– Supplementary Material –
StyleDiffusion: Controllable Disentangled Style Transfer via Diffusion Models

Zhizhong Wang* Lei Zhao† Wei Xing
College of Computer Science and Technology, Zhejiang University
{endywon, cszhl, wxing}@zju.edu.cn

Contents

A. Societal Impact

Positive Impact. There may be three positive impacts of the proposed method. (1) The proposed method may help the workers engaged in artistic creation or creative design improve the efficiency and quality of their work. (2) The proposed method may inspire researchers in similar fields to design more effective and superior approaches in the future. (3) The proposed method may help the common users obtain more satisfactory creative results.

Negative Impact. The proposed method may be used for generating counterfeit artworks. To mitigate this, further research on identification of generated content is needed.

B. Used Assets

We used the following assets to (1) conduct the comparison experiments [1.-10.] and (2) train the proposed style transfer networks [11.-12.]. To the best of our knowledge, these assets have no ethical concerns.

1. Gatys [8]: https://github.com/leongatys/PytorchNeuralStyleTransfer, MIT License.
2. EFDM [29]: https://github.com/YBZh/EFDM, MIT License.
3. StyTr^2 [3]: https://github.com/diyiiyiii/StyTR-2, No License.
4. ArtFlow [1]: https://github.com/pkuanjie/ArtFlow, No License.
5. AdaAttN [18]: https://github.com/Huage001/AdaAttN, No License.
6. IECAST [2]: https://github.com/HalbertCH/IEContraAST, MIT License.
7. MAST [4]: https://github.com/diyiiyiii/Arbitrary-Style-Transfer-via-Multi-Adaptation-Network, No License.
8. TPFR [26]: https://github.com/nnaisense/conditional-style-transfer, View License in repository.
9. Johnson [14]: https://github.com/abhiskk/fast-neural-style, MIT License.
10. LapStyle [17]: https://github.com/PaddlePaddle/PaddleGAN/blob/develop/docs/en_US/tutorials/lap_style.md, Apache-2.0 License.
11. ADM [5]: https://github.com/openai/guided-diffusion, MIT License.
12. ImageNet [23]: https://image-net.org/, Unknown License.

C. Details of Style Removal Process

As detailed in Algorithm 1, the style removal process consists of two steps. In the first step, we remove the color of the input image I using a color removal operation $\mathcal{R}_{color}$ (*e.g.*, the commonly used ITU-R 601-2 luma transform [9]), obtaining grayscale image I'. In the second step, we use the pre-trained diffusion model ϵ_θ and adopt the deterministic DDIM forward and reverse processes to gradually remove the style information. To accelerate the process without sacrificing much performance, we use fewer discretization steps $\{t_s\}_{s=1}^{S_{for}}$ such that $t_1 = 0$ and $t_{S_{for}} = T_{remov}$. We

*This work was done when Zhizhong Wang was an intern at Huawei.
†Corresponding author.

```
Algorithm 1: Style Removal Process.
  Input: pre-trained model $\epsilon_\theta$, input image $I$, return
         step $T_{remov}$, forward step $S_{for}$, reverse step
         $S_{rev}$, iteration $K_r$
  Output: input image's content $I^c$
  // Remove color
  $I' = \mathcal{R}_{color}(I)$
  // Diffusion-based style removal
  Compute $\{t_s\}_{s=1}^{S_{for}}$ s.t. $t_1 = 0, t_{S_{for}} = T_{remov}$
  $x_0 \leftarrow I$
  for $k = 1 : K_r$ do
      for $s = 1 : S_{for} - 1$ do
          $x_{t_{s+1}} \leftarrow \sqrt{\bar{\alpha}_{t_{s+1}}} f_\theta(x_{t_s}, t_s) + \sqrt{1 - \bar{\alpha}_{t_{s+1}}} \epsilon_\theta(x_{t_s}, t_s)$
      end
      $x_{t_{S_{rev}}} \leftarrow x_{t_{S_{for}}}$
      for $s = S_{rev} : 2$ do
          $x_{t_{s-1}} \leftarrow \sqrt{\bar{\alpha}_{t_{s-1}}} f_{\hat{\theta}}(x_{t_s}, t_s) + \sqrt{1 - \bar{\alpha}_{t_{s-1}}} \epsilon_{\hat{\theta}}(x_{t_s}, t_s)$
      end
  end
  $I^c \leftarrow x_0$
```

set $S_{for} = 40$ for forward process and $S_{rev} = 40$ for reverse process in all experiments. While using larger S_{for} or S_{rev} could reconstruct the high-frequency details better, we found the current setting is enough for our task. *For more details about their effects, we suggest the readers refer to [15].* After K_r iterations (we set $K_r = 5$ for all experiments) of forward and reverse processes, the style characteristics of I' will be dispelled, and thus we obtain the content I^c of the input image.

D. Details of StyleDiffusion Fine-tuning

Similar to [15] and detailed in Algorithm 2, we first precompute the content latents $\{x^{ci}\}_{i=1}^N$ using the deterministic DDIM forward process of the pre-trained diffusion model ϵ_θ. *The precomputed content latents can be stored and reused for fine-tuning other styles.* In our experiments, we fine-tune the diffusion models for all styles using the same precomputed latents of 50 content images sampled from ImageNet [23]. Fine-tuning with more content images may improve the results but also increases the time cost. Thus, we made a trade-off and found the current setting could work well for most cases. To accelerate the process, we use fewer discretization steps $\{t_s\}_{s=1}^{S_{for}}$ such that $t_1 = 0$ and $t_{S_{for}} = T_{trans}$. We set $S_{for} = 40$ for forward process and $S_{rev} = 6$ for reverse process in all experiments. We found $S_{rev} = 6$ is enough to reconstruct clear content structures during style transfer.

```
Algorithm 2: StyleDiffusion Fine-tuning.
  Input: pre-trained model $\epsilon_\theta$, content images'
         contents $\{I_{ci}^c\}_{i=1}^N$, style image's content $I_s^c$,
         style image $I_s$, return step $T_{trans}$, forward
         step $S_{for}$, reverse step $S_{rev}$, fine-tuning
         epoch $K$, style reconstruction iteration $K_s$
  Output: fine-tuned model $\epsilon_{\hat{\theta}}$
  // Precompute content latents
  Compute $\{t_s\}_{s=1}^{S_{for}}$ s.t. $t_1 = 0, t_{S_{for}} = T_{trans}$
  for $i = 1 : N$ do
      $x_0 \leftarrow I_{ci}^c$
      for $s = 1 : S_{for} - 1$ do
          $x_{t_{s+1}} \leftarrow \sqrt{\bar{\alpha}_{t_{s+1}}} f_\theta(x_{t_s}, t_s) + \sqrt{1 - \bar{\alpha}_{t_{s+1}}} \epsilon_\theta(x_{t_s}, t_s)$
      end
      Save the latent $x^{ci} \leftarrow x_{t_{S_{for}}}$
  end
  // Precompute style latent
  Compute $\{t_s\}_{s=1}^{S_{for}}$ s.t. $t_1 = 0, t_{S_{for}} = T_{trans}$
  $x_0 \leftarrow I_s^c$
  for $s = 1 : S_{for} - 1$ do
      $x_{t_{s+1}} \leftarrow \sqrt{\bar{\alpha}_{t_{s+1}}} f_\theta(x_{t_s}, t_s) + \sqrt{1 - \bar{\alpha}_{t_{s+1}}} \epsilon_\theta(x_{t_s}, t_s)$
  end
  Save the latent $x^s \leftarrow x_{t_{S_{for}}}$
  // Fine-tune the diffusion model
  Initialize $\epsilon_{\hat{\theta}} \leftarrow \epsilon_\theta$
  Compute $\{t_s\}_{s=1}^{S_{rev}}$ s.t. $t_1 = 0, t_{S_{rev}} = T_{trans}$
  for $k = 1 : K$ do
      // Optimize the style reconstruction loss
      for $i = 1 : K_s$ do
          $x_{t_{S_{rev}}} \leftarrow x^s$
          for $s = S_{rev} : 2$ do
              $x_{t_{s-1}} \leftarrow \sqrt{\bar{\alpha}_{t_{s-1}}} f_{\hat{\theta}}(x_{t_s}, t_s) + \sqrt{1 - \bar{\alpha}_{t_{s-1}}} \epsilon_{\hat{\theta}}(x_{t_s}, t_s)$
              $I_{ss} \leftarrow f_{\hat{\theta}}(x_{t_s}, t_s)$
              $\mathcal{L} \leftarrow \mathcal{L}_{SR}(I_{ss}, I_s)$
              Take a gradient step on $\nabla_{\hat{\theta}} \mathcal{L}$
          end
      end
      // Optimize the style disentanglement loss
      for $i = 1 : N$ do
          $x_{t_{S_{rev}}} \leftarrow x^{ci}$
          for $s = S_{rev} : 2$ do
              $x_{t_{s-1}} \leftarrow \sqrt{\bar{\alpha}_{t_{s-1}}} f_{\hat{\theta}}(x_{t_s}, t_s) + \sqrt{1 - \bar{\alpha}_{t_{s-1}}} \epsilon_{\hat{\theta}}(x_{t_s}, t_s)$
              $I_{cs} \leftarrow f_{\hat{\theta}}(x_{t_s}, t_s)$
              $\mathcal{L} \leftarrow \mathcal{L}_{SD}(I_{ci}^c, I_{cs}, I_s^c, I_s)$
              Take a gradient step on $\nabla_{\hat{\theta}} \mathcal{L}$
          end
      end
  end
```

In the second step, we precompute the style latent x^s with the same process as above. The style latent will be used to optimize the style reconstruction loss.

In the third step, we copy ϵ_θ to $\epsilon_{\hat{\theta}}$ and start to update $\epsilon_{\hat{\theta}}$ in two substeps. In the first substep, we feed the style latent x^s and generate the stylized image I_{ss} through the deterministic DDIM reverse process. The model is updated under the guidance of the style reconstruction loss $\mathcal{L}_{SR}$. The first substep is repeated K_s times (we set $K_s = 50$ for all experiments) until converged. In the second substep, we feed each content latent in $\{x^{ci}\}_{i=1}^N$ and generate the stylized image through the deterministic DDIM reverse process. The model is updated under the guidance of the style disentanglement loss $\mathcal{L}_{SD}$. At last, we repeat the whole third step K epochs (we set $K = 5$ for all experiments) until converged.

E. Timing and Resource Information

Here, we provide more details on the timing and resource information of our StyleDiffusion using an Nvidia Tesla A100 GPU when stylizing 512×512 size images.

Style Removal. When we use the default setting $(S_{for}, S_{rev}) = (40, 40)$, the forward and reverse processes each takes around 4.921 seconds. Therefore, the whole style removal process takes around $2 \times 4.921 \times 5 = 49.21$ seconds. It requires about 11GB of GPU memory to run at resolution 512×512 pixels.

Fine-tuning. As illustrated in Algorithm 2, the StyleDiffusion fine-tuning process consists of a latent precomputing stage and a model updating stage. The latent precomputing stage is carried out just once and can be reused for fine-tuning other styles. When we use $S_{for} = 40$ as default, the forward process takes around 4.921 seconds. Therefore, when we precompute the latents from 50 images, it takes around $50 \times 4.921 = 246.05$ seconds and requires about 11GB GPU memory. For the model updating stage, when the batch size is 1 and $S_{rev} = 6$, the first substep (optimizing the style reconstruction loss $\mathcal{L}_{SR}$) takes around 2.092 seconds for each repeat, and the second substep (optimizing the style disentanglement loss $\mathcal{L}_{SD}$) takes around 3.351 seconds for each content latent. Therefore, one epoch with 50 repeated first substep and 50 precomputed content latents for the second substep takes around $50 \times 2.092 + 50 \times 3.351 = 272.15$ seconds. When we fine-tune the model with 5 epochs, it takes around 23 minutes in total. The fine-tuning process requires about 26GB of GPU

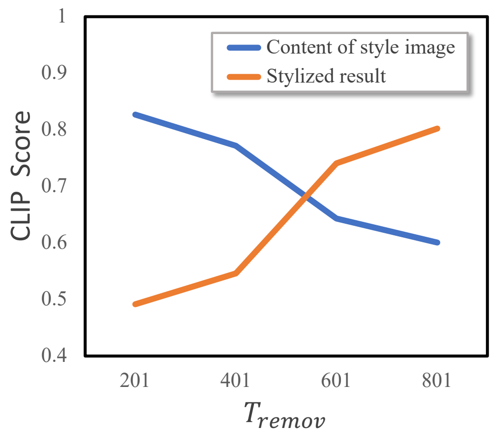


Figure 1. **C-S disentanglement of style image** achieved by adjusting the return step T_{remov} of the *style removal module*. CLIP score (averaged on 384 image pairs) measures the style similarity with the style image. When more style information is removed (blue line), it will be transferred to the stylized result (orange line).

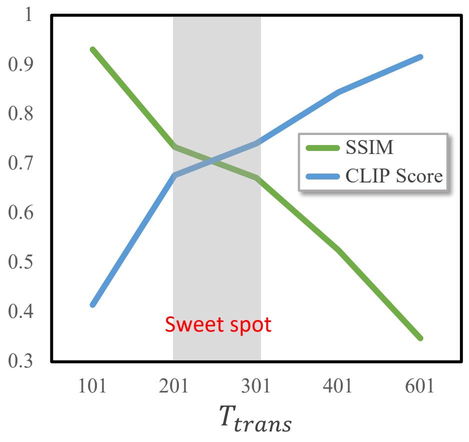


Figure 2. **C-S trade-off** achieved by adjusting the return step T_{trans} of the *style transfer module* at the **training** stage while fixing $T_{trans} = 301$ at the testing stage. SSIM and CLIP score (averaged on 384 image pairs) measure the content similarity and the style similarity, respectively.

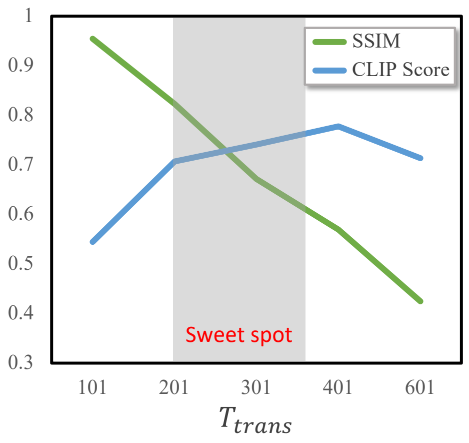
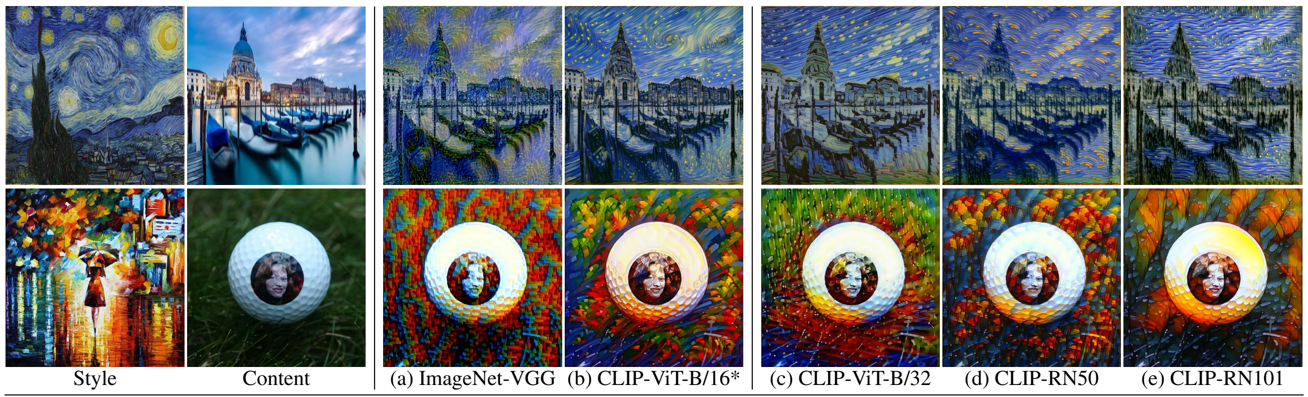


Figure 3. **C-S trade-off** achieved by adjusting the return step T_{trans} of the *style transfer module* at the **testing** stage while fixing $T_{trans} = 301$ at the training stage. SSIM and CLIP score (averaged on 384 image pairs) measure the content similarity and the style similarity, respectively.



SSIM / CLIP Score:	(a) ImageNet-VGG	(b) CLIP-ViT-B/16*	(c) CLIP-ViT-B/32	(d) CLIP-RN50	(e) CLIP-RN101
	0.541 / 0.602	0.672 / 0.741	0.591 / 0.705	0.529 / 0.677	0.582 / 0.719

Figure 4. **Ablation study** on different **disentanglement space** (VGG vs. CLIP, columns (a-b)) and **CLIP image encoders** (columns (b-e)). * denotes our default setting. Zoom-in for better comparison.

memory.

Inference. When we use the default setting $(S_{for}, S_{rev}) = (40, 6)$, the forward process takes around 4.921 seconds and the reverse process takes around 0.691 seconds. Therefore, the total inference time is $4.921 + 0.691 = 5.612$ seconds. The inference process requires about 13GB of GPU memory.

Currently, *we have not optimized the model size and GPU memory consumption here*. We believe there is substantial room for improvement, and we would like to elaborate on that in future work.

F. More Ablation Study and Analyses

Quantitative Analyses of C-S Disentanglement. Here, we provide more quantitative results to analyze the C-S Disentanglement achieved by our StyleDiffusion. As shown in Fig. 1, we observe that the style is well disentangled from the content in the style image by adjusting the return step T_{remov} of the style removal module. As such, when more style information is removed (blue line), it will be transferred to the corresponding stylized result (orange line). The quantitative analyses are consistent with the qualitative results displayed in Fig. 4 of our main paper.

Quantitative Analyses of C-S Trade-off. We also provide more quantitative results to analyze the C-S trade-off achieved by our StyleDiffusion. As shown in Fig. 2 and Fig. 3, we can flexibly control the C-S trade-off at both the training stage (Fig. 2) and the testing stage (Fig. 3) by adjusting the return step T_{trans} of diffusion models. The sweet spot areas are highlighted in the figures, which are the most probable for obtaining satisfactory results. Overall, the quantitative analyses are consistent with the qualitative results displayed in Fig. 5 of our main paper.

CLIP Space vs. VGG Space. As discussed in our main paper, we leverage the open-domain CLIP [22] space to formulate the style disentanglement. The pre-trained CLIP space integrates rich cross-domain image (and supplementarily, text) knowledge and thus can measure the "style distance" more accurately. As shown in Fig. 4 (a-b), we compare it with the ImageNet [23] pre-trained VGG-19 [24], which has been widely adopted in prior arts [8, 13] to extract the style information. As is evident, using the CLIP space recovers the style information more sufficiently and realistically, significantly outperforming the VGG space (which is also validated by the bottom quantitative scores). It may be attributed to the fact that the VGG is pre-trained on ImageNet and therefore lacks a sufficient understanding of artistic styles. In contrast, the CLIP space encapsulates a myriad of knowledge of not only the photograph domain but also the artistic domain, which is more powerful in depicting the style of an image. Besides, it is worth noting that the CLIP space naturally provides multi-modal compatibility, which can facilitate users to control the style transfer with multi-modal signals, *e.g.*, image and text (see later Sec. G).

Different CLIP Image Encoders. We also investigate the effects of different CLIP [22] image encoders to conduct the style disentanglement. As shown in Fig. 4 (b-e), in general, ViTs [6] achieve better visual results than ResNets (RN) [10], *e.g.*, the brushstrokes are more natural in the top row, and the colors are more vivid in the bottom row. And ViT-B/16 performs better than ViT-B/32 in capturing more fine-grained styles. Interestingly, our findings coincide with the reported performance of these image encoders on high-level vision tasks (*e.g.*, classification) in the original CLIP paper [22]. It indicates that our stylization performance is closely related to the high-level semantic representations learned by the image encoder, which also gives evidence to the correlations between high-level vision tasks and low-level vision tasks.

Diffusion-based Style Removal vs. AE-based Style Removal. To demonstrate the superiority of diffusion-based style removal, we compare it with a possible alter-

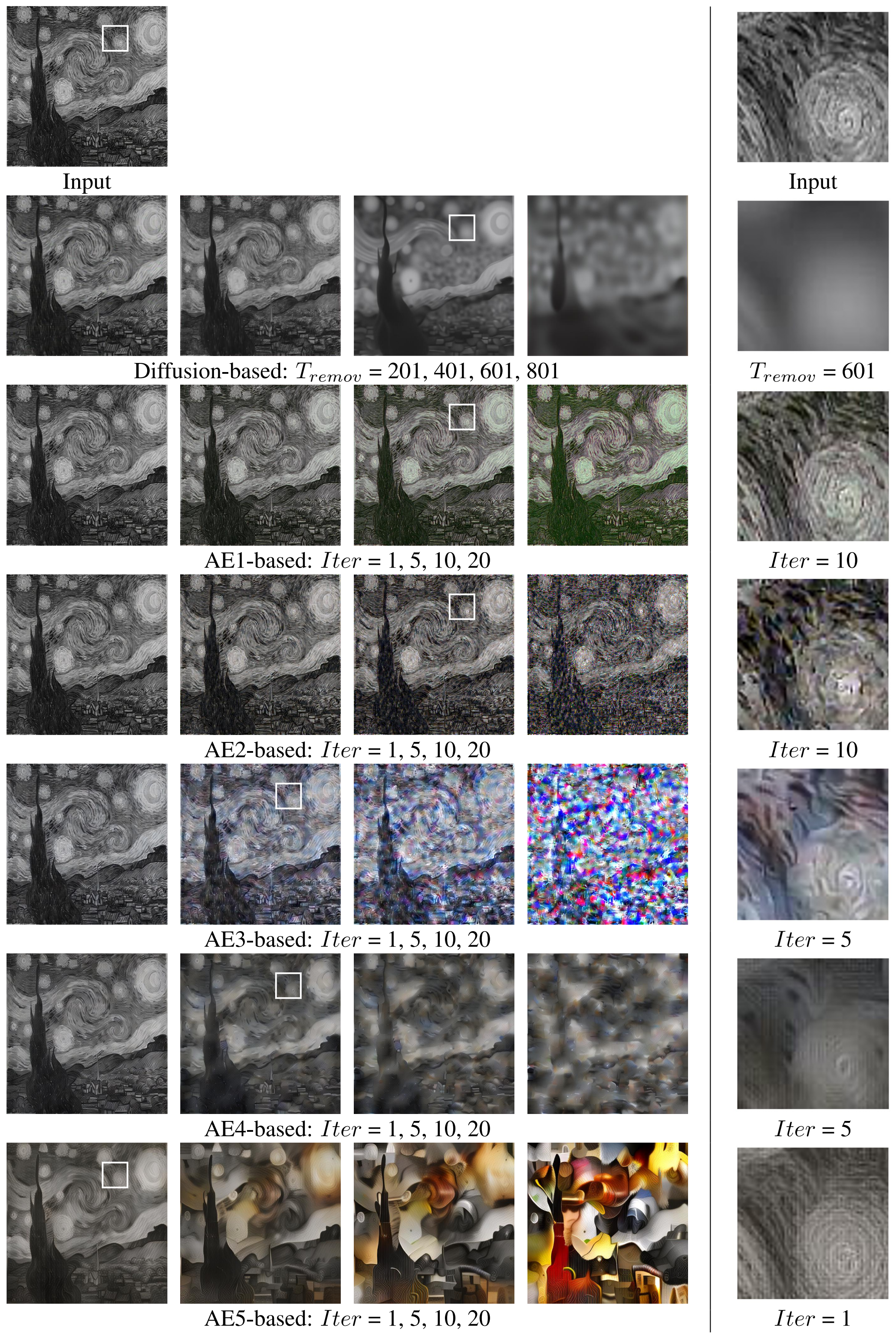

Input
Input
Diffusion-based: T_{remov} = 201, 401, 601, 801
T_{remov} = 601
AE1-based: $Iter$ = 1, 5, 10, 20
$Iter$ = 10
AE2-based: $Iter$ = 1, 5, 10, 20
$Iter$ = 10
AE3-based: $Iter$ = 1, 5, 10, 20
$Iter$ = 5
AE4-based: $Iter$ = 1, 5, 10, 20
$Iter$ = 5
AE5-based: $Iter$ = 1, 5, 10, 20
$Iter$ = 1

Figure 5. **Diffusion-based style removal vs. AE-based style removal.** The last column shows the enlarged areas of the corresponding best style removed results manually selected in each row. As can be observed, diffusion-based style removal can better remove the detailed style of the style image while preserving the main content structures. In contrast, AE-based style removal cannot plausibly remove the detailed style and often introduces color noises/artifacts and destroys the content structures.

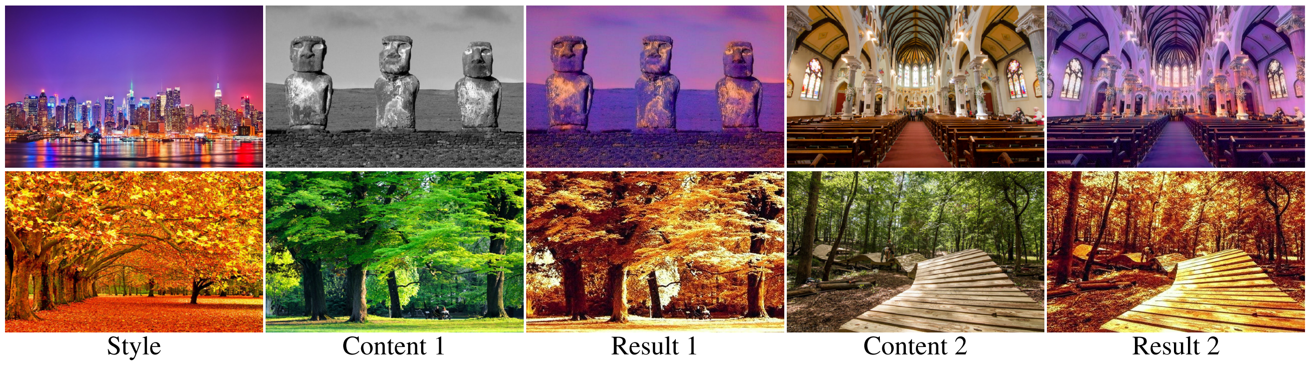


Figure 6. **Photo-realistic style transfer** achieved by our StyleDiffusion. We set $T_{remov} = 401$ and $T_{trans} = 101$ for this task.

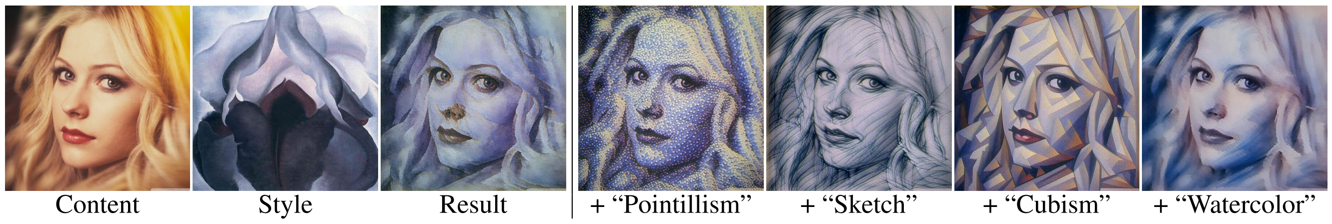


Figure 7. **Multi-modal style manipulation.** Our framework is compatible with image and text modulation signals, which provides users with a more flexible way to manipulate the style of images.

native, *i.e.*, Auto-Encoders (AEs), since one may argue that the diffusion model is a special kind of (Variational) Auto-Encoder network [21]. We directly use the AEs released by Li *et al*. [16], which employ the VGG-19 network [24] as the encoders, fix them and train decoder networks for inverting VGG features to the original images. They select feature maps at five layers of the VGG-19, i.e., Relu_X_1 (X=1,2,3,4,5), and train five decoders accordingly, which we denote as AEX (X=1,2,3,4,5) in the following. When used for style removal, we iteratively perform the encoding and decoding processes of AEs for the input images. The comparison results are shown in Fig. 5. As can be observed in the bottom five rows, AE-based style removal cannot plausibly remove the detailed style and often introduces color noises/artifacts and destroys the content structures, which is undesirable for style removal. By contrast, diffusion-based style removal can smoothly remove the style details while preserving the main content structures, significantly outperforming AE-based style removal.

G. Extensions

Photo-realistic Style Transfer. Our StyleDiffusion successfully separates style from content in a controllable manner. Thus, it can easily achieve photo-realistic style transfer [20] by adjusting the content extraction of the style removal module. Specifically, since the style of a photo is mainly reflected by the low-level and high-frequency features such as colors and brightness, we reduce T_{remov} to a relatively smaller value, *e.g.*, 401. Moreover, to better preserve the content structures, we adjust the style transfer process and reduce T_{trans} to 101. We show some photo-realistic style transfer results synthesized by our StyleDiffusion in Fig. 6.

Multi-modal Style Manipulation. As our framework leverages the open-domain CLIP [22] space to measure the "style distance", it is naturally compatible with image and text modulation signals. By adding a directional CLIP loss term [7, 15] to our total loss, our framework can easily achieve multi-modal style manipulation, as shown in Fig. 7. *As far as we know, our framework is the first unified framework to achieve both image and text guided style transfer.*

Diversified Style Transfer. In the fine-tuning, our style transfer module adopts the deterministic DDIM [25] forward and reverse processes (Eq. (8) and Eq. (9) in the main paper). However, during inference, we can directly replace the deterministic DDIM forward process with the stochastic DDPM [12] forward process (Eq. (2) in the main paper) to achieve diversified style transfer [27], as shown in Fig. 8. The users can easily trade off the diversity and quality by adjusting the return step or iteration of the DDPM forward process. The diverse results can give users endless choices to obtain more satisfactory results [27, 28].

H. More Comparison Results

In Fig. 12 and 13, we provide more qualitative comparison results with state-of-the-art style transfer methods.

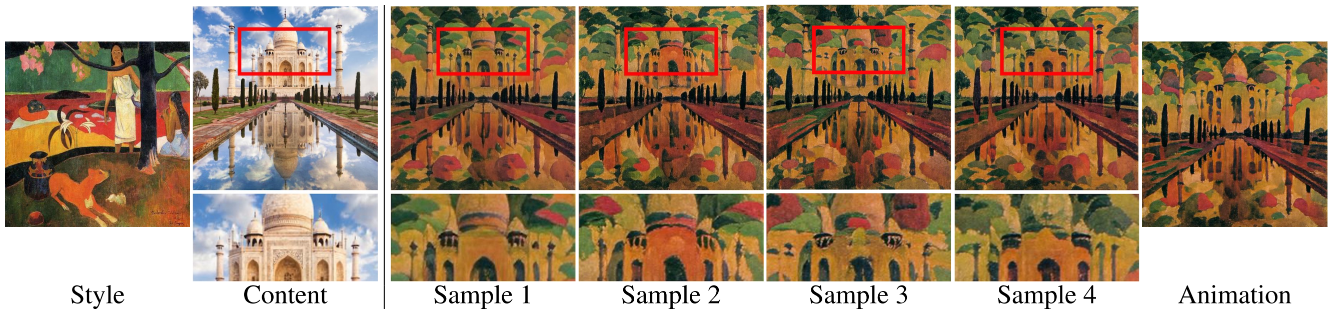


Figure 8. **Diversified style transfer.** Our framework can easily achieve diversified style transfer during inference by directly adopting the stochastic DDPM [12] forward process. Click on the last image to see animation using Adobe Reader.

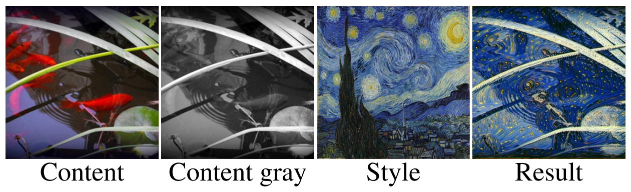


Figure 9. **Failure case of type 1: vanishing of salient content.** Some results generated by our method may vanish the salient content of the content image, *e.g*., the red carps.

I. Additional Stylized Results

In Fig. 14 and 15, we provide additional stylized results synthesized by our proposed StyleDiffusion.

J. Limitation and Discussion

Except for the limitations we have discussed in the main paper, here we provide some failure cases and analyze the reasons behind them. Further, we also discuss the possible solutions to address them, which may help inspire future improvements to our framework.

Vanishing of Salient Content. Some of our generated results may vanish the salient content of the content image, *e.g*., the red carps in Fig. 9. It can be attributed to the color removal operation used in our style removal module. The commonly used ITU-R 601-2 luma transform [9] may not well preserve the original RGB image's color contrast and color importance, as shown in column 2 of Fig. 9. We adopt it here mainly for its simplicity and fast speed. This problem may be addressed by using more advanced contrast-preserving decolorization techniques, like [19].

Biased Color Distribution. As shown in Fig. 10, though our method learns the challenging pointillism style well, the color distribution seems to stray from that of the style image. This problem can be alleviated by increasing the style reconstruction iteration K_s (see Algorithm 2) to inject more style prior, but the training time also increases significantly. One may consider borrowing some ideas from existing color transfer approaches [11] to address this problem.

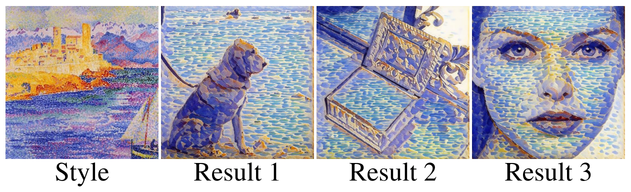


Figure 10. **Failure case of type 2: biased color distribution.** Our method may generate results that deviate from the color distribution of the style image.

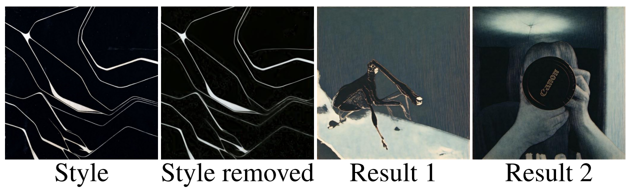


Figure 11. **Failure case of type 3: inseparable content and style.** Our method is hard to transfer plausible style for style images with inseparable content and style. The second column shows the style removed result of the style image.

Inseparable Content and Style. Our method is hard to achieve plausible style transfer for style images with inseparable content and style, *e.g*., the simple line art shown in Fig. 11. Since the content of line art is also its style, our framework is hard to separate them properly, as shown in column 2 of Fig. 11. One possible solution is to treat line art as the style only and increase the return step T_{remov} of the style removal module to dispel as much style information as possible, or increase the return step T_{trans} of the style transfer module to learn as sufficient line art style as possible.

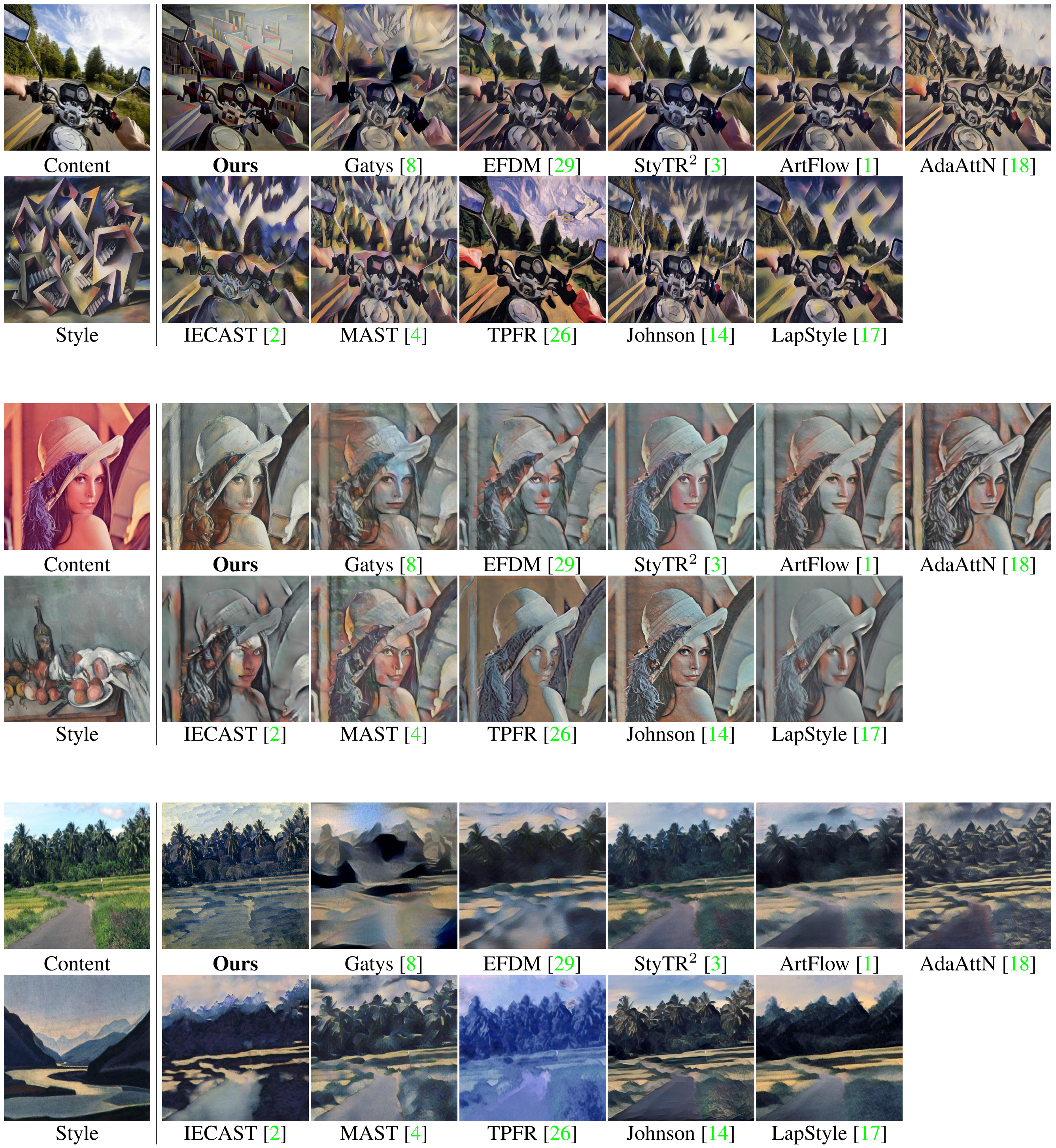


Figure 12. **More qualitative comparison results (set 1)** with state of the art. Zoom-in for better comparison.

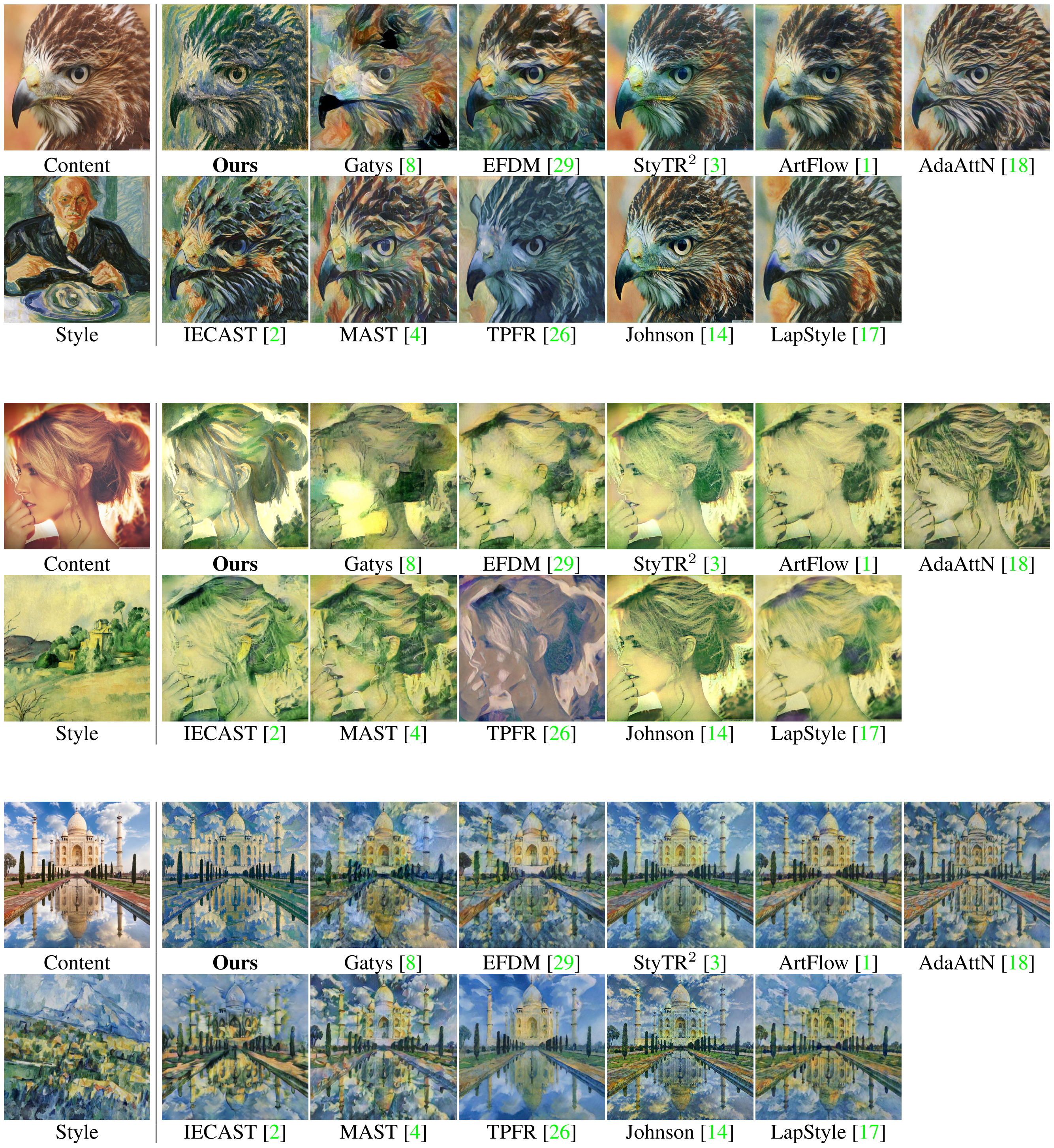


Figure 13. **More qualitative comparison results (set 2)** with state of the art. Zoom-in for better comparison.

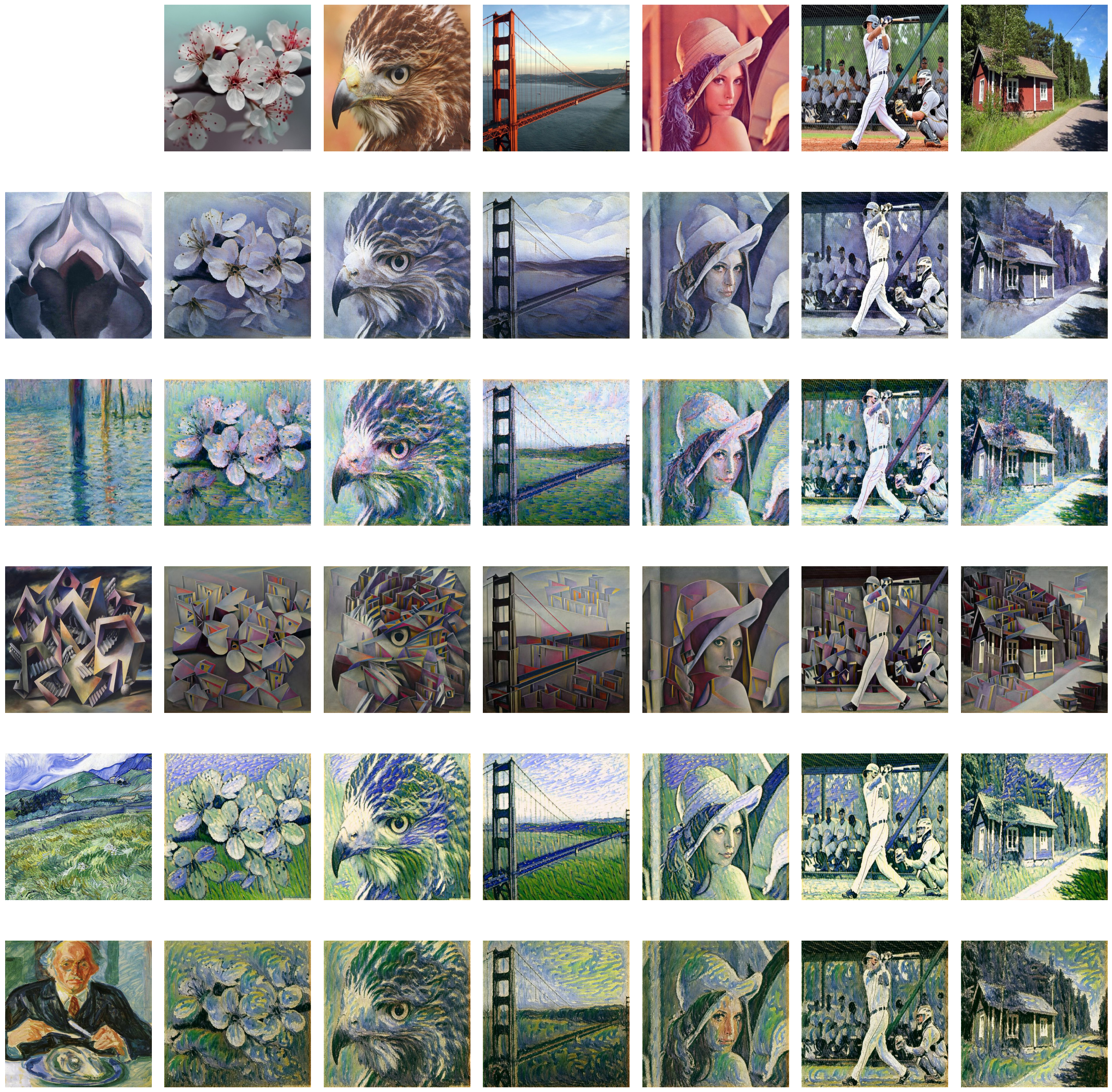


Figure 14. **Additional stylized results (set 1)** synthesized by our proposed StyleDiffusion. The first row shows content images and the first column shows style images.

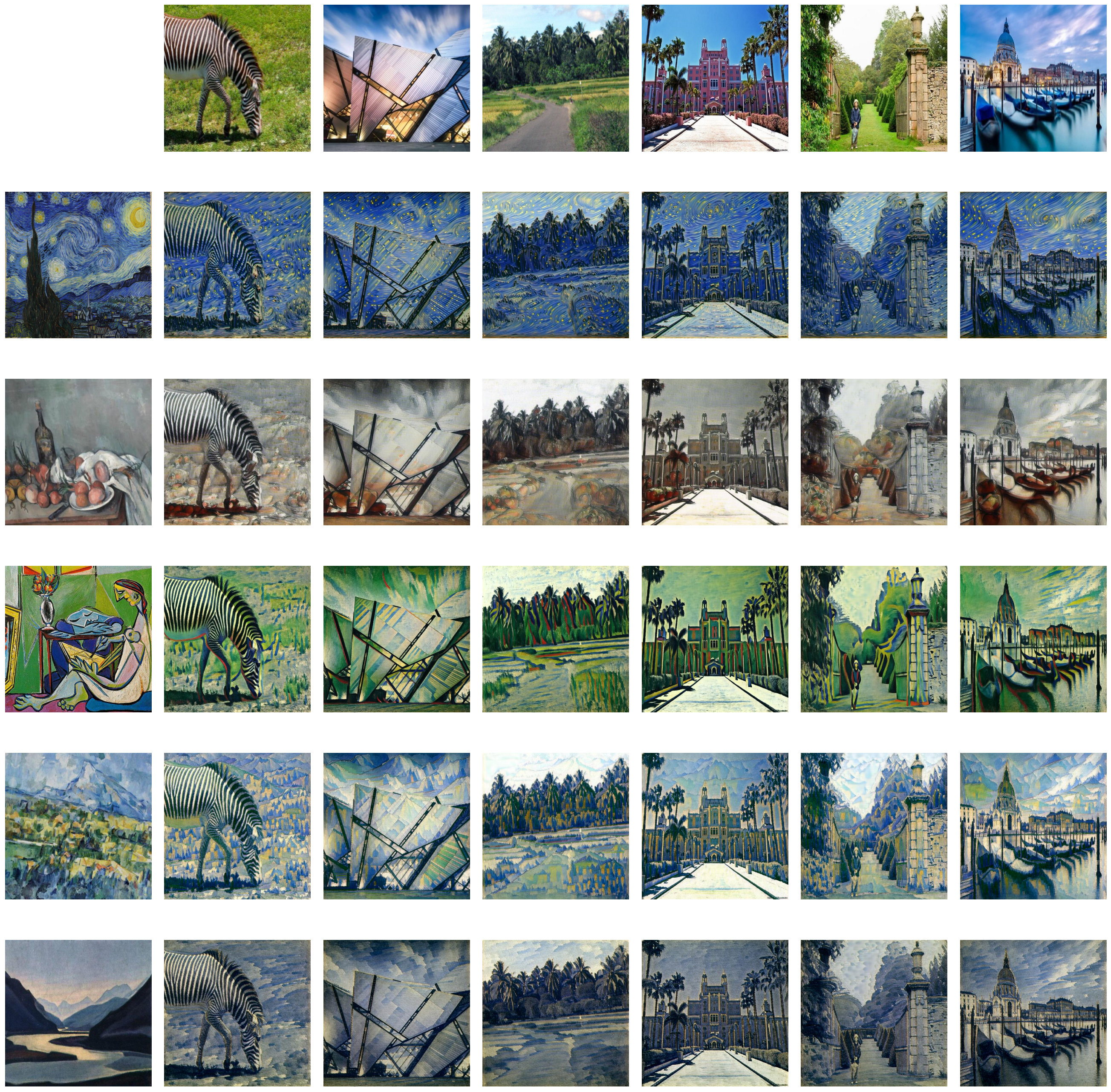


Figure 15. **Additional stylized results (set 2)** synthesized by our proposed StyleDiffusion. The first row shows content images and the first column shows style images.

References

[1] Jie An, Siyu Huang, Yibing Song, Dejing Dou, Wei Liu, and Jiebo Luo. Artflow: Unbiased image style transfer via reversible neural flows. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 862–871, 2021. 1, 8, 9

[2] Haibo Chen, Lei Zhao, Zhizhong Wang, Huiming Zhang, Zhiwen Zuo, Ailin Li, Wei Xing, and Dongming Lu. Artistic style transfer with internal-external learning and contrastive learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:26561–26573, 2021. 1, 8, 9

[3] Yingying Deng, Fan Tang, Weiming Dong, Chongyang Ma, Xingjia Pan, Lei Wang, and Changsheng Xu. Stytr2: Image style transfer with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11326–11336, 2022. 1, 8, 9

[4] Yingying Deng, Fan Tang, Weiming Dong, Wen Sun, Feiyue Huang, and Changsheng Xu. Arbitrary style transfer via multi-adaptation network. In *Proceedings of the 28th ACM International Conference on Multimedia (ACM MM)*, pages 2719–2727, 2020. 1, 8, 9

[5] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:8780–8794, 2021. 1

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2020. 4

[7] Rinon Gal, Or Patashnik, Haggai Maron, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. Stylegan-nada: Clip-guided domain adaptation of image generators. *ACM Transactions on Graphics (TOG)*, 41(4):1–13, 2022. 6

[8] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423, 2016. 1, 4, 8, 9

[9] Rafael C Gonzalez. *Digital image processing*. Pearson education india, 2009. 1, 7

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 4

[11] Mingming He, Dongdong Chen, Jing Liao, Pedro V Sander, and Lu Yuan. Deep exemplar-based colorization. *ACM Transactions on Graphics (TOG)*, 37(4):1–16, 2018. 7

[12] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:6840–6851, 2020. 6, 7

[13] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1501–1510, 2017. 4

[14] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 694–711. Springer, 2016. 1, 8, 9

[15] Gwanghyun Kim, Taesung Kwon, and Jong Chul Ye. Diffusionclip: Text-guided diffusion models for robust image manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2426–2435, 2022. 2, 6

[16] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 386–396, 2017. 6

[17] Tianwei Lin, Zhuoqi Ma, Fu Li, Dongliang He, Xin Li, Errui Ding, Nannan Wang, Jie Li, and Xinbo Gao. Drafting and revision: Laplacian pyramid network for fast high-quality artistic style transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5141–5150, 2021. 1, 8, 9

[18] Songhua Liu, Tianwei Lin, Dongliang He, Fu Li, Meiling Wang, Xin Li, Zhengxing Sun, Qian Li, and Errui Ding. Adaattn: Revisit attention mechanism in arbitrary neural style transfer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6649–6658, 2021. 1, 8, 9

[19] Cewu Lu, Li Xu, and Jiaya Jia. Contrast preserving decolorization with perception-based quality metrics. *International Journal of Computer Vision (IJCV)*, 110(2):222–239, 2014. 7

[20] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. Deep photo style transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4990–4998, 2017. 6

[21] Calvin Luo. Understanding diffusion models: A unified perspective. *arXiv preprint arXiv:2208.11970*, 2022. 6

[22] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, pages 8748–8763. PMLR, 2021. 4, 6

[23] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 1, 2, 4

[24] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 4, 6

[25] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations (ICLR)*, 2020. 6

[26] Jan Svoboda, Asha Anoosheh, Christian Osendorfer, and Jonathan Masci. Two-stage peer-regularized feature recombination for arbitrary image style transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13816–13825, 2020. 1, 8, 9

[27] Zhizhong Wang, Lei Zhao, Haibo Chen, Lihong Qiu, Qihang Mo, Sihuan Lin, Wei Xing, and Dongming Lu. Diversified

arbitrary style transfer via deep feature perturbation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7789–7798, 2020. 6

[28] Zhizhong Wang, Lei Zhao, Haibo Chen, Zhiwen Zuo, Ailin Li, Wei Xing, and Dongming Lu. Divswapper: Towards diversified patch-based arbitrary style transfer. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4980–4987, 2022. 6

[29] Yabin Zhang, Minghan Li, Ruihuang Li, Kui Jia, and Lei Zhang. Exact feature distribution matching for arbitrary style transfer and domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8035–8045, 2022. 1, 8, 9