[Supplementary Material]
StyleMix: Separating Content and Style for Enhanced Data Augmentation

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 λ_s.

In StyleCutMix, we define a mixed image x_m as Eq. (1). Remind that $\mathbb{R}_s = r_s\mathbb{1}$ $(0 \leq r_s \leq 1)$ and $\mathbb{R}_c \in \{0,1\}^{W\times H}$ are defined by 1 inside the box $\mathbf{B}$, otherwise 0.

$$x_m = \mathbb{T} \odot g_{11} + (\mathbb{1} - \mathbb{R}_c - \mathbb{R}_s + \mathbb{T}) \odot g_{22} + (\mathbb{R}_c - \mathbb{T}) \odot g_{12} + (\mathbb{R}_s - \mathbb{T}) \odot g_{21}. \quad (1)$$

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

Inside the bounding box $\mathbf{B}$, the elements of $\mathbb{R}_c$ are always 1, so $\max(0, \mathbb{R}_c + \mathbb{R}_s - \mathbb{1})$ is naturally identical to $\mathbb{R}_s$, which is the same with $\min(\mathbb{R}_c, \mathbb{R}_s)$, too. Outside the bounding box $\mathbf{B}$, the elements of $\mathbb{R}_c$ are always 0, and thus $\max(0, \mathbb{R}_c + \mathbb{R}_s - \mathbb{1}) = 0$ and $\min(\mathbb{R}_c, \mathbb{R}_s) = 0$. Therefore, both inside and outside of the box $\mathbf{B}$ always satisfy $\mathbb{T} = \max(0, \mathbb{R}_c + \mathbb{R}_s - \mathbb{1}) = \min(\mathbb{R}_c, \mathbb{R}_s)$.

Since the content component can be obtained as done in CutMix, let us consider only the style component. Inside the box $\mathbf{B}$, as mentioned earlier, the elements of $\mathbb{R}_c$ have a value of 1, and the elements of $\mathbb{T}$ 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):

$$x_{m,in} = r_s g_{11,in} + (1 - r_s) g_{12,in} \quad (2)$$

The reason that g_{21} and g_{22} disappear in the equation is because inside the box the elements of $\mathbb{T}$ are r_s and the elements of $\mathbb{R}_c$ 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 γ, $x_{m,in}$ 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 $\mathbb{T}$ and $\mathbb{R}_c$ are 0, so Eq. (3) holds:

$$x_{m,out} = (1 - r_s) g_{22,out} + r_s g_{21,out} \quad (3)$$

Since g_{22} has no style of x_1, and g_{21} 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, λ_s, the style label coefficient of y_1, is calculated as $\lambda(1 - (1 - r_s)(1-\gamma)) + (1-\lambda) r_s (1-\gamma) = \gamma\lambda + (1-\gamma) r_s$.

Algorithm 1 StyleMix pseudo-code

```
for Image x_1, Target y_1 = Batch(Data) do
    x_2, y_2 = randShuffle(x_1, y_1)
    Sample r_c, r_s ~ Beta(α, α)
    Sample t ~ Unif(max(0, r_c + r_s − 1), min(r_c, r_s))
    f_11, f_22 = encoder(x_1), encoder(x_2)
    f_12, f_21 = AdaIN(f_11, f_22), AdaIN(f_22, f_11)
    x_m = decoder(t f_11 + (1 − r_c − r_s + t) f_22 + (r_c − t) f_12 + (r_s − t) f_21)
    y_c = r_c y_1 + (1 − r_c) y_2
    y_s = r_s y_1 + (1 − r_s) 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
```

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 $r_s \sim Beta(\alpha_1, \alpha_1)$
  $\mathbb{R}_s = r_s \mathbb{1}$
  $\mathbb{R}_c$ = zeros_like($\mathbb{R}_s$)
  $\mathbb{R}_c[:, :, bbw_1 : bbw_2, bbh_1 : bbh_2] = 1$
  $\mathbb{T} = \max(0, \mathbb{R}_c + \mathbb{R}_s - \mathbb{1})$
  if method is auto-γ **then**
    $\gamma = tanh(D(y_1, y_2)/\delta)$
  else
    Sample $\gamma \sim Beta(\alpha_2, \alpha_2)$
  end if
  f_{11}, f_{22} = encoder(x_1), encoder(x_2)
  f_{12}, f_{21} = AdaIN(f_{11}, f_{22}), AdaIN(f_{22}, f_{11})
  $g_{11}, g_{22} = x_1, x_2$
  $g_{12} = \gamma(x_1) + (1 - \gamma)$ decoder(f_{12})
  $g_{21} = \gamma(x_2) + (1 - \gamma)$ decoder(f_{21})
  $x_m = \mathbb{T} \odot g_{11} + (\mathbb{1} - \mathbb{R}_c - \mathbb{R}_s + \mathbb{T}) \odot g_{22} + (\mathbb{R}_c - \mathbb{T}) \odot g_{12} + (\mathbb{R}_s - \mathbb{T}) \odot g_{21}$
  $\lambda_s = \gamma\lambda + (1 - \gamma) r_s$
  $y_c = \lambda y_1 + (1 - \lambda) y_2$
  $y_s = \lambda_s y_1 + (1 - \lambda_s) y_2$
  $y_m = r y_c + (1 - r) y_s$
  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 ℓ_∞ $\epsilon = \{1, 2, 4\}/255$. 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.

Method	FGSM (1) Top-1 Err(%)	FGSM (2) Top-1 Err(%)	FGSM (4) Top-1 Err(%)
Baseline	41.75	59.60	70.18
Cutout [1]	41.61	60.44	71.28
CutMix [5]	28.28	34.45	40.61
StyleMix	26.47	34.69	42.78
StyleCutMix	25.64	32.57	**40.02**
StyleCutMix(auto-γ)	**16.61**	**22.36**	48.50

Table 1: Top-1 error rates of multiple mixup methods on CIFAR-10 dataset when FGSM Attack is applied. The baseline is the vanilla PyramidNet-200 model.

Model + Method	Top-1 Err(%)
ResNet50 + Baseline	5.96
ResNet50 + Cutout [1]	4.57
ResNet50 + CutMix [5]	4.40
ResNet50 + PuzzleMix [3]	4.57
ResNet50 + StyleMix	5.83
ResNet50 + StyleCutMix	4.41
ResNet50 + StyleCutMix(auto-γ)	**4.07**

Table 2: Comparison with state-of-the-art mixup methods on CIFAR-10 with ResNet50. The baseline is the vanilla ResNet50 model.

Model + Method	Top-1 Err(%)	Top-5 Err(%)
ResNet50 + Baseline	22.37	5.89
ResNet50 + Cutout [1]	23.13	6.38
ResNet50 + CutMix [5]	19.67	4.65
ResNet50 + PuzzleMix [3]	**17.16**	**4.19**
ResNet50 + StyleMix	23.28	6.56
ResNet50 + StyleCutMix	17.59	4.42
ResNet50 + StyleCutMix(auto-γ)	17.21	4.65

Table 3: Comparison with state-of-the-art mixup methods on CIFAR-100 with ResNet50. The baseline is the vanilla ResNet50 model.

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 ℓ_∞ $\epsilon = 8/255$, a step size of $\alpha = 2/255$ and the number of steps $= \{4, 6, 8\}$. In CIFAR-100, we use

Method	PGD (4) Top-1 Err(%)	PGD (6) Top-1 Err(%)	PGD (8) Top-1 Err(%)
Baseline	64.36	72.64	76.63
Cutout [1]	70.11	78.84	82.59
CutMix [5]	52.03	58.82	62.13
PuzzleMix [3]	52.93	59.09	61.79
StyleMix	69.73	75.50	77.92
StyleCutMix	**47.96**	**53.04**	**55.76**
StyleCutMix(auto-γ)	51.97	60.62	65.39

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.

Method	PGD (4) Top-1 Err(%)	PGD (6) Top-1 Err(%)	PGD (8) Top-1 Err(%)
Baseline	69.29	71.83	73.25
Cutout [1]	65.46	69.55	71.23
CutMix [5]	56.77	59.60	61.10
PuzzleMix [3]	64.93	68.99	70.76
StyleMix	63.68	66.65	67.89
StyleCutMix	**54.31**	**58.67**	**60.39**
StyleCutMix(auto-γ)	57.91	62.66	65.28

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.

ℓ_∞ $\epsilon = 4/255$, 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

[1] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017. 2, 3

[2] Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep pyramidal residual networks. In *CVPR*, 2017. 2

[3] Jang-Hyun Kim, Wonho Choo, and Hyun Oh Song. Puzzle mix: Exploiting saliency and local statistics for optimal mixup. In *ICML*, 2020. 2, 3

[4] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018. 2

[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

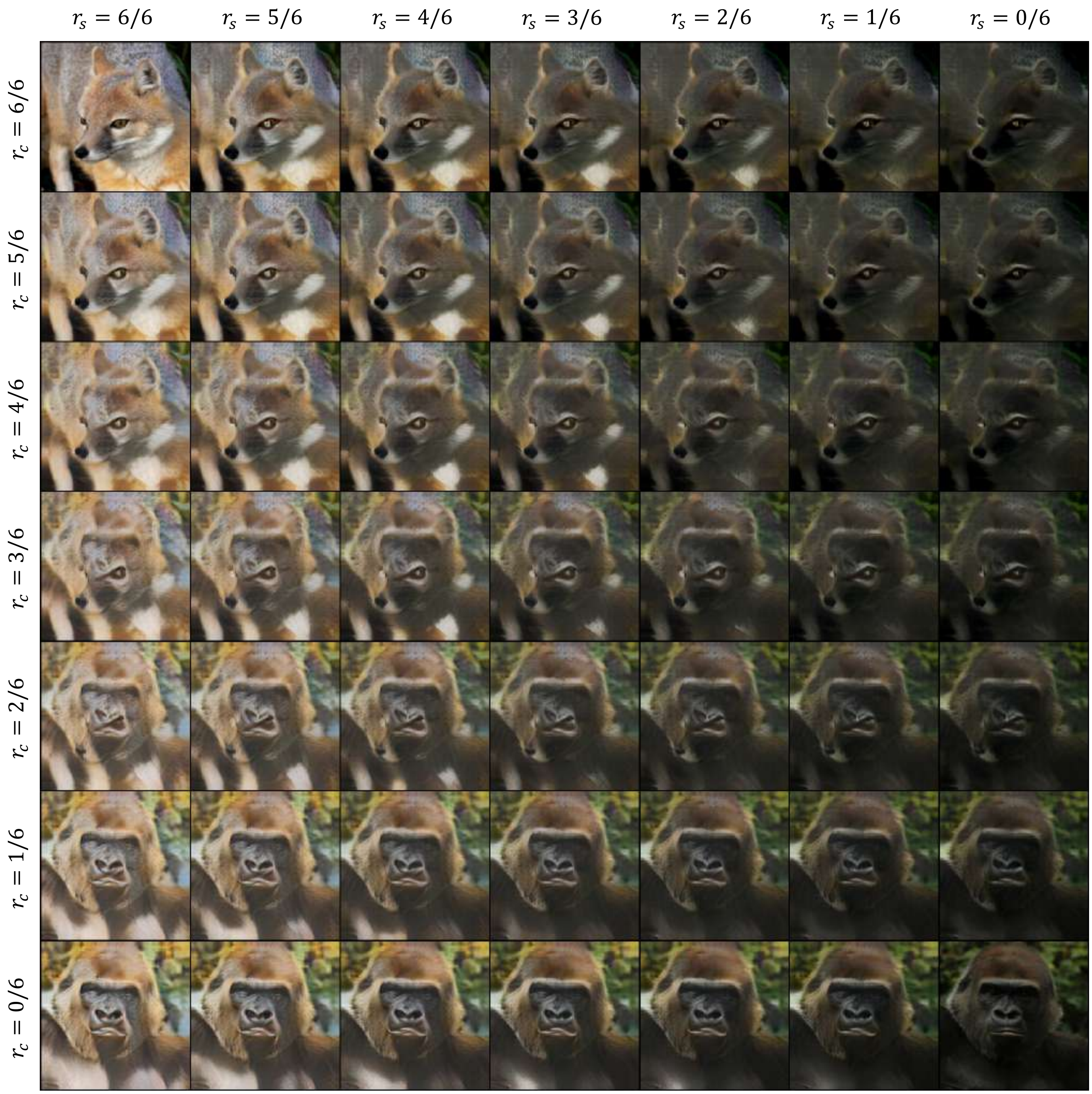

Figure 1: A grid visualization of mixed images created by adjusting the content and style ratios (r_c, r_s) in StyleMix.

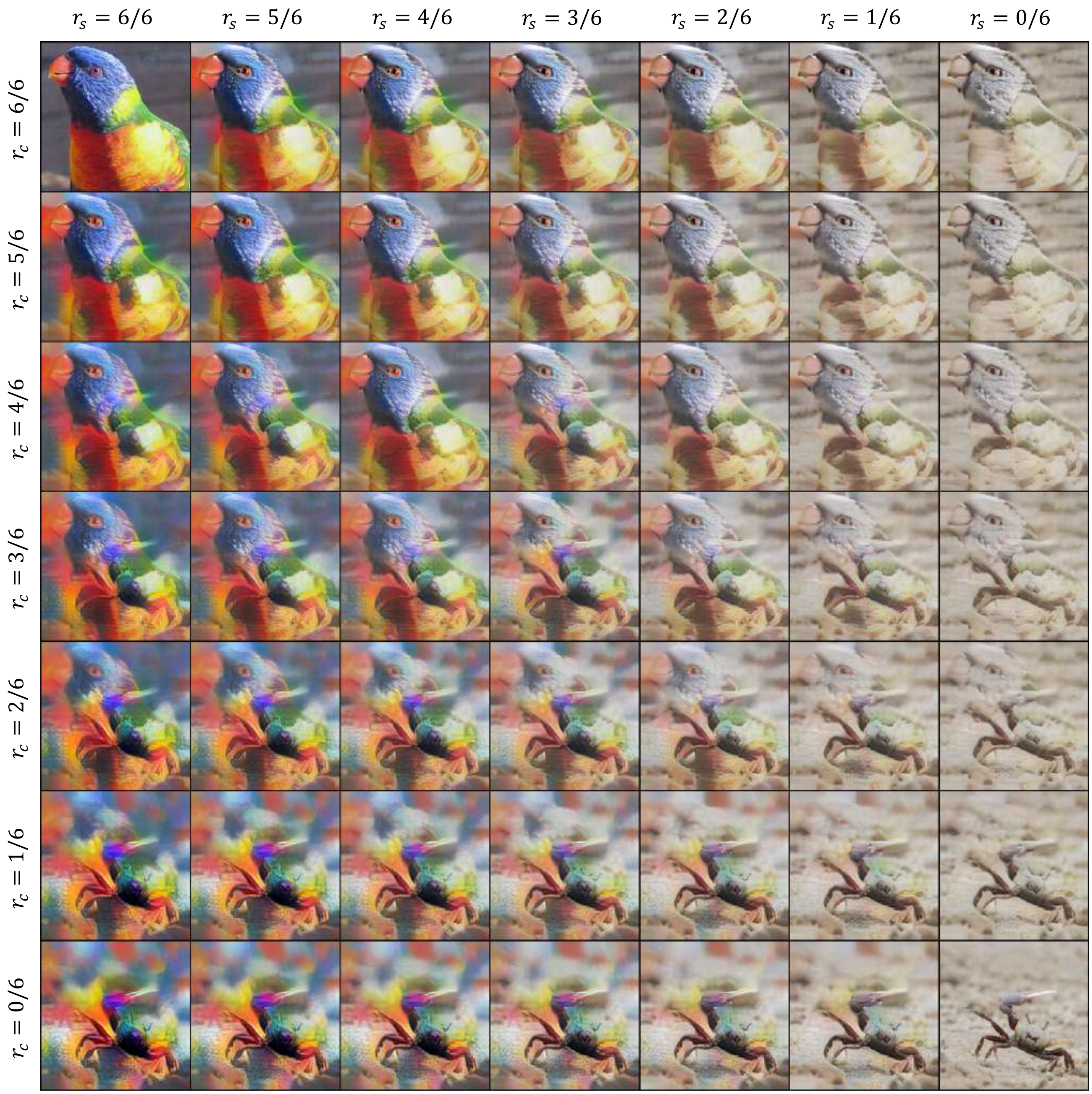

Figure 2: A grid visualization of mixed images created by adjusting the content and style ratios (r_c, r_s) in StyleMix.

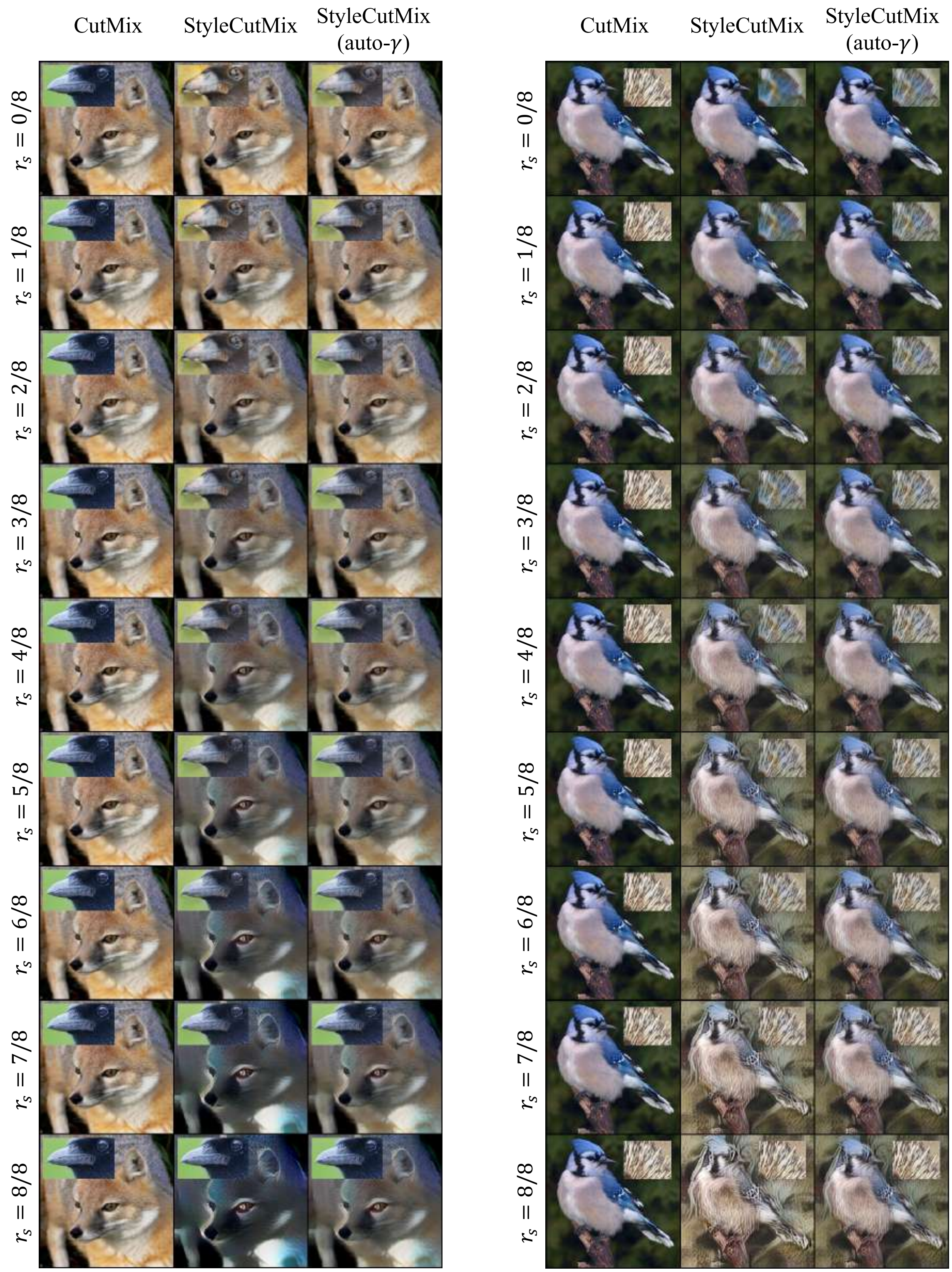

Figure 3: Two sets of grid visualizations of mixed images created by adjusting the style ratios r_s in StyleCutMix and StyleCutMix (auto-γ)