StyleMix: Separating Content and Style for Enhanced Data Augmentation

Minui Hong* Jinwoo Choi* Gunhee Kim
Seoul National University, Seoul, Korea
{alsdml123, jinwoo71, gunhee}@snu.ac.kr
https://github.com/alsdml/StyleMix

Abstract

In spite of the great success of deep neural networks for many challenging classification tasks, the learned networks are vulnerable to overfitting and adversarial attacks. Recently, mixup based augmentation methods have been actively studied as one practical remedy for these drawbacks. However, these approaches do not distinguish between the content and style features of the image, but mix or cut-and-paste the images. We propose StyleMix and StyleCutMix as the first mixup method that separately manipulates the content and style information of input image pairs. By carefully mixing up the content and style of images, we can create more abundant and robust samples, which eventually enhance the generalization of model training. We also develop an automatic scheme to decide the degree of style mixing according to the pair's class distance, to prevent messy mixed images from too differently styled pairs. Our experiments on CIFAR-10, CIFAR-100 and ImageNet datasets show that StyleMix achieves better or comparable performance to state of the art mixup methods and learns more robust classifiers to adversarial attacks.

1. Introduction

Although deep neural networks have achieved remarkable achievements in many classification tasks [18, 15, 31, 10], the learned networks are vulnerable to overfitting and adversarial attacks. As one of the most promising approaches to alleviate these issues, the data augmentation and regularization methods have been actively studied [30, 4, 29, 27, 1, 16]. As the earliest work, Mixup [30] augments a new sample by mixing two samples via interpolation of both images and labels. Instead of using the entire object regions, CutMix [29] cuts and pastes patches from one image onto the other image along with the ground truth labels being mixed proportionally to the area of patches. Manifold Mixup [27] regularizes to prevent the network from overfitting on the intermediate representations that interpolate hidden states. These approaches certainly improve classification performance and robustness over input noise and attacks, but they do not distinguish between the *content* and *style* of the images for generating new mixup samples. Content generally refers to the shape and form of an image, while the style primarily includes texture and color [5]. If convolutional neural networks are trained with no distinction between content and style, they tend to be biased to focus on a small faction of information to learn categories [6] (*e.g.* an elephant is identified using only gray skin texture no matter how the foreground looks like).

In this work, we argue that carefully mixing up the content and style of two input images can benefit creating more abundant and robust samples, which eventually enhance the generalization of model training. The style transfer methods [5, 14, 26, 19] have shown great progress recently and proved that deep networks encode not only the content but also the style information of images and offer the possibility to alter the distribution of low-level visual features in the image while preserving the semantic content. Thus, by separately considering the content and style of each input image, we have two levels of options to augment training data such as generating content-preserved images with diverse styles, varying foreground objects in the same styled images, or both. Thus, classifiers are fully learned to recognize both content and style features to enhance the classification performance and robustness over noise and attacks.

We propose *StyleMix* as a new mixup method for data augmentation that can generate various training samples through convex combinations of content and style characteristics (Figure 1). We then extend to *StyleCutMix* that allows sub-image level manipulation based on the cut-and-paste idea of CutMix [29]. Finally, we develop a scheme to automatically decide the degree of style mixing according to the class distance between a given pair of images. With classification experiments on CIFAR-10 [17], CIFAR-100 [17] and ImageNet datasets [3], each of our proposals nontrivially improves the classification performance and eventually attains comparative performance to state-of-the-

*Equal contribution

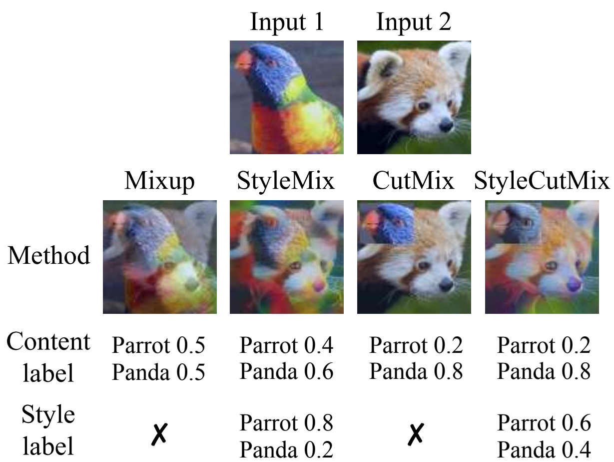

Figure 1: Visual comparison of our StyleMix and StyleCutMix with previous approaches using two inputs from Rainbow lorikeet and Red panda. While Mixup [30] and CutMix [29] are done at the image level, our methods separately consider the content and style of images to create more abundant and robust samples.

art mixup methods. The contributions of this work can be outlined as follows:

1. We propose StyleMix as the first mixup method that separately manipulates the content and style information of input image pairs, to the best of our knowledge.

2. Contrary to style transfer tasks where the content and style inputs are clearly identified, in the mixup context, two input images are any pairs from the training set. To prevent messy mixed images from too differently styled pairs that significantly harm the performance, we propose an automatic scheme to decide the degree of style mixing according to the pair's class distance.

3. In the experiments, StyleCutMix outperforms state-of-the-art mixup methods on CIFAR-10/100 and is comparable on ImageNet, including Cutout [4], GridMix [1], Manifold Mixup [27], CutMix [29], AugMix [11] and Puzzle Mix [16]. We also show that our methods improve the robustness of classifiers to adversarial attacks better than other recent mixup methods.

2. Related Works

Mixup. Mixup [30] trains a neural network to linearly interpolate two random training examples and their labels. Mixup encourages neural networks to favor simple linear behaviors in-between training images and eventually improves their generalization ability. Manifold Mixup [27] extends this linear interpolation from input-level to feature-level and makes the networks less confident in the hidden layer's interpolated representations. Instead of using all regions of the image, CutMix [29] cuts and pastes patches onto the other images and mixes the ground truth labels being proportionally to the area of patches. GridMix [1] divides two input images into grid cells and randomly selects each patch from the two images. Puzzle Mix [16] utilizes the saliency signal to resolve one issue of CutMix that does not discriminate foreground from background when cut-and-pasting patches. AugMix [11] proposes a simple data processing technique that utilizes stochasticiy and diverse augmentations for handling domain mismatch between training and test data. However, previous approaches do not distinguish between content and style of the images for mixing new samples. We decompose each image into separated representations of content and style and carefully mix up them to generate more abundant and robust samples.

Style Transfer. Style transfer algorithms change the style of the source image to the target style while maintaining the semantic content of the source image. Gatys et al. [5] demonstrate that feature representations derived from CNNs [18, 25] can separate and recombine the image content and style of natural images. However, this method is rather slow to optimize the content and style loss for generating a high-quality stylized image. Many following approaches [14, 26, 7, 12, 20] have been proposed to overcome these shortcomings and make style transfer possible to arbitrary styles in real time.

In the meantime, data augmentation based on style transfer algorithms has been studied. Neural augmentation [23] uses CycleGAN [33] to generate images of different styles. STaDA [32] employs neural style transfer as a data augmentation method to add more variation to the training dataset. They train each transformer network for every style individually, so the style set can be limited. Style augmentation [13] randomizes texture, contrast and color of images using the style transfer pipeline while preserving the content. Shape-texture debiased training [21] applies style transfer to a random image and then blend the labels of the paired images. Unlike ours, previous works have not considered mixup augmentation and focused on varying the styles of training images. On the other hands, our approach generates samples with not only any convex combination of both style and content but also any patch-wise cut and pastes from images. Also, our method automatically decides the degree of style mixing according to the pair's class distance, instead of a prefixed or simply randomized degree.

3. Approach

We propose StyleMix and StyleCutMix for data augmentation, considering both style and content of two training samples (section 3.1–3.2). We also present how to choose the degree of style mixing for any given pairs (section 3.3).

	f_{11}	f_{22}	f_{12}	f_{21}
Content	x_1	x_2	x_1	x_2
Style	x_1	x_2	x_2	x_1

Table 1: Given an input image pair (x_1, x_2), four image variables $f_{11/22/12/21}$ have the different content and style components from the input images.

3.1. StyleMix

Let $x_1, x_2 \in \mathbb{R}^{W\times H\times C}$ be the two input images, and y_1, y_2 be their labels. We aim to create a mixed image of x_m using two input images x_1 and x_2. Let the pre-trained style encoder and decoder be f and g, respectively. We adopt the same encoder-decoder architecture proposed in AdaIN [12]. We use AdaIN as our base style transfer because of its real-time computation and applicability to arbitarary styles. The encoder f is the first few layers up to `relu4_1` of a pre-trained VGG-19 [25]. The decoder g is an inverted encoder, except replacing all pooling layers with nearest up-sampling layers to avoid the checker-board effect. We pre-trained the encoder and the decoder on ImageNet [3].

We want to separate the content and style component of the input image x_1 and x_2 so that the output image x_m has a mixed content and style component of x_1 and x_2 with an arbitrary ratio. Since we make x_m using a linear interpolation, we start from four feature map variables as

$$f_{11} = f(x_1), \qquad f_{22} = f(x_2), \tag{1}$$
$$f_{12} = \text{AdaIN}(f_{11}, f_{22}), \;\; f_{21} = \text{AdaIN}(f_{22}, f_{11}). \tag{2}$$

$\text{AdaIN}(f_{ii}, f_{jj})$ is an adaptive instance normalization layer proposed in [12] as

$$\text{AdaIN}(f_{ii}, f_{jj}) = \sigma(f_{jj})(\frac{f_{ii} - \mu(f_{ii})}{\sigma(f_{ii})}) + \mu(f_{jj}). \tag{3}$$

The mean $\mu(.)$ and variance $\sigma(.)$ are computed across spatial dimension independently for each channel and sample.

Table 1 summarizes what content and style component four feature map variables have. For instance, f_{12} has the content component of x_1 and the style of x_2. Figure 2 shows output images passed through the pre-trained style decoder g to the four feature maps, and confirms that the decomposition of Table 1 is correct.

To obtain a new mixed image, we linearly interpolate these four feature maps with a content ratio r_c and a style ratio r_s. That is, unlike previous mixup methods, our method generates a sample at two levels of content and style, which subsequently leads to more abundant augmentation effects. Specifically, the mixed image x_m has the content component of x_1 as much as r_c (*i.e.* the content of x_2 with $1 - r_c$), and has the style component of x_1 as much as r_s (*i.e.* the style of x_2 with $1 - r_s$). With a free variable $\max(0, r_c + r_s - 1) \leq t \leq \min(r_c, r_s)$, the mixed image x_m is linearly interpolated from content and style components:

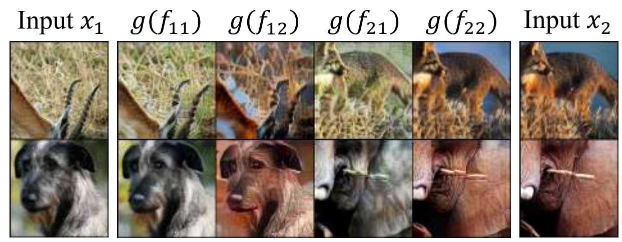

Figure 2: Two sets of examples showing four images through the pre-trained style transfer encoder and decoder $(g(f_{11}), g(f_{22}), g(f_{12}), g(f_{21}))$ for two input images (x_1, x_2). We confirm that $g(f_{11})$ has the content component of x_1 and the style of x_1, and $g(f_{12})$ has the content of x_1 and the style of x_2.

$$\begin{aligned} x_m = g(&tf_{11} + (1 - r_c - r_s + t)f_{22} \\ &+ (r_c - t)f_{12} + (r_s - t)f_{21}). \end{aligned} \tag{4}$$

Figure 3 visualizes how a mixed image x_m is obtained from Eq. (4) by adjusting the values of r_c and r_s in a grid. It also shows that the content and style components of the two images x_1 (top-left) and x_2 (bottom-right) are well separated by linear interpolation. Therefore, the mixup approach in Eq. (4) based on Table 1 is appropriate to well mix the content and style of two images.

The free parameter t in Eq. (4) is introduced to complete the linear interpolation between four feature maps. For example, if we want that a new sample x_m has 70% of content from x_1 (*i.e.* $r_c = 0.7$), we have to decide how much getting from f_{11} and f_{12}, both of which contain the content of x_1. t decides the ratio between f_{11} and f_{12}, and fortunately the outputs hardly change no matter how to set t. Figure 4 shows an example where even if the value of t changes, the resulting mixed image x_m barely changes.

Finally, we set the label based on the ratio of the content and style component. We first compute a content-based label y_c and style-based label y_s, and merge them to a final label y_m with a hyperparameter r, which controls how much the content and label labels are used for the final label:

$$y_m = ry_c + (1 - r)y_s, \quad \text{where} \tag{5}$$
$$y_c = r_c y_1 + (1 - r_c)y_2, \; y_s = r_s y_1 + (1 - r_s)y_2. \tag{6}$$

In training time, r_c and r_s are randomly sampled from the beta distribution $Beta(\alpha, \alpha)$ so that we use all combinations of various content and style for training.

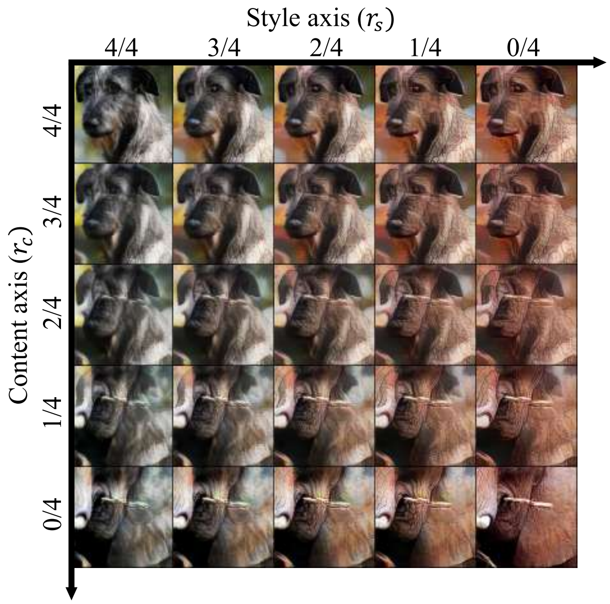

Figure 3: A grid visualization of mixed images by adjusting the content and style ratios (r_c, r_s). The top-left and the bottom-right are two input images. The content and style components are well separated; along the content and style axes, for a fixed r_c, varying r_s only changes the style while the content remains the same. Likewise, for a fixed r_s, the content only changes according to r_c with a fixed style.

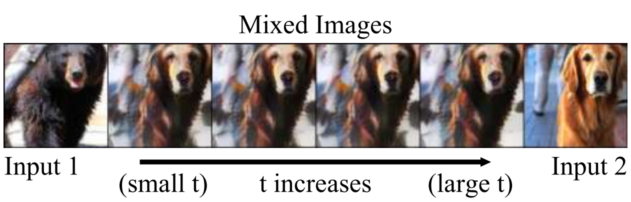

Figure 4: An example of creating a series of mixed images by adjusting the value of t for two input images. Even with varying t, the mixed images hardly change.

3.2. StyleCutMix

We extend our StyleMix to StyleCutMix by using the idea of CutMix [29], which defines a bounding box and creates a mixed image by filling the image inside the box with x_1 and outside with x_2. Since we separately consider the content and style components of images, we apply different cut-and-paste schemes for content and style. For content, it is the same as CutMix; the inside of the box is filled with the content component of x_1 and the outside with that of x_2. On the other hand, the scheme is different for style, which will be discussed below.

As done in StyleMix, we separate the content and style component of the two inputs x_1 and x_2, and start from four

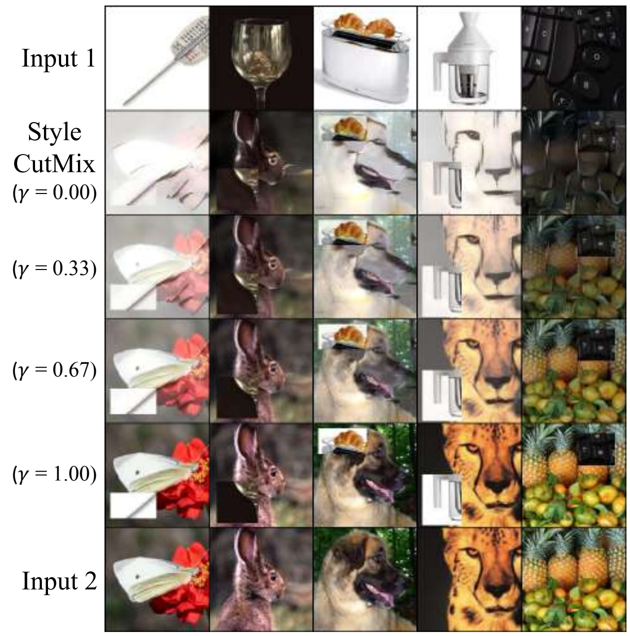

Figure 5: Examples of StyleCutMix with varying γ. When two input images have too different styles, the mixed images could be messy with low γ. In such cases, γ is adjusted to have a high value to reduce the degree of style mixing.

image variables:

$$g_{11} = x_1, \quad g_{22} = x_2, \tag{7}$$

$$g_{12} = \gamma x_1 + (1-\gamma) g(f_{12}), \tag{8}$$

$$g_{21} = \gamma x_2 + (1-\gamma) g(f_{21}). \tag{9}$$

One difference from StyleMix is that we run linear interpolation in the image space in Eq. (7)–(9) unlike the feature space in Eq. (1)–(2). It is because the feature map size is much smaller than the image size for patch cut-and-paste, and thus the performance degrades when using feature maps as reported in [29].

We assume that each variable includes content and style components as shown in Table 2, which is slightly different from Table 1. Since $g_{11} = x_1$, g_{11} surely has the content and style from x_1. So does g_{22}. On the other hand, as shown in Eq. (8), g_{12} has the content from x_1 (since both x_1 and f_{12} have the content from x_1), but has a mixed style from x_1 and x_2 with a ratio of $(\gamma, 1-\gamma)$ (since x_1 has style of x_1 and f_{12} has style of x_2). Thus, the new parameter γ adjusts the degree of style mixing; if $\gamma = 1$, the style mixing is the same as CutMix, since g_{12} now has the style of x_1 only.

The reason why we introduce the parameter γ is illustrated in Figure 5. In style transfer, a pair of content and style images are given as input with a clear intent that the new image must contain the content from the content image having the style from the style image. On the other hand,

	g_{11}	g_{22}	g_{12}	g_{21}
Content	x_1	x_2	x_1	x_2
Style	x_1	x_2	$\gamma x_1 + (1-\gamma)x_2$	$(1-\gamma)x_1 + \gamma x_2$

Table 2: Given an input image pair x_1, x_2, four image variables $g_{11/22/12/21}$ have the different content and style components from the input images.

in the mixup context, the two input images are any pairs of training images, and thus if they have severely different styles, the quality of mixed images would be severely poor (See examples with $\gamma = 0$ in Figure 5). This can be prevented by setting a high value of γ that suppresses the degree of style mixing between images. In next section, we will discuss a scheme to automatically set γ according to class similarity between x_1 and x_2.

As done in CutMix [29], we first sample the bounding box coordinates $\mathbf{B} = (r_x, r_y, r_w, r_h)$, crop the region from x_1, and paste it into x_2:

$$\begin{aligned} r_x &\sim \text{Unif }(0, W), \quad r_w = W\sqrt{\lambda}, \\ r_y &\sim \text{Unif }(0, H), \quad r_h = H\sqrt{\lambda}, \end{aligned} \tag{10}$$

where $\lambda = \frac{r_w r_h}{WH}$ is the ratio of the area occupied by x_1 in the mixed image. With the cropped region, the binary mask $\mathbb{R}_c \in \{0,1\}^{W\times H}$ is defined by 1 inside the box $\mathbf{B}$, otherwise 0. We then create a mixed image x_m by linearly interpolating g_{11}, g_{22}, g_{12} and g_{21} as

$$\begin{aligned} x_m = \mathbb{T} \odot g_{11} + (\mathbb{1} - \mathbb{R}_c - \mathbb{R}_s + \mathbb{T}) \odot g_{22} \\ + (\mathbb{R}_c - \mathbb{T}) \odot g_{12} + (\mathbb{R}_s - \mathbb{T}) \odot g_{21}, \end{aligned} \tag{11}$$

where $\odot$ is the element-wise multiplication and $\mathbb{T}$, $\mathbb{R}_c$, $\mathbb{R}_s$, $\mathbb{1} \in \mathbb{R}^{W\times H\times C}$. $\mathbb{1}$ is a matrix of ones, $\mathbb{R}_s = r_s\mathbb{1}$ with a scalar value r_s that controls the strength of style. $\mathbb{T}$ should satisfy $\max(0, \mathbb{R}_c + \mathbb{R}_s - \mathbb{1}) \le \mathbb{T} \le \min(\mathbb{R}_c, \mathbb{R}_s)$, but we can easily derive that always $\mathbb{T} = \max(0, \mathbb{R}_c + \mathbb{R}_s - \mathbb{1}) = \min(\mathbb{R}_c, \mathbb{R}_s)$.

Figure 6 shows grid examples of mixed images by Eq. (11). To see the effects of γ and r_s, we separately look into the inside and outside of the bounding box. The inside of the box is clearly composed of x_1's content, and the outside with x_2's content. The style ratio could be a little complicated; the styles of x_1 and x_2 are in the ratio of $1-(1-r_s)(1-\gamma)$ and $(1-r_s)(1-\gamma)$ inside the box, while in the ratio of $r_s(1-\gamma)$ and $1-r_s(1-\gamma)$ outside the box. Therefore, with $\gamma = 1$, it reduces to the normal CutMix (*i.e.* only x_1's style inside but only x_2's style outside). With $\gamma = 0$, both inside and outside have the styles of x_1 and x_2 in a ratio of r_s and $1-r_s$.

Finally, to determine the label y_m of x_m, let us calculate the content and style components of the entire mixed image.

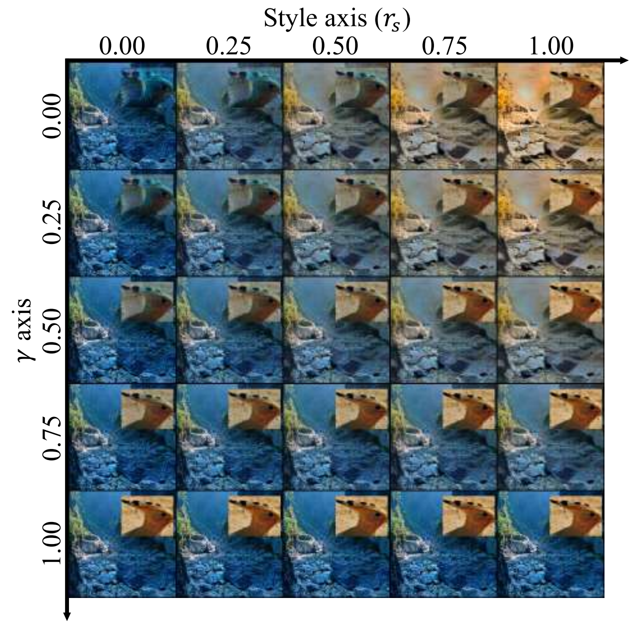

Figure 6: A grid visualization of mixed images by adjusting the degree of style mixing and style ratio (γ, r_s). When $\gamma = 0$, the styles for both inside and outside of the bounding box change in the same way. When $\gamma = 1$, the style changes are independent between inside and outside of the box, and thus StyleCutMix behaves identically to CutMix.

Given that the area of the bounding box is λ, the content ratios of x_1 and x_2 are λ and $1-\lambda$, and the style ratios are $\gamma\lambda + (1-\gamma)r_s$ and $1-(\gamma\lambda + (1-\gamma)r_s)$. Based on them, the label y_m is defined from the content label y_c and the style label y_s as

$$y_m = r y_c + (1-r) y_s, \quad \text{where} \tag{12}$$

$$y_c = \lambda y_1 + (1-\lambda) y_2, \; y_s = \lambda_s y_1 + (1-\lambda_s) y_2. \tag{13}$$

$\lambda_s = \gamma\lambda + (1-\gamma)r_s$, and r is a hyperparameter that adjusts the ratio of content and style label. During training, γ and r_s are randomly sampled from the beta distribution $Beta(\alpha, \alpha)$ to consider all possible combinations.

3.3. Style Mixing with the Style Distance

To find an adequate value of γ for any two training images x_1 and x_2, we need to measure how different the styles are between x_1 and x_2. Remind that a low value of γ for too differently styled x_1 and x_2 leads to a messy mixed image, as shown in Figure 5. Thus, we propose a style distance function D, which measures style differences between classes to which input images belong (instead of between images) to reduce the training cost. Naturally, if the value of D is large, the style distance between classes is large and γ is set high to prevent excessive style mixing.

Figure 7: Top-5 classes with the shortest distance D.

Suppose that there are K classes in the dataset and n_c is the number of images of class c. We let ϕ_i denote the output of layer i in VGG-19 used to compute the style loss where $i \in \{1, \dots, L\}$. We use `relu1_1`, `relu2_1`, `relu3_1`, `relu4_1` layers, and thus $L = 4$. We define the mean and variance of the feature map per channel for class c as

$$\bar{\mu}_{ci} = \frac{1}{n_c} \sum_{j=1}^{n_c} \mu(\phi_i(x_{cj})), \ \bar{\sigma}_{ci} = \frac{1}{n_c} \sum_{j=1}^{n_c} \sigma(\phi_i(x_{cj})). \quad (14)$$

As a result, we define two sets of variables that represent the class style: $\mathbb{M}_c = (\bar{\mu}_{c1}, \bar{\mu}_{c2}, \dots, \bar{\mu}_{cL})$ and $\Sigma_c = (\bar{\sigma}_{c1}, \bar{\sigma}_{c2}, \dots, \bar{\sigma}_{cL})$. In our implementation, the dimension of $|\mathbb{M}|$ and $|\Sigma|$ is $960 = 64 + 128 + 256 + 512$ with $L = 4$. We now define the style distance function D as a simple difference mean of two variables between class c_1 and c_2:

$$D(c_1, c_2) = \frac{1}{2|\mathbb{M}|} \|\mathbb{M}_{c_1} - \mathbb{M}_{c_2}\|_2^2 + \frac{1}{2|\Sigma|} \|\Sigma_{c_1} - \Sigma_{c_2}\|_2^2, \quad (15)$$

Figure 7 shows that the style distance D works reasonably as similar classes are retrieved as the closest neighbors.

Finally, we set γ for (x_1, x_2) whose classes are (c_1, c_2):

$$\gamma = tanh(D(c_1, c_2)/\delta), \quad (16)$$

where δ is a hyperparameter that controls the influence of style distance. We will experiment the effect of δ on the performance in section 4.4. Figure 8 shows an example where γ set by Eq. (16) can avoid the messy style mix between the images of quite different classes.

4. Experiments

Following previous works, we evaluate our StyleMix and StyleCutMix on CIFAR-100 [17], CIFAR-10 [17] and ImageNet datasets [3]. We first compare with other mixup methods for image classification tasks (section 4.2). Next, we

Figure 8: StyleCutMix with γ set by Eq. (16) can avoid messy mixed images in the third column.

show our methods can improve the robustness of classifiers to adversarial attacks (section 4.3). Finally, we carry out performance analysis according to hyperparameter choices (section 4.4). We provide more experimental results in Appendix, including diverse adversarial attacks, and additional baselines in image classification.

4.1. Experiment Setting

We report the results of three variants of our methods: StyleMix and StyleCutMix with and without auto-γ described in section 3.3, to clearly see the performance improvement by each key component of our methods.

CIFAR-10/100. We follow the experiment setting in CutMix [29]. The batch size is 64, and the training epochs are 300. The learning rate starts at 0.25 and is decayed by a factor of 0.1 at epoch 150 and 225. As done in [29], we use a strong base classifier PyramidNet-200 [9] of a widening factor $\tilde{\alpha} = 240$ with 26.8M parameters. We set hyperparameters as follows; (i) $r = 0.7$, (ii) StyleMix: random sampling of $r_c, r_s \sim Beta(0.5, 0.5)$, and (iii) StyleCutMix: random sampling of $\gamma \sim Beta(0.8, 0.8)$ without auto-γ, and $\delta = 3.0$ for CIFAR-100 and 1.0 for CIFAR-10 with auto-γ.

ImageNet. We use ResNet-50 as the base classifier. We train models for 100 epochs. The batch size is set to 1024 until epoch 15, and 448 after that. For more efficient training, we use pre-resized images and the cycling learning rate [28] with the same scheduling strategy as Puzzle Mix [16]. The images are resized to 160×160 and 352×352 in advance; we use 160×160 up to epoch 15, and 352×352 after that, following [28]. We set hyperparameters as follows; (i) StyleMix: $r = 0.7$ and random sampling of both $r_c, r_s \sim Beta(0.5, 0.5)$, (ii) StyleCutMix: $r = 0.7$ and random sampling of $\gamma \sim Beta(0.8, 0.8)$ without, and $r = 0.8$ and $\delta = 0.5$ with auto-γ.

Model + Method	Top-1 Err(%)	Top-5 Err(%)
WRN28-4 + GridMix [1]*	18.51	-
WRN28-10 + Puzzle Mix [16]‡	15.95	3.92
WRN28-10 + Puzzle Mix (half) [16]‡	16.23	3.90
Pyramid-200 + Baseline†	16.45	3.69
Pyramid-200 + RA(N=1,M=2) [2]	15.86	3.32
Pyramid-200 + Cutout [4]†	16.53	3.65
Pyramid-200 + Mixup (α=1.0) [30]†	15.63	3.99
Pyramid-200 + Manifold (α=1.0) [27]†	16.14	4.07
Pyramid-200 + CutMix [29]†	14.47	2.97
Pyramid-200 + StyleMix	16.37	3.69
Pyramid-200 + StyleCutMix	14.61	2.95
Pyramid-200 + StyleCutMix(auto-γ)	**14.17**	**2.66**

Table 3: Comparison with state-of-the-art mixup methods on CIFAR-100. We include results cited from GridMix [1]*,CutMix [29]† and Puzzle Mix [16]‡. GridMix did not report Top-5 Err on their paper.

PyramidNet-200 ($\tilde{\alpha}$=240)	Top-1 Err(%)
Baseline†	3.85
RA(N=3,M=4) [2]	3.44
Cutout [4]†	3.10
Mixup (α=1.0) [30]†	3.09
Manifold Mixup (α=1.0) [27]†	3.15
CutMix [29]†	2.88
StyleMix	3.56
StyleCutMix	2.79
StyleCutMix (auto-γ)	**2.55**

Table 4: Comparison with state-of-the-art mixup methods on CIFAR-10. We include results cited from CutMix [29]†.

4.2. Classification Results

CIFAR-100/10. Table 3 compares our StyleMix methods with other state-of-the-art mixup models on CIFAR-100. StyleCutMix (auto-γ) achieves the best top-1 and top-5 error rates among other augmentation strategies. The automatic selection of γ using the style distance in section 3.3 improves StyleCutMix with a non-trivial margin of 0.44%p. Table 4 summarizes the results on CIFAR-10. Similar to the results on CIFAR-100, we find that StyleCutMix (auto-γ) performs the best scores on CIFAR-10 too. StyleMix itself is not as good as CutMix, but StyleCutMix outperforms the CutMix using the idea of cut-and-paste. The performance gaps increase further with adding the automatic selection of γ since it can prevent the creation of messy images.

Model	Top-1 Err(%)	Top-5 Err(%)
Vanilla†	24.31	7.34
Input†	22.99	6.48
Manifold [27]†	23.15	6.50
CutMix [29]†	22.92	6.55
AugMix [11]†	23.25	6.70
Puzzle Mix [16]†	**22.49**	**6.24**
StyleMix	24.06	7.09
StyleCutMix	23.42	6.75
StyleCutMix(auto-γ)	22.71	6.36

Table 5: Top-1/Top-5 error rates on ImageNet on ResNet-50. We include results cited from Puzzle Mix [16]†.

ImageNet. Table 5 shows the top-1/top-5 errors of our StyleMix methods and other state-of-the-art mixup models on ImageNet. Our StyleCutMix (auto-γ) outperforms most existing methods and is comparable to the best performing Puzzle Mix. In ImageNet, StyleCutMix shows lower performance than CutMix mainly because as the number of classes increases, the creation of messy images is more likely as the chances for mixing too different classes increase. However, the automatic selection of γ rescues StyleCutMix (auto-γ) to be significantly improved over CutMix.

We observe that StyleCutMix attains slightly lower performance than Puzzle Mix since Puzzle Mix can prevent foreground objects of two images from overlapping, which could be especially helpful in ImageNet. Nonetheless, note that our key idea, separation of content and style, is orthogonal to any mixup methods, and thus it can be applied on top of Puzzle Mix.

4.3. Robustness to Adversarial Attacks

Deep Neural Networks can be easily fooled by even a slight perturbation on input images, coined as adversarial attacks [8]. One way to prevent this is to create and learn new samples made by data augmentation [22, 30, 29]. We test FGSM attack to evaluate how robust each data augmentation strategy is against adversarial attack. The FGSM (Fast Gradient Sign Method) [8] is a white-box attack, assuming that adversary has all the information about the model.

We apply FGSM attack on the CIFAR-100 validation set with ℓ_∞ $\epsilon = \{1, 2, 4\}/255$. Table 6 reports the Top-1 error rates, where StyleCutMix greatly improves robustness to adversarial attacks compared to other state-of-the-art augmentation strategies. This may be because StyleCutMix can augment images with a wide variation of content and style for each class, which eventually train the model more robust against adversarial attacks.

Interestingly, the automatic selection of γ slightly de-

Method	FGSM (1) Top-1 Err(%)	FGSM (2) Top-1 Err(%)	FGSM (4) Top-1 Err(%)
Baseline	63.68	74.70	80.39
Cutout [4]	67.35	76.45	83.38
CutMix [29]	58.33	66.89	75.49
StyleMix	60.97	74.31	81.99
StyleCutMix	**58.09**	**65.88**	**73.46**
StyleCutMix(auto-γ)	58.81	66.15	74.72

Table 6: Top-1 error rates of multiple mixup methods on CIFAR-100 dataset when FGSM Attack is applied. The baseline is the vanilla PyramidNet-200 model.

grades the defense performance of StyleCutMix. Surely, it can improve the generalization performance of StyleCutMix as shown in classification results in section 4.2 but much wider style mixing may make the model more robust against adversarial attack. That is, messy images due to excessive style mixing could be helpful here.

4.4. Performance Analyses

In this section, we investigate the performance variation of StyleCutmix according to two hyperparameters, which can be determined via cross-validation in practice.

We first compare the performance of StyleCutmix (auto-γ) by changing the δ parameter of Eq. (16), which controls the influence of style distance. Remind that as δ decreases, γ increases and the style mixing is amplified. Table 7 shows that the performance of StyleCutMix improves when the auto-γ method is applied. With appropriate setting of δ, the performance by auto-γ is maximized.

We also inspect the performance variation according to hyperparameter $r = [0.3, 0.5, 0.7, 1]$, which controls how much the content label is considered to determine the final label y_m. We observe that the performance is the best at $r = 0.7$; the content is more important than the style to decide the final mixed label, which meets our intuition. Nevertheless, considering both content and style is still better in performance than considering only the content that existing augmentation strategies do. It partly agrees with that CNNs often learn to focus on the texture, one of key elements of style, to recognize the objects [6].

5. Conclusion

We proposed StyleMix and StyleCutMix as a data augmentation strategy that separates the content and style component of images and flexibly mixes them to create new data samples. In addition, we developed a method to automatically control the degree of style mixing by calculating the style distance between classes. Our method improved both image classification accuracy and the robustness to adversarial attacks in our experiments on CIFAR-10, CIFAR-100, and ImageNet. Looking forward, it would be interesting to further separate foreground from background to minimize the effect of background when obtaining content and style components from images.

Method	Top-1 Err(%)	Top-5 Err(%)
StyleCutMix	14.61	2.95
StyleCutMix(auto-γ, $\delta = 1.5$)	14.35	2.98
StyleCutMix(auto-γ, $\delta = 2.0$)	14.52	3.00
StyleCutMix(auto-γ, $\delta = 3.0$)	**14.17**	**2.66**
StyleCutMix(auto-γ, $\delta = 4.0$)	14.42	2.93

Table 7: Performance variation of StyleCutMix(auto-γ) according to the hyperparameter δ of Eq. (16) on CIFAR-100.

Method	Top-1 Err(%)	Top-5 Err(%)
StyleCutMix(auto-γ, $r = 0.3$)	15.01	3.18
StyleCutMix(auto-γ, $r = 0.5$)	14.77	3.00
StyleCutMix(auto-γ, $r = 0.7$)	**14.17**	**2.66**
StyleCutMix(auto-γ, $r = 1.0$)	14.96	2.97

Table 8: Performance variation of StyleCutMix(auto-γ) according to the hyperparameter r of Eq. (12) on CIFAR-100.

(a)
(b)
(c)

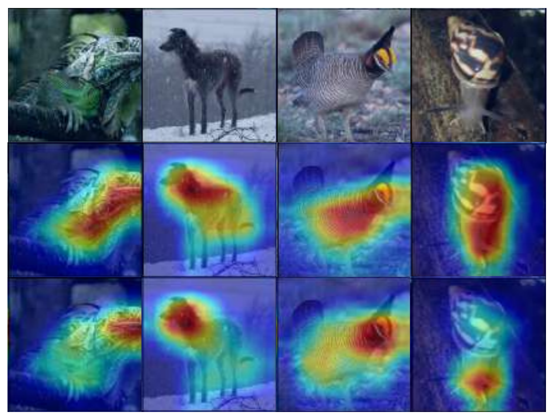

Figure 9: Grad-CAM [24] visualization comparing StyleMix (auto-γ) with CutMix [29]. (a) Input, (b) StyleCutMix (auto-γ), (c) CutMix. CutMix often focuses on only representative parts of content, such as snail's eye, the bird's beak, and the animal's head. On the other hand, StyleCutMix (auto-γ) detects not only content but also class-specific styles such as skin patterns and colors.

Acknowledgments This work was supported by SAMSUNG DISPLAY Co., Ltd and Samsung Research Funding Center of Samsung Electronics (No. SRFC-IT1502-51). Gunhee Kim is the corresponding author.

References

[1] Kyungjune Baek, Duhyeon Bang, and Hyunjung Shim. Gridmix: Strong regularization through local context mapping. *Pattern Recognition*, 109:107594, 2021. 1, 2, 7

[2] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *NeurIPS*, 2020. 7

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 1, 3, 6

[4] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017. 1, 2, 7, 8

[5] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016. 1, 2

[6] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. In *ICLR*, 2019. 1, 8

[7] Golnaz Ghiasi, Honglak Lee, Manjunath Kudlur, Vincent Dumoulin, and Jonathon Shlens. Exploring the structure of a real-time, arbitrary neural artistic stylization network. In *BMVC*, 2017. 2

[8] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 7

[9] Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep pyramidal residual networks. In *CVPR*, 2017. 6

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 1

[11] Dan Hendrycks, Norman Mu, Ekin D. Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. AugMix: A simple data processing method to improve robustness and uncertainty. *ICLR*, 2020. 2, 7

[12] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017. 2, 3

[13] Philip TG Jackson, Amir Atapour Abarghouei, Stephen Bonner, Toby P Breckon, and Boguslaw Obara. Style augmentation: data augmentation via style randomization. In *CVPR Workshops*, pages 83–92, 2019. 2

[14] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, pages 694–711. Springer, 2016. 1, 2

[15] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. 1

[16] Jang-Hyun Kim, Wonho Choo, and Hyun Oh Song. Puzzle mix: Exploiting saliency and local statistics for optimal mixup. In *ICML*, 2020. 1, 2, 6, 7

[17] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. 1, 6

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012. 1, 2

[19] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *ECCV*, 2016. 1

[20] Xueting Li, Sifei Liu, Jan Kautz, and Ming-Hsuan Yang. Learning linear transformations for fast image and video style transfer. In *CVPR*, 2019. 2

[21] Yingwei Li, Qihang Yu, Mingxing Tan, Jieru Mei, Peng Tang, Wei Shen, Alan Yuille, and Cihang Xie. Shape-texture debiased neural network training. *arXiv preprint arXiv:2010.05981*, 2020. 2

[22] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018. 7

[23] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017. 2

[24] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, 2017. 8

[25] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 2, 3

[26] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor S Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, 2016. 1, 2

[27] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *ICML*, 2019. 1, 2, 7

[28] Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. In *ICLR*, 2019. 6

[29] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *CVPR*, 2019. 1, 2, 4, 5, 6, 7, 8

[30] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018. 1, 2, 7

[31] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *NeurIPS*, 2015. 1

[32] Xu Zheng, Tejo Chalasani, Koustav Ghosal, Sebastian Lutz, and Aljosa Smolic. Stada: Style transfer as data augmentation. In *VISAPP*, 2019. 2

[33] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017. 2