STDA-inf: Style Transfer for Data Augmentation Through In-data Training and Fusion Inference

Tao Hong, Yajun Zou, and Jinwen Ma(✉)

Department of Information and Computational Sciences, School of Mathematical Sciences and LMAM, Peking University, Beijing 100871, China
{paul.ht, zouyj}@pku.edu.cn, jwma@math.pku.edu.cn

Abstract. Style transfer has been effectively applied to data augmentation. However, previous work requires careful selection of style images out of the concerned datasets, and neglects the impact of style transfer on the inference procedure. In this paper, we propose a novel method **STDA-inf** for image classification: **S**tyle **T**ransfer for **D**ata **A**ugmentation through **in**-data training and **f**usion inference. Firstly, we acquire the transferred training data in an adaptive way of in-data, in which style images are extracted from the training data itself. An online end-to-end training strategy is utilized to create an adversarial training effect, thereby alleviating the overfitting on textures when identifying different classes. Moreover, we fuse the outputs of the original and transferred images from the trained network, obtaining a more accurate classification. It is demonstrated by the experiments that our proposed method outperforms the previous style augmentation method with 7% improvement of classification accuracy on STL-10 and 3% on Caltech-256 dataset, respectively. Its superiority is also demonstrated over the other data augmentation methods.

Keywords: Style transfer · Data augmentation · In-data style · Fusion inference

1 Introduction

In recent years, deep neural networks have performed superiorly in many computer vision tasks such as classification, object detection, and so on. Driven by deep learning research, more effective data are needed under the challenges of lack of data, expensive tags, imbalanced categories, *etc*. Data augmentation is a powerful tool to solve this problem. In general, generating new samples via label-preserving transformations [1] can expand the training dataset, resulting in a better performance on the relative models.

On the other hand, the data learning mechanism of networks is also worthy of exploration. In terms of human perception, it is naturally believed that we classify objects majorly by shapes. Nevertheless in [2], Geirhos *et al.* found out that the ImageNet trained Convolutional Neural Networks (CNNs) are strongly biased towards recognizing textures rather than shapes. Since texture is considered to be closely related to image style, this discovery leads researchers to utilize the style to implement

D.-S. Huang et al. (Eds.): ICIC 2021, LNCS 12837, pp. 76–90, 2021.
https://doi.org/10.1007/978-3-030-84529-2_7

classification strategies. Style transfer [3, 4] appeals to be a promising efficient tool to achieve advanced data augmentation, by utilizing the effect of textures to divert more attention of networks to shapes.

Jackson *et al.* adopted an arbitrary style transfer network to perform style randomization [5]. Yet the style images are specially chosen from the Office dataset [6], which lacks a relationship with the concerned dataset of the given task. We refer to these style images as *out-of-data*. In [7], it reveals that styles adding too many colors and shapes on original images lead to bad performance, which reflects the difficulty of locating proper out-of-data styles. And in [7], the *offline* style transfer policy is adopted, such that the transferred dataset is given in advance and stored locally. Certainly, this non-end-to-end policy desires extra space storage, especially when trying many styles one time. Besides, all the previous work only concentrates on the style augmentation during training, neglecting to explore its effect on inference.

In this paper, we apply the style transfer based data augmentation on classification tasks with improvement and innovation (named **STDA-inf**). We extract the style images from the given dataset in the way of so-called *in-data*, which is more adaptive and controllable. In-data style augmentation harvests more robust models about style features at a higher level, which is orthogonal with other augmentation methods. Note that our proposed online training is end-to-end. Moreover, we adopt the style augmentation into the inference course in a way of *fusion inference*. The original image and the transferred images are fed into the trained network, and the final classification result is obtained via the weighted sum of their classification results. Our experiments strongly verify the effectiveness of STDA-inf. The main contributions of our work are as follows:

- We improve the selection source of styles from *out-of-data* to *in-data*, and get the *state-of-the-art* performance of style augmentation. The latter fully utilizes texture information of the datasets themselves, to create an adversarial training effect, thereby avoiding further overfitting, especially overfitting textures. And random selection can already reach good enough behavior.
- We adopt the *fusion inference* which gets a remarkable improvement of accuracy in comparison with the universe inference. This operation makes full use of style transferred information, because not only the original distribution is learned during training, but also the style transferred distribution.
- We present a comprehensive exploration on style augmentation in detail, such as style selection strategy, transfer policy, *etc.* And we present the superiority over interpolation-based data augmentation such as Mixup [8].

The rest of the paper is organized as follows. In Sect. 2, we introduce and review the related work. Our proposed method is presented in Sect. 3. In Sect. 4, experiments and analyses are given to illustrate the efficiency and effectiveness of our method. Finally, a brief conclusion is made in Sect. 5.

2 Related Work

2.1 Style Transfer

Style transfer means adopting a new style of an image into another image. Style is thought related to the variance or eigenvalue or gradient of the pixel tensor. The first attempted work adopts Gram matrix to encode the deep features of style representation [3, 4]. In [9], Huang *et al.* proposed an adaptive instance normalization (AdaIN) layer, achieving faster speed. Different from artistic effects, Luan *et al.* introduced a photo-realistic loss term to optimize towards photorealistic visual effects [10]. To alleviate the time consumption, PhotoWCT [11, 12] adopts a non-end-to-end architecture to insert whitening and coloring transform (WCT) modules in auto-encoders. Further, Yoo *et al.* [13] proposed Wavelet Corrected Transfer (WCT^2) aiming at eliminating post-processing steps while preserving fine details. And neural architecture search is adopted in StyleNAS [14].

In a word, the current mainstream framework adopts CNNs to encode content images (denoted as c) and style images (denoted as s) into feature maps. After encoding, we try to transfer the feature maps with style relative modules and decode them into style transferred images (denoted as cs). We adopt the AdaIN [9] model as the style transfer module in this paper. AdaIN simply aligns the channel-wise mean and variance of the content to match the style's, without learnable affine parameters. The input is simply shifted with mean μ and scaled with variance σ:

$$\mathrm{AdaIN}(c, s) = \sigma(s)\left(\frac{c - \mu(c)}{\sigma(c)}\right) + \mu(s) \tag{1}$$

After getting cs, we adopt weighted interpolation between cs and s to control the degree of style transfer and content reservation. Note that we all set the weight of cs as 1.0.

2.2 Data Augmentation

Data augmentation has been a standard part of training deep neural networks ever since the work of Krizhevsky *et al.* [15]. Data augmentation is plausible to avoid overfitting so that the trained models will acquire stronger generalization capacity. Certain traditional augmentation techniques are very popular and effective, including horizontal flipping, random rotation, random cropping. They can make the model get more robust training in position, angle, *etc.* [15].

On the other hand, the convex combinations of pairs of inputs and their labels (Mixup) [8] also generate new samples, which inspires a series of augmentation methods such as MixMatch [16]. Cutout [17] refers to randomly masking out square regions of inputs, while CutMix [18] means that patches are cut and pasted among training images. Further, Manifold Mixup [19] interpolates between feature maps rather than just inputs to get better representations of hidden states.

3 Proposed Approach

In this section, we propose a systematic style augmentation approach STDA-inf, whose overview is shown in Fig. 1.

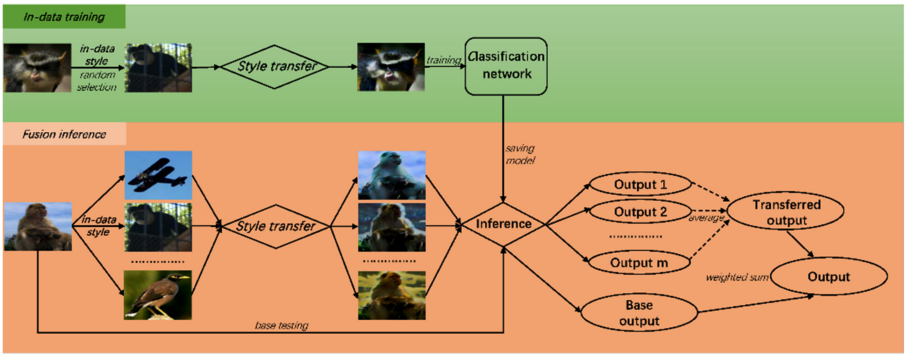

Fig. 1. The overview of our proposed approach STDA-inf: in-data training & fusion inference.

3.1 In-data Training

Before feeding training samples into a classification network, we usually apply some transformations Φ on them (with a certain probability p) such as horizontal flipping, which is just the traditional augmentation. Style augmentation also works in this stage, then the network is optimized towards

$$\min \mathcal{L}(f(\Phi(x)), y) \tag{2}$$

where $\mathcal{L}, f, x$ and y denote the loss function, network, sample and label.

During training, content images are the whole training samples generally. And style images in the way of in-data are from a subset of the training samples. Denoting the training and test set as Tr and Te, and the set of content images and style images as C and S respectively. Then for every paired (c, s), we have.

$$\text{Train: } c \in C = Tr, s \in S \subset Tr \tag{3}$$

$$\text{Inference: } c \in C = Te, s \in S \subset Tr \tag{4}$$

where the explanation of inference will be given in Sect. 3.2.

It's worth noting that Φ_{in} (in-data style transfer) could be utilized to force the network to train towards an adversarial direction. This is superior to Φ_{out} (out-of-data style transfer) adopted in the previous work [5, 7]. As for the chosen style number $|S|$, we can adjust it according to the whole training number $|C|$. For example, choosing 10 images per class from STL-10 (total 10 class) or choosing 100 images from Caltech-256 is a good choice.

In addition to random choice, we have explored different choice strategies called *min-loss* and *max-loss* choice. Feeding the original training samples into the trained models, we can sort them according to the classification loss $\mathcal{L}(f(x), y)$. Min-loss means choosing top images with the smallest loss as styles, while max-loss is corresponding to top images with the largest loss.

Moreover, we can divide the transfer procedure $\Phi(x)$ into two modes, called *offline* and *online* respectively. Offline means transferring samples in advance of training and storing locally, *i.e.* all raw inputs are original samples or transferred samples. While online means operating style transfer in the pre-processing stage, *i.e.* all raw inputs are original samples. Offline is space-consuming while online is time-consuming. It's acknowledged that to some extent, the more patterns we provide, the better result data augmentation will get. Offline will consume too much space if we want to get many patterns. Therefore, we mainly adopt end-to-end online operation mode in our work. As introduced in Sect. 2.1, AdaIN module is very fast, relatively speaking. This is one reason why we prefer to combine online mode with AdaIN rather than other algorithms such as StyleNAS. Under online mode, we set the style transfer proportion p as 0.3, which means 30% of training samples get transferred.

Considering the category information, we can divide the transfer policy into two modes: *inter-class transfer* and *intra-class transfer*. During training, if we transfer the images of one class with the style images of the corresponding class, we call this operation *inter-class training*. In other words, every paired (c, s) comes from the same class. If we transfer regardless of the class match between contents and styles, we call the operation *intra-class training*. Intra-class training defeats inter-class training at model performance, since the former retards the overfitting while the latter exacerbates the overfitting on the textures. We adopt intra-class training mode unless otherwise specified.

3.2 Fusion Inference

After training, the universal approach is testing original test samples ($x \in Te$, called *base test*) via the trained model directly. Supposing there are n classes, then the last layer of a network is an n-dimension vector v_{base}. The index of the largest value in the vector represents the divided category.

Apart from base test samples Te, we transfer them with style images (chosen from $S \subset Tr$, the same as training) to get transferred test samples $\Phi(Te)$. After every base sample is transferred once, we call it a round. So, we can get one complete test data in every round. Something different from the training case, *inter-class test* means style images are all from one class in one round, while *intra-class test* means transferring without the limit of class in each round. We adopt intra-class test mode unless otherwise specified. For a specific sample $\Phi(x)$, it has the same content but different styles in different rounds. Denoting the vector of the last layer in round i as v_i, then we get the average vector

$$v_{\text{avg}} = \frac{1}{m} \sum\nolimits_{i=1}^{m} v_i \tag{5}$$

where m denotes the total rounds. Furthermore, we can interpolate between v_{base} and v_{avg} with the weight coefficient $\beta \in [0, 1]$, then the final vector is

$$v_{\text{final}} = \beta \cdot v_{\text{base}} + (1 - \beta) \cdot v_{\text{avg}} \tag{6}$$

Classifying according to v_{final} rather than v_{base} will get a considerable promotion in accuracy since training also happens on style transferred distribution rather than just on original distribution. Although classifying just according to a certain round's vector v_i gets lower accuracy than v_{base}, every round's vector contains its judgment for classes. (We infer that since $|S|$ is far smaller than $|C|$, the classifier tries to grasp style patterns naturally. After all, it is much easier to remember a few patterns than many.) Therefore, the test accuracy increases along with the fusion of vectors within a certain range. We call this inference method *fusion inference*.

Examining the two hyperparameters, m and β, in the above algorithm, how shall we determine them? Within a certain range, increasing of m brings cumulative improvement on accuracy. A reasonable m should bring great improvement while not consume too much time, which can be fine-tuned in different datasets. As for β, we adopt a simple grid search strategy to search for the proximate optimal β_{opt}, with steps of 0.1 and 0.01. β improves the accuracy over a wide range of right intervals in $[0, 1]$, which reflects the domination of v_{base}.

4 Experiments

4.1 Dataset

We evaluate our proposed STDA-inf on several datasets for classification task, *i.e.* STL-10 and Caltech-256, inherited from our mainly compared work [5, 7].

- STL-10[1] has 10 classes such as airplane, bird, monkey, with 500 training images and 800 test images per class. Besides, STL-10 has 100000 unlabeled images for unsupervised learning. Every image is 96×96 pixels. Because the number of test images exceeds training images, data augmentation plays a big role.
- Caltech-256[2] consists of 257 classes and the number of images per class is not equal. We randomly choose 60 images per class as the training dataset, the remaining images as the test dataset. Every image is about 200–300 pixels, so it is resized to 224×224 firstly.

In the out-of-data way, the style images are mainly chosen from the Office dataset [6] in the same way as [5], with 7 inherited from [7]. These oil paintings are so colorful that their style transfer effect is more intense than general style images of in-data way. On the other hand, AdaIN is more intense than StyleNAS. Taking an image from STL-10 as an example, we can observe the different degrees of style transfer in Fig. 2. *AdaIN module*

[1] https://cs.stanford.edu/~acoates/stl10/.

[2] http://www.vision.caltech.edu/Image_Datasets/Caltech256/.

cooperated with style images of in-data way is plausible for style augmentation, neither too intense like out-of-data way nor too soft and slow like StyleNAS.

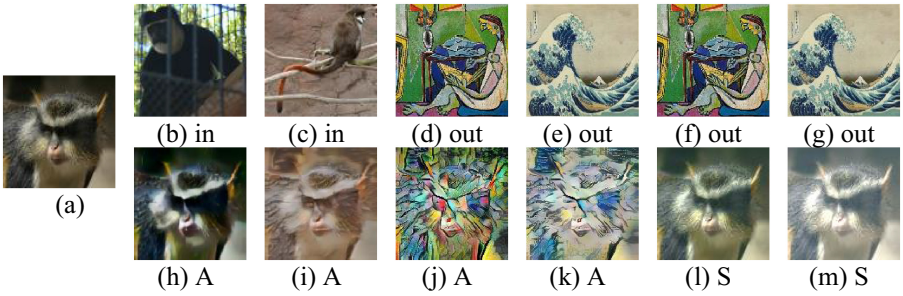

Fig. 2. Different effects of style transfer. (a) is an original image of the monkey class in STL-10. The 1st row is the style image and the 2nd row is the corresponding transferred image. *In* and *out* respectively represent in-data way and out-of-data way. *A* and *S* represent AdaIN and StyleNAS, respectively.

4.2 Implementation Details

We finish all the experiments on 2 NVIDIA Tesla P100 GPUs by PyTorch framework. The classification network is the widely applied ResNet50 [20]. Different datasets just need a little change in the stride and kernel size of convolution. Our adopted style transfer model AdaIN is released from Github[3]. AdaIN model adopts the pre-trained VGG19 as the encoder, and then only needs to train the decoder. Apart from style augmentation, we also exploit two traditional augmentation methods (*i.e.* horizontal flipping and random cropping, abbreviated as *tra*) and Mixup, *etc*.

- Training STL-10: total epoch is 150, training batch is 256, optimizer is Adam with momentum $\beta_1 = 0.9$, $\beta_2 = 0.999$, initial learning rate is 0.001 and it decays by 0.1 at the epoch of 80, 120, weight decay is 5×10^{-4}.
- Training Caltech-256: total epoch is 150, training batch is 32, optimizer is SGD with default parameters, initial learning rate is 0.01 and it decays by 0.2 at the epoch of 60, 120, weight decay is 5×10^{-4}.

4.3 Experiment Results

We focus on STL-10 dataset firstly and systematically illustrate the experiment results on it. Then there is much similarity on Caltech-256.

STL-10. We first present comprehensive results which verify the strength of our STDA-inf over out-data style augmentation and other augmentation methods like Mixup. Next, the presentation will follow the order as: style source, style number, hyperparameter optimization, robustness exploration and time analysis.

[3] https://github.com/xunhuang1995/AdaIN-style.

Table 1. Test accuracy (%) of our STDA-inf on STL-10 with and without traditional augmentation. Out style with number 10 and in style with number 10×10. † are the reproduce results of compared work.

+Tra	Style type	Base test	Rounds m			
			1	3	10	15
w/o 60.17	out random	67.18†	71.08	71.63	72.31	72.54
	in random	68.95	71.73	71.89	72.83	72.88
	in min-loss	68.48	71.05	72.09	72.85	72.94
	in max-loss	**70.33**	71.90	72.79	73.89	**74.05**
w/ 78.13	out random	79.56†	81.76	81.84	81.88	81.89
	in random	**82.26**	83.43	83.70	**83.78**	83.69
	in min-loss	81.42	81.75	81.95	82.03	82.14
	in max-loss	81.54	82.04	82.31	82.41	82.59

As Table 1 shows, style augmentation of in-data way behaves much better than out-of-data way. Out-of-data is not as stable as in-data, getting bigger variance in the repeated experiments. From out-of-data to in-data, the base test accuracy gets around 3% improvement. As for fusion inference, 1 round improves the accuracy a lot and enlarging to 15 rounds still harvests some improvement. Without or with traditional augmentation, the test accuracy gets beyond 4% or 1% improvement further, after 15-rounds fusion inference. As we can see, our method STDA-inf could at most improve 14% without traditional augmentation and 5% with traditional augmentation compared to the baseline.

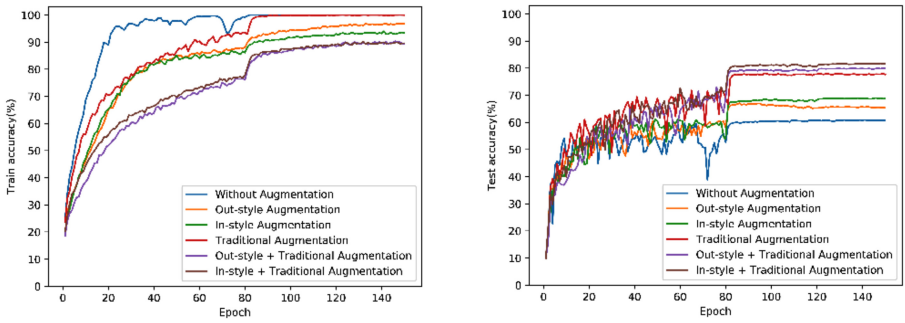

Fig. 3. The comparison of training and test accuracy on STL-10 under different augmentation methods.

Moreover, we compare the training accuracy and test accuracy in Fig. 3. The jump points exactly correspond to the changing points of learning rate. Note that the corresponding configuration is: *in random* with 100 styles; *out random* with 10 styles. 100 and 10 are chosen according to the search of proper number (shown in Table 3). In-data way would manifest greater superiority if we both choose the same style number. As we can see, training without augmentation or only with traditional augmentation causes severe overfitting. While training with style augmentation is harder to converge to a

very high accuracy due to the attack of styles' uncertainty. And training with both traditional and style augmentation brings a better cumulative effect. Also, in-data way behaves much better than out-of-data, which verifies our first contribution powerfully.

Table 2. Test accuracy (%) of composite augmentation methods on STL-10. Column 2&4 shows the superiority of our style augmentation over interpolation-based augmentation methods. *Style* corresponds to our *in random* way in Table 1.

	Baseline	+style	+tra	+tra+style
Baseline	60.17	–	78.13	–
+Mixup [19]	63.74	68.98	81.14	83.34
+Cutout [3]	61.10	**71.45**	80.13	82.06
+CutMix [18]	63.39	71.21	80.90	**83.63**
+Manifold [15]	60.10	68.35	76.45	77.21
+style (ours)	**68.95**	–	**82.26**	–

In addition, we compare style augmentation with Mixup, Cutout, CutMix and Manifold Mixup. The specific parameter configuration is explained in the Appendix. MixUp *etc.* are simply linear interpolations while style augmentation utilizes semantic information (nonlinear). As Table 2 shows, our style augmentation performs better (see Column 2&4), and it can be used in conjunction with existing forms of data augmentation to further improve model performance (see Column 3&5). It needs to be emphasized that these augmentation methods don't work during inference, yet style augmentation performs better along with fusion inference. And the Test Time Augmentation (TTA) is carried out on the basis of traditional augmentation, which is trivial and doesn't get significant promotion, compared to our fusion inference.

Style Source. Except for selecting styles randomly, we investigate the cross-entropy loss of every training sample and sort them. The minimal loss is in the magnitude of 10^{-6} while the maximal loss is in the magnitude of 10^{-3}. It seems that the max-loss images are harder to classify while the min-loss images are easier, somewhere related to the semantic information of their styles.

Style Number. As shown in Table 3, we investigate the proper number of style images. In this experiment, out-of-data style images are randomly chosen, while in-data style images are chosen on average from each class (for example, 100 means 10 images per class). We speculate that the variation trend is: as the number of style images $|S|$ increases, the test accuracy increases first and then decreases. And the optimal $|S|$ of in-data way seems bigger than out-of-data way. A reasonable explanation is that the intensity of style augmentation should be in an appropriate range. To put it in another way, style images of out-of-date way are so colorful that too many intense styles make training models hard to catch dominant patterns and converge. In turn, too few soft styles can't give full play to the role of data augmentation. It's worth noting that in *out random*, 7 images are inherited from [7]. [7] choose 8 different styles that look different from each other and only 7 styles bring about positive performance. The test accuracy is lower than 67.18% as we substitute the 7 images with other random images.

Table 3. Searching proper number of style images: test accuracy (%) on STL-10 without traditional augmentation. The baseline is 60.17%.

Style type	Style number		
	10	30	100
out random	**67.18**	66.98	66.29
in random	69.23	**69.30**	68.95
in min-loss	67.19	67.48	**68.48**
in max-loss	69.63	**70.75**	70.33

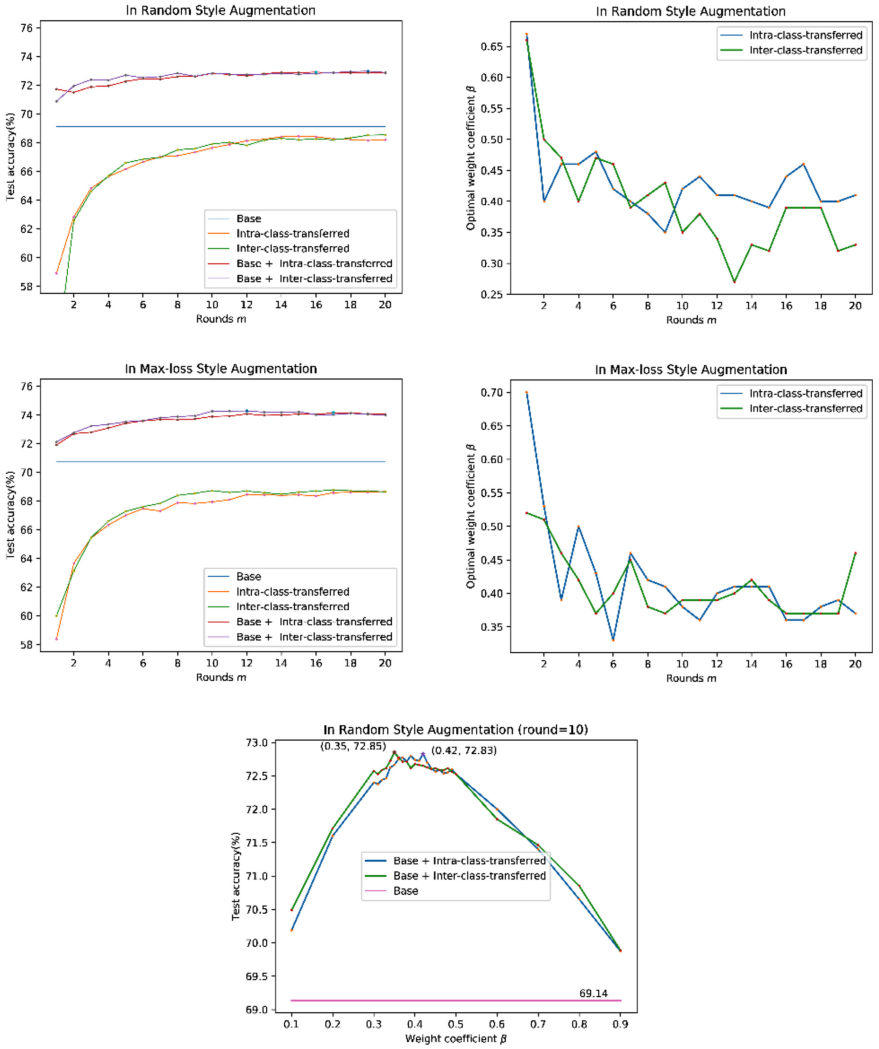

Fig. 4. Hyperparameter search on the fusion inference rounds m and the weight coefficient β. The horizontal line represents the accuracy of base test. Transferred without base means classifying only via the style transferred samples.

Hyperparameter Optimization. Fusion inference is effective no matter which kind of style type. Figure 4 shows a complete hyperparameter search on the fusion inference rounds m and the weight coefficient β on STL-10. Note that the presented figure corresponds to one result of the repeated experiment. Since the base test dominates in fusion inference, the accuracy only with transferred samples can't exceed the base test. Within a certain range, the accuracy increases as rounds m increases until steady. And the accuracy after only 1 round gets remarkable improvement. On STL-10, around 15-rounds harvests optimal test accuracy. Considering the accuracy and the time consumption together, we can fix $m = 5$ or less. Besides *intra-class test*, we compare *inter-class test* with it, and find that the test accuracy gets somewhat higher in the *inter-class* mode, especially in the former several rounds, which is in line with our expectations. We speculate that classification with a smaller number of style images from one class in one round makes the classifier easier to identify the patterns per class, less misled by different styles. With regard to β, it works well over a wide range of $[0, 1]$. And optimal β decreases as m increases, which means transferred test samples play a more and more important role in the classification.

Robustness. On the other hand, we explore the behavior of different trained models on different test samples (style robustness). As Table 4 shows, each row corresponds to a kind of trained model (baseline model or trained through out-data/in-data way) while each column corresponds to a kind of test dataset (base samples or transferred by out-data/in-data styles, denoted as Te, $\Phi_{\text{out}}(Te)$, $\Phi_{\text{in}}(Te)$ respectively). The out styles are the same as the 10 images adopted in out-data random training and the in styles are the same as the 100 images adopted in in-data random training. The 10 mixed styles (generating $\Phi_{\text{mix}}(Te)$) consist of 5 out styles and 5 in styles, and none of them is the same as the styles adopted in training, which is more convincing to prove the generalization of in-data training (see the last Column).

Table 4. Accuracy (%) of different trained models on different test samples of STL-10.

Model\Data	Te	$\Phi_{\text{out}}(Te)$	$\Phi_{\text{in}}(Te)$	$\Phi_{\text{mix}}(Te)$
base	78.13	19.71	23.86	16.84
out	80.79	**62.60**	46.81	36.58
in (ours)	**82.06**	55.05	**61.89**	**46.45**

Speaking of adversarial attacks, Kurakin *et al.* illustrated adversarial examples in the physical world [21] and Ilyas *et al.* explored robust features and non-robust features in detail [22]. Besides, many attack methods are proposed such as the classical Fast Gradient Sign Method (FGSM) [23] and Project Gradient Descent (PGD) [24]. We report the results of FGSM attack in Table 5. Considering that STL-10 is challenging due to the excess of test images over training images, we don't set the value of hyperparameter ϵ to be large. Note that $\epsilon = 0.004, 0.016, 0.030$ correspond to attacking only $1/255$, $4/255$, and $8/255$ magnitude of pixels, respectively. In-data way of style augmentation defeats out-of-data way. Style augmentation performs better when the

attack is not very intense. And Mixup and CutMix perform well. We speculate that on one hand, it's harder to get strong robustness on STL-10 since it has a bigger proportion of test samples. On the other hand, style augmentation is not as intense as augmentation methods like Mixup. But when involving with the disturbance of styles, attacking with other classes' styles will make it harder for the classifier to classify samples during training. Thus we can avoid the overfitting of textures to grasp essential patterns, and thereby obtain stronger (style) robustness.

Table 5. Accuracy (%) after FGSM white-box attack with different intensity on STL-10. Note that except for the baseline, all other rows adopt the traditional augmentation. And the last 3 rows belong to ours.

ϵ	0	0.004	0.016	0.030
Baseline	60.91	25.28	1.90	0.23
Traditional	78.13	57.50	16.34	7.00
Mixup	81.14	57.20	24.43	17.19
Cutout	80.13	56.56	13.71	6.39
CutMix	80.90	54.74	**24.81**	**20.10**
Manifold	76.45	54.15	17.64	10.93
out random	80.79	60.14	15.68	6.15
in random	**83.15**	**62.10**	17.76	9.06
in min-loss	81.53	58.40	16.26	7.68
in max-loss	81.74	59.51	15.14	6.35

Time Analysis. Finally, we illustrate the time consumption briefly, as shown in Table 6. On the basis of training without any data augmentation, traditional augmentation takes about another 0.03 h (hour) while style augmentation takes about another 0.30 h. As for the inference time, one-round fusion inference takes about 120 s (second), compared to 82 s of the base test. As for Mixup *etc.*, they are not very time consuming. The time consumption of style transfer is also relative to the resized size of content and style images. In future work, the function and mechanism of style transfer can be explored further, especially reducing the inference time.

Table 6. Training (2 GPU) and inference (1 GPU) time on STL-10. 120 s is the inference time in one round of fusion inference.

Augmentation	Base	+tra	+style
Training time (h)	1.30	+0.03	+0.30
Inference time (s)	82	–	120

Caltech-256. The main experiments on Caltech-256 dataset are the same as STL-10. However, Caltech-256 contains 257 classes so it's much more challenging. The exploration of the number of style images $|S|$ can be seen in Table 7. The variation trend is much similar to STL-10: the test accuracy increases first and then decreases as $|S|$ increases. We set $|S|$ as 100 and randomly choose style images from the whole training images. There is no need to explore the optimal $|S|$ since 100 is efficient enough. With style augmentation, it's observed that the training accuracy still has improvement space after the last epoch. So we extend the training epoch from 150 to 200 and change the learning rate by multiplying 0.1 at the 190th epoch, getting higher accuracy.

Table 7. Test accuracy (%) on Caltech-256 with traditional augmentation. The baseline is 60.86%. Note that 771 means choosing 3 style images per class for *in random*.

Style category	Style number			
	10	100	250	771
out random	61.48	**63.17**	62.74	63.13
in random	61.63	**64.44**	63.65	63.98

We both set $|S|$ as 100, and randomly choose style images from the whole training images for in-data way. Table 8 shows the fusion inference accuracy on Caltech-256 with traditional data augmentation: style images of in-data way perform better than out-of-data way. And the base test accuracy without traditional augmentation is 39.98%. We get 10+ improvement easily via style augmentation.

Table 8. Test accuracy (%) of our STDA-inf on Caltech-256 with traditional augmentation. The baseline is 60.85%. + means extending the training epoch to 200 and † is the reproduce result of compared work.

Style type	Style number	Base test	Rounds m			
			1	5	10	15
out random$^+$	100	63.36†	64.31	64.79	64.79	64.75
in random$^+$	100	**64.92**	65.79	66.18	66.35	**66.41**

5 Conclusion

In this paper, we have proposed a novel method of style augmentation named STDA-inf, which consists of in-data training and fusion inference. Style images of *in-data* way are more proper and targeted than *out-of-data* way during training since classifiers may overfit textures. In addition, the learned transfer distribution can be utilized during inference. Current methods of data augmentation harvest better performance combined with our method.

Improvements of style augmentation vary along with different transfer degrees, intense or soft. *Online* transfer mode creates much richer samples and is more flexible to control the transfer proportion compared to *offline* mode. As for the strategy of style choice, *intra-class* mode is superior to *inter-class* mode during training since it creates an adversarial effect. In future work, it's worth studying to reduce the time consumption of style augmentation (for example, using a unified network to accomplish style transfer and classification simultaneously), and apply it to other tasks such as object detection.

6 Appendix

Configuration of Compared Augmentation Methods
We illustrate the specific parameter configuration of compared data augmentation methods here. If the reference paper provides the experiments of STL-10, we adopt the parameters directly. If not, we try some simple tuning of parameter rs based on the reported datasets. Note that the traditional augmentation refers to horizontal flipping and random cropping.

- Mixup: No regulated parameters.
- Cutout: The mask area of square region is 24×24 without traditional augmentation and 32×32 with traditional augmentation.
- CutMix: The CutMix probability is 0.5 and distributional parameter $\beta = 1$.
- Manifold Mixup: We adopt the mixed layers as $[0, 1, 2]$. Then we try to take distributional parameter α as $0.2, 1, 2$, and get the best result when applying 1. But regrettably, this method still can't defeat the baseline. Maybe training for more epochs is necessary than the vanilla training, since Manifold Mixup is a strong regularizer.

Acknowledgment. This work was supported by the National Key Research and Development Program of China under grant 2018AAA0100205.

References

1. Yaeger, L.S., Lyon, R.F., Webb, B.J.: Effective training of a neural network character classifier for word recognition. In: Advances in Neural Information Processing Systems, pp. 807–816 (1997)
2. Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F.A., Brendel, W.: ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In: The International Conference on Learning Representations (ICLR) (2019)
3. Gatys, L.A., Ecker, A.S., Bethge, M.: A neural algorithm of artistic style. Nature Communications (2015)
4. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2414–2423 (2016)
5. Jackson, P.T., Atapour-Abarghouei, A., Bonner, S., Breckon, T.P., Obara, B.: Style augmentation: data augmentation via style randomization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 83–92 (2019)

6. Saenko, K., Kulis, B., Fritz, M., Darrell, T.: Adapting visual category models to new domains. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) Computer Vision – ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV, pp. 213–226. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15561-1_16
7. Zheng, X., Chalasani, T., Ghosal, K., Lutz, S., Smolic, A.: STaDA: style transfer as data augmentation. arXiv preprint arXiv:1909.01056 (2019)
8. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: beyond empirical risk minimization. In: The International Conference on Learning Representations (ICLR) (2018)
9. Huang, X., Belongie, S.: Arbitrary style transfer in real-time with adaptive instance normalization. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1501–1510 (2017)
10. Luan, F., Paris, S., Shechtman, E., Bala, K.: Deep photo style transfer. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4990–4998 (2017)
11. Li, Y., Fang, C., Yang, J., Wang, Z., Lu, X., Yang, M.H.: Universal style transfer via feature transforms. In: Advances in Neural Information Processing Systems, pp. 386–396 (2017)
12. Li, Y., Liu, M.-Y., Li, X., Yang, M.-H., Kautz, J.: A closed-form solution to photorealistic image stylization. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11207, pp. 468–483. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01219-9_28
13. Yoo, J., Uh, Y., Chun, S., Kang, B., Ha, J.W.: Photorealistic style transfer via wavelet transforms. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 9036–9045 (2019)
14. An, J., Xiong, H., Huan, J., Luo, J.: Ultrafast photorealistic style transfer via neural architecture search. In: AAAI, pp. 10443–10450 (2020)
15. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
16. Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., Raffel, C.A.: MixMatch: a holistic approach to semi-supervised learning. In: Advances in Neural Information Processing Systems, pp. 5050–5060 (2019)
17. DeVries, T., Taylor, G.W.: Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552 (2017)
18. Yun, S., Han, D., Oh, S.J., Chun, S., Choe, J., Yoo, Y.: CutMix: regularization strategy to train strong classifiers with localizable features. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 6023–6032 (2019)
19. Verma, V., et al.: Manifold mixup: better representations by interpolating hidden states. In: International Conference on Machine Learning, pp. 6438–6447. PMLR (2019)
20. He, K., Zhang, X., Ren, S., Jian, S.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
21. Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial examples in the physical world. arXiv preprint arXiv:1607.02533 (2016)
22. Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., Madry, A.: Adversarial examples are not bugs, they are features. In: Advances in Neural Information Processing Systems, pp. 125–136 (2019)
23. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
24. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083 (2017)