanged.

Through this way, multi-AIT combines the advantages of AIT and DML. Networks are trained to make consistent predictions between both variants and their peer networks. Thus they can learn inter-network augmentation invariant.

4. Experiments

4.1. Datasets and Settings

CIFAR-10 & CIFAR-100: For samples in CIFAR-10 [10] and CIFAR-100 [10], following [13], we apply 4 pixels zero padding on each side, and then randomly crop out a 32×32 patch from the padded sample or its horizontal flip. Those 32×32 patches are the variants fed to into network during the training phase. At the testing stage, no augmentation is applied to the original samples. In order to verify the generalization of our algorithm, three different kinds of networks including MobilenetV2 [19], Resnet32, and Wide Resnet WRN-28-10 [25] are adopted. We use SGD with momentum of 0.9 to train the network, and set weight decay as 0.0001 except WRN-28-10, for which weight decay is set to 0.0005 so as to reproduce the baseline accuracy described in their paper. The initial learning rate is 0.1 and divided by 10 at 80 and 120 epoch. The models are trained for 160 epoches with a mini-batch size as 128.

ImageNet: Our augmentations for ImageNet mainly follow the practice in [4]. Firstly, a crop of size randomly sampled from 0.08 to 1.0 of the original size and an aspect ratio randomly sampled from 3/4 to 4/3 of the original aspect ratio is made from the original sample or its horizontal flip. Then we resize that crop to 224×224 and generate a variant of the sample. In testing, the sample is firstly resized to 256×256, and then a 224×224 patch is center cropped out from it. SGD with momentum of 0.9 and weight decay of 0.0001 are adopted to train Resnet18 and Resnet50 on ImageNet. The initial learning rate is 0.1 and devided by 10 at 30, 60, 80 and 90 epoch. The models are trained for 100 epoches with a mini-batch size as 256.

4.2. Ablation Study

4.2.1 Experiments on the formulation of AiLoss

As discussed in Section 3.1.2, the numerously available choices to formulate AiLoss can be summarized as two types: (i) which similarity metric Ailoss uses and (ii) which layer it applies to.

For the similarity metric, any arbitrary differentiable distance can be used, and we experiment the mostly common three metrics: KL divergence, Euclidean distance, and the combination of the former two.

The layers in the network can be classified into three categories based on the output type, spatial feature map layer(FM-layer), feature layer(F-layer) and classification layer(C-layer), which outputs feature map, feature and logits respectively. We select a layer from each category as a comparison. Specifically, this experiment is carried out on CIFAR-100 using Resnet32, and the last three layers of this network are utilized, whose outputs are 8×8 feature map, 64-dim feature and 100-dim logits, respectively. For a fair

Dataset	Network	Baseline	AIT
CIFAR-10	Resnet32	92.56	**93.14**
	MobileNetV2	93.76	**94.57**
	WRN-28-10	95.83	**96.03**
CIFAR-100	Resnet32	69.83	**71.48**
	MobileNetV2	74.51	**76.30**
	WRN-28-10	80.50	**81.65**
ImageNet	Resnet18	70.29/89.56	**70.65/89.93**
	Resnet50	76.02/92.96	**76.53/93.12**

Table 1. Top-1 accuracy (%) on CIFAR-10, CIFAR-100 and top-1/top-5 accuracy (%) on ImageNet.

comparison, the loss weight is set as $\lambda = 1$. And the results are presented in Figure 3.

Two observations can be obtained from the results:

(i) It is not advisable to apply AiLoss before pooling. FM-layer performs the worst and even can not converge in most cases, while F-layer achieves better results than baseline but a little worse than C-layer. Noting that there is only one average pooling layer between FM-layer and F-layer, this indicates that AiLoss only takes effect after pooling layer. This is because pooling combines features from each spatial location, and eliminates position mismatch between feature maps.

(ii) KL divergence performs better for formulating AiLoss in C-layer, while the combination of KL divergence and Euclidean distance is preferable in F-layer. The main difference between KL divergence and Euclidean distance is that the former measures the similarity of two probability distributions while the latter estimates the distance of two vectors. In other words, there is statistical meaning in KL divergence, whereas Euclidean distance pays more attention to the value itself. In neural networks, logits are manipulated for computing probabilities, and features are widely used for comparing distance in numerous instance recognition tasks such as face recognition [22] and person re-identification [17]. Thus, the proposition at the beginning of this paragraph is scientifically reasonable.

Above all, we formulate AiLoss with KL divergence and apply it in the classification layer.

4.2.2 Experiments on loss weight λ

Since the hyper parameter λ dominates intra-augmentation variation, it is essential to the final performance. Comparative experiments using Resnet32 on CIFAR-10 and CIFAR-100 are conducted to investigate the sensitiveness of this parameter.

We vary λ from 0 to 4, where $\lambda = 0$ is the baseline without AiLoss. The results are presented in Figure 4. It shows that AiLoss can significantly improve the accuracy with a fitting λ. For both CIFAR-10 and CIFAR-100, the best setting is $\lambda = 1$. We also observe that smaller λ is better for

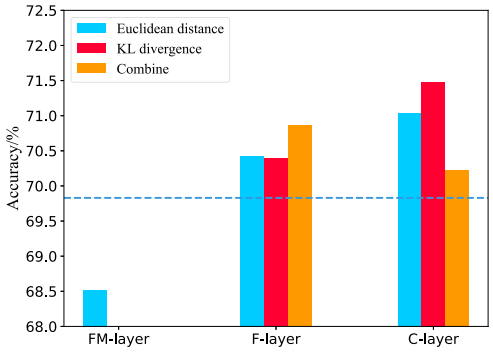
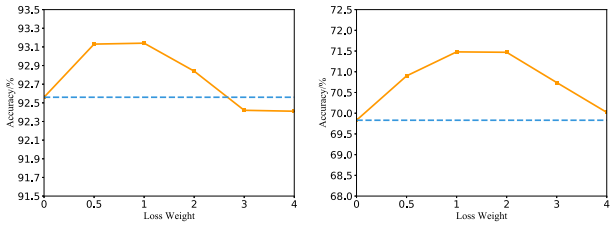


Figure 3. Top-1 accuracy (%) on CIFAR-100 using Resnet32 with different similarity metrics and layer types. The blue dotted line illustrates the baseline accuracy trained w/o AiLoss.

CIFAR-10 while a larger one is properer on CIFAR-100. This results from the substantially more classes in CIFAR-100, which leads to the larger value of Softmax loss. Therefore, a larger loss weight for AiLoss is necessary to balance them. Based on these results, we set $\lambda = 1$ in the following experiments.



(a) CIFAR-10 (b) CIFAR-100

Figure 4. Comparison of results trained with different loss weights λ for AIT on CIFAR-10 and CIFAR-100 (Top-1 accuracy (%)).

4.3. Classification Experiments

4.3.1 CIFAR-10 & CIFAR-100

We apply various network architectures to compare our approach with baseline. From the results of CIFAR-10 and CIFAR-100 shown in Table 1, two major observations are

Network Types		Baseline	AIT	DML		multi-AIT	
Net1	Net2	Net1	Net1	Net1	Net2	Net1	Net2
Resnet32	Resnet32	69.83	71.48	70.64	71.07	**72.32**	72.13
MobileNetV2	MobileNetV2	74.51	76.30	76.21	76.10	**76.78**	76.70
WRN-28-10	WRN-28-10	80.50	81.65	82.31	81.95	82.69	**82.80**
Resnet32	MobileNetV2	69.83	71.48	71.50	76.59	**73.15**	75.87
Resnet32	WRN-28-10	69.83	71.48	71.06	78.88	**73.23**	79.24
MobileNetV2	WRN-28-10	74.51	76.30	77.19	81.92	**78.14**	81.14

Table 2. Comparison between multi-AIT and DML on CIFAR-100.

as follows.

First, AIT is effective in boosting the performance of networks. The top-1 accuracy on CIFAR-10 is raised by 0.56% on average, and the improvement on CIFAR-100 is more significant, with an average of up to 1.51%. This demonstrates that AiLoss can enhance the generalization ability of networks by learning augmentation invariant.

Second, AIT is adaptive to different network architectures. We adopt Resnet, MobileNet and Wide-Resnet (WRN) to evaluate the sensitivity to structures. The results turn out that our algorithm can improve the performance of all the three kinds of networks, especially for those with smaller capacity (Resnet32 and MobileNetV2).

4.3.2 ImageNet

The results of ImageNet are summarized in the last two rows of Table 1. Our algorithm shows certain advantages on this large-scale dataset, although the performance gain is relatively not that significant, which is +0.36/0.37% for Resnet18 and +0.51/0.16% for Resnet50. We believe there are two major causes: (i) AIT works partly by reducing overfitting. As a regularization term in Equation 3, AiLoss is label-free and penalizes the estimates. Therefore, networks trained with AiLoss are less likely to memorize the labels and thus obtain better generalization ability [27]. (ii) However, for large-scale datasets like ImageNet, overfitting is not the major issue, so the effect of AiLoss is slightly degraded.

We further conduct the following two experiments to verify the above thoughts.

Firstly, following [27], we evaluate the memorization of labels by a label corruption experiment. Three CIFAR-100 training sets are generated according to an open-source implementation [26], where 20%, 50% and 80% of the labels are substituted with random noise separately. The labels of the test set are kept intact for fair evaluation. As presented in Table 3, AIT outperform the baseline in all settings. The results illustrate that AiLoss can reduce the memorization of labels and prevent the model from overfitting the corrupted labels.

Label Corruption	Baseline	AIT
20%	62.20	64.68
50%	52.10	55.11
80%	26.73	28.08

Table 3. Results on CIFAR-100 using Resnet32 with corrupted labels.

Data	Baseline	AIT	Improvement
10%	46.57/71.07	48.01/72.24	+1.44/1.17
30%	61.72/83.43	62.73/84.45	+1.01/1.02
50%	66.26/86.82	66.78/87.36	+0.52/0.54

Table 4. Results of Resnet18 trained with part data of ImageNet.

Secondly, we conduct a comparative experiment of different training set sizes to study the correlation between AIT and overfitting. 10%, 30% and 50% of the original ImageNet training set are randomly sampled to generate three training datasets respectively, and the test set remains unchanged. We summarize the results in Table 4. With the training data increasing, the overfitting phenomenon is gradually reduced, and the performance improvement of AIT is also decreasing. Therefore, this confirms our previous conjecture that AIT performs better on tasks suffering from overfitting.

4.3.3 Extension to more networks

Following DML, we train two networks together to evaluate the performance of multi-AIT. Noting that we set $\lambda = 3$ for multi-AIT. From the results in Table 2, we can draw two observations:

First, multi-AIT achieves better performance than both DML and AIT in classification. As shown in the first three rows, multi-AIT surpasses DML on both networks and also outperforms AIT. Multi-AIT can help networks learn inter-network augmentation invariant, and thus promotes the generalization ability to a greater extent. However, comparing with AIT, multi-AIT consumes more computing resource due to the multi-network structure, while AIT achieves a

Aug	original		random crop		grayscale		random color	
	Baseline	AIT	Baseline	AIT	Baseline	AIT	Baseline	AIT
rh	63.77	67.42	51.50	55.09	42.51	45.17	59.40	63.06
rp	69.83	71.48	68.33	70.67	44.94	47.39	65.07	66.79
rg	65.25	66.31	54.55	54.89	54.96	59.58	62.10	64.16
rc	64.15	65.81	52.51	55.25	46.39	53.75	63.47	65.58
rpg	69.91	71.44	69.20	70.04	58.66	63.42	66.27	68.68
rpc	69.26	69.35	68.27	68.10	48.71	53.08	68.56	69.16
rgc	64.38	63.91	51.07	52.16	56.36	61.05	64.04	63.56
rpgc	69.61	68.87	69.03	67.82	60.27	64.32	68.11	68.68
rp,rg	-	71.39	-	70.03	-	61.93	-	67.97
rp,rc	-	70.58	-	69.40	-	54.65	-	70.03
rg,rc	-	66.57	-	53.75	-	60.66	-	66.16

Table 5. Experiments on the augmentation transfer ability of networks. The networks are evaluated on four test sets, including original, random crop, grayscale and random color. Here Aug means augmentation method.

satisfying tradeoff between computing overhead and performance.

Second, multi-AIT performs better than DML in the teacher-student training manner[5]. We train two networks with different capacity together, with the larger one as teacher and smaller as student. Different from Distillation[5], both teacher and student network are trainable and trained from scratch. In DML, two networks learn collaboratively and teach each other throughout the training process. Multi-AIT further introduces inter-network augmentation invariant, promoting the student networks to obtain better generalization ability. Results in the last three rows indicate that student networks trained by multi-AIT outperform those trained by DML.

4.4. Augmentation Transfer

In order to verify whether our algorithm improves the augmentation transfer ability of networks, we conduct extensive experiments on CIFAR100 using Resnet32.

Four categories of random augmentations are utilized in the training process:

(i) Random horizontal flip (rh): Randomly flipping the sample horizontally with a probability of 0.5.

(ii) Random crop (rp): Applying zero padding 4 pixels on each side and then randomly cropping out a 32×32 patch from the padded sample.

(iii) Random grayscale (rg): Randomly convert the sample to grayscale with a probability of 0.1.

(iiii) Random color jittering (rc): Randomly change the brightness, contrast and saturation of the sample with a max bias of 0.1.

The combination of different augmentation methods is denoted as rpg (random crop and random grayscale), rpc (random crop and random color), etc. We also try applying different augmentations for each variant. For example, 'rp,rg' means applying random crop to one variant and random grayscale to the other. Moreover, it is worth noting that as a basic augmentation method, random horizontal flip is applied in all the experiments.

In the test phase, we generate three augmented test sets from the original one using random crop, grayscale, and random color jittering respectively. The test sets are kept intact once generated.

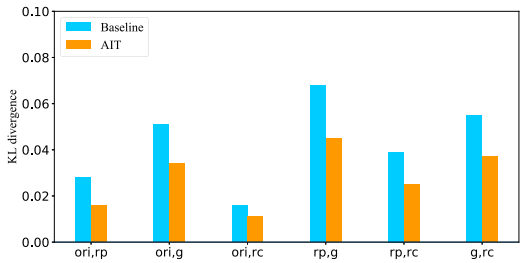
The three observations obtained from the results in Table 5 are as follows:

First, AiLoss can improve the augmentation transfer ability of networks. The results in the first four rows show that AIT achieves better performance than baseline on both original test set and augmented test sets. Networks trained with AiLoss can efficiently transfer to a test set with augmentations not used during training, and accomplish a comparable accuracy to the network trained with that augmentation.

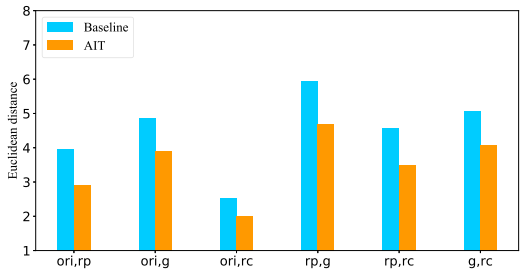
Second, a proper combination of augmentations is important and piling up augmentations is a rather efficient way to improve augmentation transfer ability. As shown in the middle four rows, networks perform better on augmented test sets with the number of augmentations increasing, while the results on the original test set decrease as a trade-off between augmentation transfer ability and accuracy. The best performance on original test set is achieved by 'rpg' rather than 'rpgc', emphasizing the importance of proper augmentation combination. Moreover, AIT achieves more balanced results among the four test sets, indicating a better augmentation transfer ability.

Last, using different augmentations for each variant is also an efficient way to obtain both accuracy and augmentation transfer ability. Compared with the first three rows in the middle, the last three achieve better results in most instances, especially when random color jittering is applied during training.

4.5. Intra-augmentation Variation



(a) KL divergence of predictions



(b) Euclidean distance of features

Figure 5. KL divergence of predictions and Euclidean distance of features between different test sets. 'ori,rp' means comparison between original test set and random crop test set, and other parameters on the X axis is analogous.

We compare the KL divergence of predictions and Euclidean distance of features between different test sets to study the intra-augmentation variation. Two Resnet32 models are trained on CIFAR-100 with only random horizontal flip and random crop augmentations. Conventional training strategy and AIT are applied to train them respectively. And the four test sets for comparison are the ones mentioned in Section 4.4.

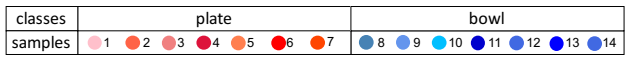
As shown in Figure 5, variation of both features and predictions between test sets is significantly reduced in AIT. Although the Euclidean distance of features is not specifically optimized during training, AiLoss can help the network learn features that are invariant to augmentations.

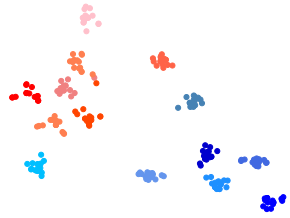
In order to further study the effect of AiLoss on features, we visualize the feature distributions of baseline model and AIT model respectively. We select 14 samples from the test set of CIFAR-100, and for each sample 16 variants are generated by applying random crop and random color jittering augmentations to them. From the results in Figure 6, we can draw two observations:

First, AiLoss can reduce intra-augmentation variation significantly. Under the supervision of AiLoss, features of variants generated from the identical sample are clustered together and there are apparent gaps between different samples. The feature distribution of baseline shows that Softmax Loss can also reduce intra-augmentation variation to a certain extent, while AiLoss further enhances the effect and obtains better feature distributions.

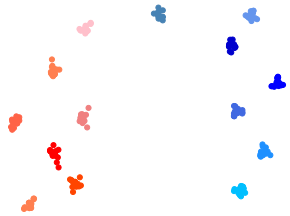
Second, AiLoss can optimize the decision boundary between classes. Features of AIT are more discriminative than those of baseline, and the margin between different classes is also larger.

Above all, AiLoss shows powerful capability to minimize intra-augmentation variation and thus can help networks learn features that are invariant to augmentations.





(a) Baseline



(b) AIT

Figure 6. tSNE Visualization of feature distributions from two Resnet32 models trained w/ and w/o AiLoss on CIFAR-100. The circles with the same color represent the features of variants generated from the identical sample.

5. Conclusions

In this work, we propose to improve the augmentation transfer ability of neural networks by learning augmentation invariant. We firstly design a novel AiLoss to minimize intra-augmentation variation. Based on AiLoss, a simple yet efficient training strategy AIT is proposed to assist networks in learning augmentation invariant. Extensive experiments on several classification benchmarks have convincingly demonstrated the effectiveness of our approach to improve the augmentation transfer ability and generalization ability of networks. Further extending AIT to multiple networks, we propose multi-AIT to learn inter-network augmentation invariant. Experiments indicate that multi-AIT can further improve performance and is promising to train small student networks.

References

[1] Binghui Chen, Weihong Deng, and Junping Du. Noisy softmax: Improving the generalization ability of dcnn via postponing the early softmax saturation. 2017.

[2] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. 2018.

[3] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. pages 764–773, 2017.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016.

[5] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *Computer Science*, 14(7):38–39, 2015.

[6] Sergey Ioffe and Christian Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. pages 448–456, 2015.

[7] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. pages 2017–2025, 2015.

[8] Jason Jo and Yoshua Bengio. Measuring the tendency of cnns to learn surface statistical regularities. 2017.

[9] Aditya Khosla, Tinghui Zhou, Tomasz Malisiewicz, Alexei A. Efros, and Antonio Torralba. Undoing the damage of dataset bias. 2012.

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

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *International Conference on Neural Information Processing Systems*, pages 1097–1105, 2012.

[12] Anders Krogh and John A. Hertz. A simple weight decay can improve generalization. In *International Conference on Neural Information Processing Systems*, 1991.

[13] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. *eprint arXiv:1409.5185*, pages 562–570, 2014.

[14] Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. Revisiting batch normalization for practical domain adaptation. *Pattern Recognition*, 80, 2016.

[15] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*, pages 21–37, 2016.

[16] Xingang Pan, Ping Luo, Jianping Shi, and Xiaoou Tang. Two at once: Enhancing learning and generalization capacities via ibn-net. 2018.

[17] Rahul Rama Varior, Mrinal Haloi, and Gang Wang. *Gated Siamese Convolutional Neural Network Architecture for Human Re-Identification*. Springer International Publishing, 2016.

[18] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[19] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. 2018.

[20] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[21] Baochen Sun and Kate Saenko. *Deep CORAL: Correlation Alignment for Deep Domain Adaptation*. 2016.

[22] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation from predicting 10,000 classes. In *Computer Vision and Pattern Recognition*, pages 1891–1898, 2014.

[23] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. 2016.

[24] Yuichi Yoshida and Takeru Miyato. Spectral norm regularization for improving the generalizability of deep learning. 2017.

[25] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *CoRR*, abs/1605.07146, 2016.

[26] Zhang. Url, 2017.

[27] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. 2018.

[28] Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. Deep mutual learning. 2017.