StyleGAN-∞: Extending StyleGAN to Arbitrary-Ratio Translation with StyleBook – Supplementary Material –

Yihua Dai, Tianyi Xiang, Bailin Deng, Yong Du, Hongmin Cai, Jing Qin, and Shengfeng He, *Senior Member, IEEE,*

In the following, we provide more implementation details of our StyleGAN-∞, and the dataset processing details for multi-human generation task in Sec. 1. In Sec. 2, we show more evaluation results using sketches and other conditions, as well as results in other domains.

1 Implementation Details

1.1 Dataset Processing Details

To generate blurry backgrounds in our dataset, we use a Gaussian function with the kernel size of 25×25. The corresponding standard deviation along the X and Y axis are both set to 0.

1.2 Training Implementation Details

For the training, we use Adam optimizer with a learning rate 10^{-4}, $b_1 = 0.5$ and $b_2 = 0.9$ for training. The hyper-parameters in Eq. 9 are set to $\lambda_m = 1.0$, $\lambda_{adv} = 0.8$ and $\lambda_p = 0.8$. We implement all experiment using Pytorch and run them on a single Nvidia GeForce RTX 3090.

For distilling features of StyleGAN [1], [2] to learn the StyleBook in Stage I under different domains, we use the pre-trained StyleGANv2 [2] models which are trained on FFHQ [2] (portrait) and LSUN [3] (church, horse and cat) datasets. The weight of StyleGAN is fixed during the whole training process. Meanwhile, same as sketch-to-portrait, we apply the edge extractor [4] to obtain the corresponding sketch for other domains. The hyper-parameters of loss functions for training in non-human domains are the same as in the multi-human translation task.

The work is supported by the Guangdong Natural Science Funds for Distinguished Young Scholar (No. 2023B1515020097), the National Research Foundation Singapore under the AI Singapore Programme (No: AISG3-GV-2023-011), the Innovation and Technology Fund under Guangdong-Hong Kong Technology Cooperation Funding Scheme (ITF-TCFS) (project no. GHP/051/20GD), and the Lee Kong Chian Fellowships. (Yihua Dai and Tianyi Xiang contributed equally to this paper.) (Corresponding author: Shengfeng He.)

Yihua Dai, Tianyi Xiang, and Hongmin Cai are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China; Yihua Dai and Tianyi Xiang are also with the School of Computing and Information Systems, Singapore Management University, Singapore. E-mail: yihuadd20@gmail.com, xty435768@gmail.com, hmcai@scut.edu.cn.

Bailin Deng is with the School of Computer Science and Informatics, Cardiff University, UK. E-mail: DengB3@cardiff.ac.uk.

Yong Du is with the Faculty of Information Science and Engineering, Ocean University of China, Qingdao, China. E-mail: csyongdu@ouc.edu.cn.

Jing Qin is with the School of Nursing, the Hong Kong Polytechnic University, Hong Kong, China. E-mail: harry.qin@polyu.edu.hk.

Shengfeng He is with the School of Computing and Information Systems, Singapore Management University, Singapore. E-mail: shengfenghe@smu.edu.sg.

1.3 Model Architecture

In this section, we present more detailed information about the architecture of our proposed StyleGAN-∞ model, including more details about the Style Injection module, Encoder, Decoder, Dual Embedding Module, Transformer, and Discriminator.

To enhance the clarity in illustrating the structure of our network model, we present a structured Tab. 2 detailing the relevant modules utilized, including SE Module, ResnetBlock, AttnBlock, DownBlock and StyleInjector. Additionally, comprehensive information is provided regarding the network structure and associated parameter in Tab. 3 for key modules including the Style Injection Module, Encoder, Decoder, and Dual Embedding Module. Within this framework, variables such as H and W denote the length and width of the input condition image, while c represents the number of channels in the input condition image. Furthermore, $StyleF_x$ represents the features extracted from the StyleGAN layer (top to bottom), and F_x denotes the output of the x-th layer of the Style Injection Module.

Transformer Architecture. In the second stage, Our transformer model architecture adopts the structure of GPT2 [5], and the corresponding hyperparameters are changed for our method architecture as shown below. We set the number of Transformer blocks n_{layer} = 24, the number of attention heads in the Transformer n_h = 16, the embedding of Stylebook entries n_{embed} = 1024, the dimensionality of Stylebook entries n_z = 256, the number of Stylebook entries Z = 1024, the dropout rate to 0.0, the length of the input sequence to 512. Additionally, we set the temperature t to 1.0 and the cutoff value k to 100 for top-k random selection in StyleBook.

Discriminator Architecture. For the discriminator, we use a patch-based model as in [6].

TABLE 1: Quantitative comparison with state-of-the-art methods pretrained on FFHQ of multi-human translation using different conditions on the corresponding test set.

Conditions	Methods	FID↓	LPIPS↓	IS↑
Segmentation	CycleGAN [7]	169.76	0.538	1.287
	Pix2PixHD [8]	95.86	0.449	1.402
	VQGAN [9]	94.32	0.463	1.469
	BiCycleGAN [10]	124.06	0.471	1.372
	TSIT [11]	116.13	0.437	1.413
	SPADE [12]	124.06	0.470	1.372
	Ours	**73.89**	**0.406**	**1.602**
Segmentation + Sketch	CycleGAN [7]	126.59	0.495	1.454
	Pix2PixHD [8]	77.63	0.363	1.528
	VQGAN [9]	78.09	0.351	1.621
	BiCycleGAN [10]	120.39	0.479	1.563
	TSIT [11]	97.31	0.361	1.495
	SPADE [12]	97.88	0.436	1.397
	Ours	**61.87**	**0.272**	**1.782**

Modules
SE Module(C_1)
AdaptiveAvgPool2d(1) + Conv2d(1 × 1, C_1, C_{hidden}, stride=1) + ReLU() + Conv2d(1 × 1, C_{hidden}, C_1, stride=1) + Sigmoid()
ResnetBlock(C_{in}, C_{out}, dropout)
GroupNorm(32, C_{in}) + Conv2d(3 × 3, C_{in}, C_{out}, stride=1, padding=1) + GroupNorm(32, C_{out}) + Dropout(dropout) + Conv2d(3 × 3, C_{out}, C_{out}, stride=1 padding=1)
AttnBlock(C_{in})
GroupNorm(32, C_{in}) + (q): Conv2d(1 × 1, C_{in}, C_{in}, stride=1, padding=0) + (k): Conv2d(1 × 1, C_{in}, C_{in}, stride=1, padding=0) + (v): Conv2d(1 × 1, C_{in}, C_{in}, stride=1, padding=0) + Conv2d(1 × 1, C_{in}, C_{in}, stride=1, padding=0)
DownBlock(C_{in}, C_{out}, dropout=0.0)
ResnetBlock(C_{in}, C_{out}, dropout=dropout) + ResnetBlock(C_{out}, C_{out}, dropout=dropout) + Conv2d(3 × 3, C_{out}, C_{out}, stride=2, padding=0)
StyleInjector(f_1, f_2, C_{in}, C_{out}, C_2, dropout)
Concat(f_1, f_2) + SEModule(C_{in}, C_2) + ResnetBlock(C_{in}, C_{out}, dropout)

TABLE 2: The architecture of the relevant modules of StyleGAN-∞ model. The input and output channels' size of each module are denoted as C_{in} and C_{out}, respectively. C_1 and C_2 are the input channels and middle channel number division ratio of channels of the SE module. D is the dropout rate. The detailed structure of each module is also provided.

2 More Evaluation Results

2.1 Comparisons with State-of-the-art Methods

In Fig. 1, we present more quantitative results compared with the baselines (CycleGAN [7], Pix2PixHD [8], VQGAN [9], BiCycleGAN [10], TSIT [11], SPADE [12], and ControlNet [14]) in the multi-human sketch-to-image translation task. Our method can preserve richer details and produce better overall image quality during image generation.

2.2 Other Conditions for Multi-human Synthesis

We show more evaluations of our model in multi-human image generation guided by three other types of conditions, including low-resolution images, segmentation maps, and compound conditions (segmentation and sketch). For segmentation maps and compound conditions, we first build two corresponding test sets, each consisting of 216 images with different ratios/resolutions, then we present the quantitative (in Tab. 1) and qualitative (in Fig. 2) comparison with the baselines. We do not compare our method with ControlNet [14] in this study because there are no well-pretrained weights available for these specific conditions. Our model is able to adapt well to both segmentation maps and compound conditions, achieving the best numerical performance and generating corresponding high-quality images compared to our baselines. Since the structure of the model needs to be modified accordingly to make it adaptable to low-resolution images as conditions, to ensure fairness we do not compare our model with baselines when using limited pixels as conditions for multi-human image translation. Instead, we only present these results in Fig. 3, fully demonstrating the excellent performance of our model in super-resolution tasks.

2.3 Results in Different Domains

We also extend our model to other non-human domains including Church (Fig. 4 and Fig. 5), Cat (Fig. 6), and Horse (Fig. 7 and Fig. 8). Our model can still produce high-quality and realistic translation results of arbitrary ratio/resolution in different domains. This further demonstrates the powerful ability of our method on general image translation tasks.

References

[1] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *CVPR*, 2019, pp. 4401–4410.

[2] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of stylegan," in *CVPR*, 2020, pp. 8110–8119.

[3] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao, "Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop," *arXiv preprint arXiv:1506.03365*, 2015.

[4] E. Simo-Serra, S. Iizuka, K. Sasaki, and H. Ishikawa, "Learning to simplify: fully convolutional networks for rough sketch cleanup," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, pp. 1–11, 2016.

[5] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[6] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *CVPR*, 2017, pp. 1125–1134.

[7] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *ICCV*, 2017, pp. 2223–2232.

[8] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans," in *CVPR*, 2018, pp. 8798–8807.

[9] P. Esser, R. Rombach, and B. Ommer, "Taming transformers for high-resolution image synthesis," in *CVPR*, 2021, pp. 12 873–12 883.

[10] J.-Y. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, and E. Shechtman, "Toward multimodal image-to-image translation," *NeurIPS*, vol. 30, 2017.

[11] L. Jiang, C. Zhang, M. Huang, C. Liu, J. Shi, and C. C. Loy, "Tsit: A simple and versatile framework for image-to-image translation," in *ECCV*, 2020, pp. 206–222.

[12] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu, "Semantic image synthesis with spatially-adaptive normalization," in *CVPR*, 2019, pp. 2337–2346.

[13] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," *NeurIPS*, vol. 30, 2017.

[14] L. Zhang and M. Agrawala, "Adding conditional control to text-to-image diffusion models," *arXiv preprint arXiv:2302.05543*, 2023.

Fig. 1: More qualitative comparison with state-of-the-art methods on multi-human sketch-to-image translation task. Zoom in for better view.

Fig. 2: More qualitative comparison of synthesizing images from segmentation and compound condition of sketch and segmentation. Zoom in for better view.

Block name	Output size	Detail
Stage I - Style Injection Module		
Input	$256 \times 256 \times 128$	($StyleF_0$)
I_0	$128 \times 128 \times 256$	Conv2d(3 × 3, 128, 128, stride=1) + DownBlock(128, 256, 0.0) (output F_0)
I_1	$64 \times 64 \times 512$	StyleInjector(F_0, $StyleF_1$, 512, 256, 16, 0.0) + DownBlock(256, 512, 0.0) (output F_1)
I_2	$32 \times 32 \times 512$	StyleInjector(F_1, $StyleF_2$, 1024, 512, 16, 0.0) + DownBlock(512, 512, 0.0) (output F_2)
I_3	$16 \times 16 \times 512$	StyleInjector(F_2, $StyleF_3$, 1024, 512, 16, 0.0) + DownBlock(512, 512, 0.0) (output F_3)
I_4	$16 \times 16 \times 512$	StyleInjector(F_3, $StyleF_4$, 1024, 512, 16, 0.0) + ResnetBlock(512, 512, 0.0) + ResnetBlock(512, 512, 0.0) + AttnBlock(512) (output F_4)
OutLayer	$16 \times 16 \times 256$	GroupNorm(32, 512) + SELU() + Conv2d(3 × 3, 512, 256, stride=1, padding=1)
Decoder		
Input	$\frac{H}{16} \times \frac{W}{16} \times 256$	Input feature
InputLayer	$\frac{H}{16} \times \frac{W}{16} \times 256$	Conv2d(3 × 3, 256, 256, stride=1, padding 1) + ResnetBlock(512, 512, 0.0) + AttnBlock(512) + ResnetBlock(512, 512, 0.0)
D_0	$\frac{H}{8} \times \frac{W}{8} \times 256$	(ResnetBlock(512, 512, 0.0) + AttnBlock(512)) × 3 + interpolate(scaleFactor=2.0, mode=nearest) + Conv2d(3 × 3, 512, 512, stride=1, padding=1)
D_1	$\frac{H}{4} \times \frac{W}{4} \times 256$	ResnetBlock(512, 256, 0.0) + (ResnetBlock(256, 256, 0.0)) × 2 + interpolate(scaleFactor=2.0, mode=nearest) + Conv2d(3 × 3, 256, 256, stride=1, padding=1)
D_2	$\frac{H}{2} \times \frac{W}{2} \times 256$	ResnetBlock(256, 128, 0.0) + (ResnetBlock(128, 128, 0.0)) × 2 + interpolate(scaleFactor=2.0, mode=nearest) + Conv2d(3 × 3, 128, 128, stride=1, padding=1)
D_3	$H \times W \times 256$	ResnetBlock(128, 64, 0.0) + (ResnetBlock(64, 64, 0.0)) × 2 + interpolate(scaleFactor=2.0, mode=nearest) + Conv2d(3 × 3, 64, 64, stride=1, padding=1)
D_4	$H \times W \times 128$	ResnetBlock(128, 128, 0.0) × 3
OutLayer	$H \times W \times 3$	GroupNorm(32, 128) + SELU() + Conv2d(3 × 3, 128, 3, stride=1, padding=1)
Encoder		
Input	$H \times W \times c$	Conditional Image
InputLayer	$H \times W \times 3$	Conv2d(3 × 3, c, 128, stride=1, padding=1) + ReLU()
E_0	$\frac{H}{2} \times \frac{W}{2} \times 128$	DownBlock(128, 128, 0.0)
E_1	$\frac{H}{4} \times \frac{W}{4} \times 128$	DownBlock(128, 128, 0.0)
E_2	$\frac{H}{8} \times \frac{W}{8} \times 256$	DownBlock(128, 256, 0.0)
E_3	$\frac{H}{16} \times \frac{W}{16} \times 512$	DownBlock(256, 256, 0.0)
E_4	$\frac{H}{16} \times \frac{W}{16} \times 512$	ResnetBlock(256, 512, 0.0) + (AttnBlock(512) + ResnetBlock(512, 512, 0.0)) × 3
OutLayer	$16 \times 16 \times 256$	GroupNorm(32, 512) + SELU() + Conv2d(3 × 3, 512, 256, stride=1, padding=1)
Dual Embedding Module		
Input	$\frac{H}{16} \times \frac{W}{16} \times 512$	Concatenated Feature (Complementary Feature + Style Feature)
F_1	$\frac{H}{16} \times \frac{W}{16} \times 384$	Conv2d(3 × 3, 512, 384, stride=1, padding=1) + ReLU()
F_2	$\frac{H}{16} \times \frac{W}{16} \times 256$	Conv2d(3 × 3, 384, 256, stride=1, padding=1) + ReLU()
F_3	$\frac{H}{16} \times \frac{W}{16} \times 256$	(Conv2d(3 × 3, 256, 256, stride=1, padding=1) + ReLU()) × 2

TABLE 3: Model architecture of our proposed StyleGAN-∞, including the Style Injection Module, Encoder, Decoder, and Dual Embedding Module. Here, H and W are the height and width of the input image, respectively, and c represents the number of channels in the input condition image. $StyleF_x$ represents the features extracted from the StyleGANx layer (top to bottom), and F_x denotes the output of the x^{th} layer of the Style Injection Module. SELU() is the scaled exponential linear unit activation function proposed in [13].

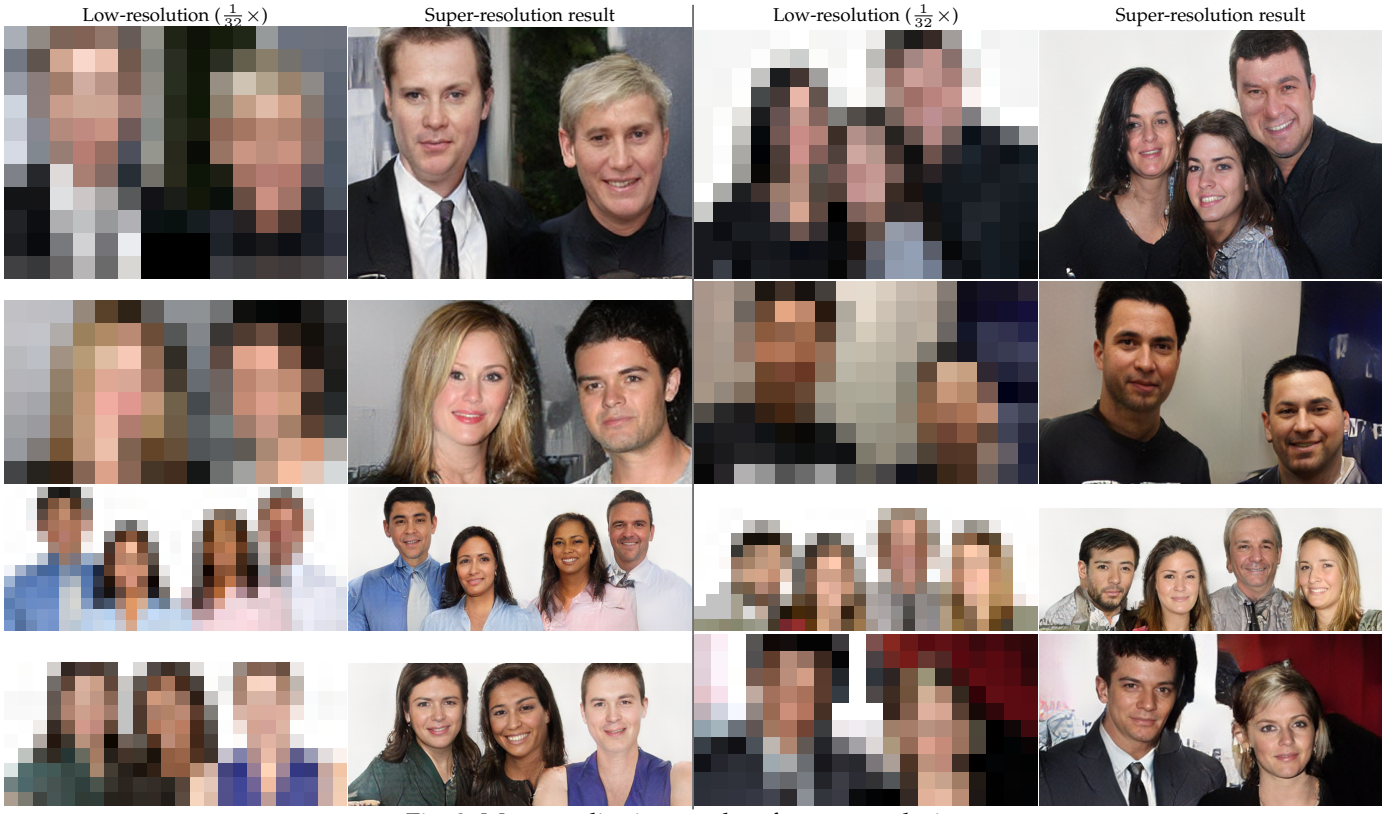

Fig. 3: More qualitative results of super-resolution.

Fig. 4: Samples synthesized from sketch under church domain.

Fig. 5: Samples synthesized from sketch under church domain.

Fig. 6: Samples synthesized from sketch under cat domain.

Fig. 7: Samples synthesized from sketch under horse domain.

Fig. 8: Samples synthesized from sketch under horse domain.