A. Detailed Description of StyleCutMix

This section provides some detailed explanations for the process of obtaining the style component inside and outside the bounding box, and the process of calculating the value of the style label coefficient $\lambda_s$.

In StyleCutMix, we define a mixed image $x_m$ as Eq. (1).

$$x_m = T \odot g_{11} + (I - R_c - R_s + T) \odot g_{22} + (R_c - T) \odot g_{12} + (R_s - T) \odot g_{21}.$$  

Before calculating the style component, it is necessary to set $T$ as the specific value. $T$ is a matrix that satisfies $\max(0, R_c + R_s - I) \leq T \leq \min(R_c, R_s)$. But here $T$ is always fixed at $T = \max(0, R_c + R_s - I) = \min(R_c, R_s)$, which is because the element of $R_c$ is always 0 or 1.

Inside the bounding box $B$, the elements of $R_c$ are always 1, so $\max(0, R_c + R_s - I)$ is naturally identical to $R_s$, which is the same with $\min(R_c, R_s)$. Too. Inside the bounding box $B$, the elements of $R_c$ are always 0, and thus $\max(0, R_c + R_s - I) = 0$ and $\min(R_c, R_s) = 0$. Therefore, both inside and outside of the box $B$ always satisfy $T = \max(0, R_c + R_s - I) = \min(R_c, R_s)$.

Since the content component can be obtained as done in CutMix, let us consider only the style component. Inside the box $B$, as mentioned earlier, the elements of $R_c$ have a value of 1, and the elements of $T$ are $r_s$.

To simplify Eq. (1), let us use $x_{m, in}, g_{11, in}, g_{12, in}$ to denote the inside the box of $x_m, g_{11}, g_{12}$, respectively. Then Eq. (2) is established from Eq. (1):

$$x_{m, in} = r_s g_{11, in} + (1 - r_s) g_{12, in}$$

The reason that $g_{21}$ and $g_{22}$ disappear in the equation is because inside the box the elements of $T$ are $r_s$ and the elements of $R_c$ are 1, so the coefficients of $g_{21}, g_{22}$ in Eq. (1) become 0. Since $g_{11}$ has the style of $x_1$, and $g_{12}$ has the style of $x_1$ as much as the degree of style mixing $\gamma$, $x_{m, in}$ has style of $x_1$ with $r_s + (1 - r_s) \gamma = 1 - (1 - r_s)(1 - \gamma)$, as mentioned in the paper.

Similarly, let us use $x_{m, out}, g_{21, out}, g_{22, out}$ to denote the outside part of $x_m, g_{21}, g_{22}$, respectively. As mentioned earlier, the elements of $T$ and $R_c$ are 0, so Eq. (3) holds:

$$x_{m, out} = (1 - r_s) g_{22, out} + r_s g_{21, out}$$

Since $g_{22}$ has no style of $x_1$, and $g_{22}$ has the style of $x_1$ as much as $(1 - \gamma)$, so $x_{m, out}$ has style of $x_1$ as much as $r_s(1 - \gamma)$, as mentioned in the paper.

In summary, the style of $x_1$ inside the bounding box is with a ratio of $1 - (1 - r_s)(1 - \gamma)$, and the style of $x_1$ outside the bounding box is with $r_s(1 - \gamma)$. Therefore, $\lambda_s$, the style label coefficient of $y_1$, is calculated as $\lambda(1 - (1 - r_s)(1 - \gamma)) + (1 - \lambda)r_s(1 - \gamma) = \gamma \lambda + (1 - \gamma)r_s$.

B. Algorithm

Algorithm 1–2 show the pseudo-codes for learning StyleMix and StyleCutMix. In Algorithm 2, $D$ is the style distance function defined in the paper. All of our methods have the similar pipeline: creating a mixed image using input images, obtaining the content and style label of the mixed image, and learning the model using the loss calculated by these labels. We use the cross-entropy loss.
Algorithm 2 StyleCutMix pseudo-code

for Image $x_1$, Target $y_1$ = Batch(Data) do

$x_2, y_2$ = randShuffle($x_1, y_1$)
$bbw_1, bbw_2, bbh_1, bbh_2 = $ getBoundingBox()
$W, H = $ getWidthAndHeight($x_1$)
$\lambda = (bbw_2 - bbw_1)(bbh_2 - bbh_1) / (WH)$
Sample $r_s \sim Beta(\alpha_1, \alpha_1)$
$R_s = r_s I$
$R_c = $ zeros_like($R_s$)
$R_c[:i, :, bbw_1 : bbw_2, bhh_1 : bhh_2] = 1$
$T = $ max($0, R_c + R_s - I$)
if method is auto-γ then

$\gamma = tanh(D(y_1, y_2) / \delta)$

end if

$\gamma_1, \gamma_2 = $ encoder($x_1$), encoder($x_2$)
$\gamma_1, \gamma_2 = $ AdaIN($\gamma_1$, $\gamma_2$), AdaIN($\gamma_2$, $\gamma_1$)
$g_1, g_2 = $ $x_1, x_2$
$g_1 = \gamma(x_1) + (1 - \gamma) decoder(f_{12})$
$g_2 = \gamma(x_2) + (1 - \gamma) decoder(f_{21})$
$x_m = T \odot g_1 + (1 - R_c - R_s + T) \odot g_2 + (R_c - T) \odot g_2 + (R_s - T) \odot g_2$
$\lambda_s = \gamma + (1 - \gamma) y_s$
$y_c = \lambda y_1 + (1 - \lambda) y_2$
$y_s = \lambda y_1 + (1 - \lambda) y_2$
$y_m = r y_c + (1 - r) y_s$
output = model($x_m$)
loss = Loss(output, $y_m$)
loss.backward
optimizer.step

end for

C. Results of FGSM Attacks on CIFAR-10

Same as in CIFAR-100, we apply FGSM attack on the CIFAR-10 validation set with $\ell_\infty \epsilon = \{1, 2, 4\} / 255$. Table 1 shows the Top-1 error rates; StyleCutMix and StyleCutMix(auto-γ) improve robustness compared to other augmentation strategies.

D. Results of ResNet50

We use ResNet50 as the base classifier instead of PyramidNet [2] to evaluate the generalization of our method. Tables 2–3 show the performance with ResNet50 on CIFAR-10/100. In CIFAR-10, StyleCutMix (auto-γ) outperforms other state-of-the-art augmentation strategies. In CIFAR-100, StyleCutMix (auto-γ) records Top1-error 0.05% lower than PuzzleMix, which may be because messy images tend to occur more frequently as the number of classes increase.

E. Results of PGD Attacks

We evaluate whether our methods improve robustness against another adversarial attack type other than FGSM. We select the PGD (Projected Gradient Descent) Attack [4] with ResNet50 on CIFAR-10/100. We apply PGD attacks on the CIFAR-10 validation set with the following settings of $\ell_\infty \epsilon = 8 / 255$, a step size of $\alpha = 2 / 255$ and the number of steps = \{4, 6, 8\}. In CIFAR-100, we use
Table 4: Top-1 error rates of multiple mixup methods on CIFAR-10 dataset when PGD Attacks are applied. The baseline is the vanilla ResNet50 model.

<table>
<thead>
<tr>
<th>Method</th>
<th>PGD (4)</th>
<th>PGD (6)</th>
<th>PGD (8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>64.36</td>
<td>72.64</td>
<td>76.63</td>
</tr>
<tr>
<td>Cutout [1]</td>
<td>70.11</td>
<td>78.84</td>
<td>82.59</td>
</tr>
<tr>
<td>CutMix [5]</td>
<td>52.03</td>
<td>58.82</td>
<td>62.13</td>
</tr>
<tr>
<td>PuzzleMix [3]</td>
<td>52.93</td>
<td>59.09</td>
<td>61.79</td>
</tr>
<tr>
<td>StyleMix</td>
<td>69.73</td>
<td>75.50</td>
<td>77.92</td>
</tr>
<tr>
<td>StyleCutMix</td>
<td>47.96</td>
<td>53.04</td>
<td>55.76</td>
</tr>
<tr>
<td>StyleCutMix(auto-γ)</td>
<td>51.97</td>
<td>60.62</td>
<td>65.39</td>
</tr>
</tbody>
</table>

Table 5: Top-1 error rates of multiple mixup methods on CIFAR-100 dataset when PGD Attacks are applied. The baseline is the vanilla ResNet-50 model.

<table>
<thead>
<tr>
<th>Method</th>
<th>PGD (4)</th>
<th>PGD (6)</th>
<th>PGD (8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>69.29</td>
<td>71.83</td>
<td>73.25</td>
</tr>
<tr>
<td>Cutout [1]</td>
<td>65.46</td>
<td>69.55</td>
<td>71.23</td>
</tr>
<tr>
<td>CutMix [5]</td>
<td>56.77</td>
<td>59.60</td>
<td>61.10</td>
</tr>
<tr>
<td>PuzzleMix [3]</td>
<td>64.93</td>
<td>68.99</td>
<td>70.76</td>
</tr>
<tr>
<td>StyleMix</td>
<td>63.68</td>
<td>66.65</td>
<td>67.89</td>
</tr>
<tr>
<td>StyleCutMix</td>
<td>54.31</td>
<td>58.67</td>
<td>60.39</td>
</tr>
<tr>
<td>StyleCutMix(auto-γ)</td>
<td>57.91</td>
<td>62.66</td>
<td>65.28</td>
</tr>
</tbody>
</table>

$\ell_\infty \epsilon = 4/255$, a step size of $\alpha = 1/255$ and the number of steps = \{4, 6, 8\}. Tables 4–5 report the top 1-errors for PGD Attacks. StyleCutMix greatly improves the robustness than other augmentation methods.

References

[5] 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. 2, 3
Figure 1: A grid visualization of mixed images created by adjusting the content and style ratios \((r_c, r_s)\) in StyleMix.
Figure 2: A grid visualization of mixed images created by adjusting the content and style ratios \((r_c, r_s)\) in StyleMix.
Figure 3: Two sets of grid visualizations of mixed images created by adjusting the style ratios $r_s$ in StyleCutMix and StyleCutMix (auto-$\gamma$).