StyleSwin: Transformer-based GAN for High-resolution Image Generation (Supplementary material)

In the next, we present the architecture and implementation details. In addition, more analysis and qualitative results are provided. For better reproducibility, we will make the model publicly available.

1. Implementation Details

We train the StyleSwin using the standard non-saturating logistic GAN loss [4] with R_1 gradient penalty [10]. Specifically, the discriminator is trained to measure the realism of image samples whereas the generator is trained to generate samples that the discriminator mistakenly recognizes as real ones. In addition, the R_1 regularization term penalizes the gradient on real data to advocate the local stability. The training loss can be formulated as:

$$\mathcal{L}_D = -\mathbb{E}_{x\sim P_x}[\log(D(x))] - \mathbb{E}_{z\sim P_z}[\log(1 - D(G(z)))] + \gamma \cdot \mathbb{E}_{x\sim P_x}[\|\nabla_x D(x)\|_2^2],$$
$$\mathcal{L}_G = -\mathbb{E}_{z\sim P_z}[\log(D(G(z)))].$$

In practice, we perform R_1 gradient penalty every 16 iterations and the corresponding weight γ varies for different datasets.

The training follows the TTUR strategy [7] in which the discriminator adopts a $4\times$ larger learning rate than the generator. We linearly decay the learning rate to 0 from the LR decay start iteration for training all datasets except CelebA-HQ 1024. We apply spectral normalization [11] upon discriminator to ensure its Lipschitz continuity. The transformers are initialized with a truncated normal distribution [5] with zero mean and standard deviation of 0.02. For the convolution 1×1 used in tRGB layers, we use Glorot initialization [3] with a gain of 0.02. We use an exponential moving average of weights of generator [8] when sampling image, with a decay rate of 0.9978 following [9].

When synthesizing 256×256 resolution images of FFHQ and CelebA-HQ, the training benefits from balanced consistency regularization (bCR) [14]. Specifically, images are augmented by {*Flipping*, *Color*, *Translation*, *Cutout*} of probability {0.5, 1.0, 1.0, 1.0} as in DiffAug [13]. *Translation* is performed within $[-1/8, 1/8]$ of the image size, and random squares of half image size are masked when applying *Cutout*.

We implement the StyleSwin using Pytorch and conduct experiments with Tesla V100 GPUs. Training on 1024×1024 resolution takes about 14 days using 8 32GB GPUs. The hyper-parameters in the experiments are summarized in Table 1.

	FFHQ-256	CelebA-HQ 256	LSUN Church 256	FFHQ-1024	CelebA-HQ 1024
Training iteration	32.0M	25.6M	48M	25.6M	25.6M
Number of GPUs	8	8	8	16	16
Batch size	32	32	32	32	32
Learning rate of D	$2e-4$	$2e-4$	$2e-4$	$2e-4$	$2e-4$
Learning rate of G	$5e-5$	$5e-5$	$5e-5$	$5e-5$	$5e-5$
LR decay start iteration	24.8M	16M	41.6M	19.2M	-
R_1 regularization γ	10	5	5	10	10
bCR	✓	✓	✗	✗	✗

Table 1. Experiment settings for different datasets.

2. Detailed Architecture

StyleSwin starts from a constant input of size $4 \times 4 \times 512$ and hierarchically upsamples the feature map with transformer blocks. We use two transformer blocks to model each resolution scale. The detailed model architecture is shown in Table 2.

"Double attn, 512-d, 4-w, 16-h" indicates a double attention block with a channel dimension of 512, window size of 4, and 16 attention heads. "Bilinear upsampling, 512-d" indicates a bilinear upsampling layer followed by feedforward MLPs with an output dimension of 512.

Input size	StyleSwin-256	StyleSwin-1024
4×4	{ Double attn, 512-d, 4-w, 16-h MLP, 512-d } × 2	{ Double attn, 512-d, 4-w, 16-h MLP, 512-d } × 2
	Bilinear upsampling, 512-d	Bilinear upsampling, 512-d
8×8	{ Double attn, 512-d, 8-w, 16-h MLP, 512-d } × 2	{ Double attn, 512-d, 8-w, 16-h MLP, 512-d } × 2
	Bilinear upsampling, 512-d	Bilinear upsampling, 512-d
16×16	{ Double attn, 512-d, 8-w, 16-h MLP, 512-d } × 2	{ Double attn, 512-d, 8-w, 16-h MLP, 512-d } × 2
	Bilinear upsampling, 512-d	Bilinear upsampling, 512-d
32×32	{ Double attn, 512-d, 8-w, 16-h MLP, 512-d } × 2	{ Double attn, 512-d, 8-w, 16-h MLP, 512-d } × 2
	Bilinear upsampling, 512-d	Bilinear upsampling, 256-d
64×64	{ Double attn, 512-d, 8-w, 16-h MLP, 512-d } × 2	{ Double attn, 256-d, 8-w, 8-h MLP, 256-d } × 2
	Bilinear upsampling, 256-d	Bilinear upsampling, 128-d
128×128	{ Double attn, 256-d, 8-w, 8-h MLP, 256-d } × 2	{ Double attn, 128-d, 8-w, 4-h MLP, 128-d } × 2
	Bilinear upsampling, 128-d	Bilinear upsampling, 64-d
256×256	{ Double attn, 128-d, 8-w, 4-h MLP, 128-d } × 2	{ Double attn, 64-d, 8-w, 4-h MLP, 64-d } × 2
		Bilinear upsampling, 32-d
512×512		{ Double attn, 32-d, 8-w, 4-h MLP, 32-d } × 2
		Bilinear upsampling, 16-d
1024×1024		{ Double attn, 16-d, 8-w, 4-h MLP, 16-d } × 2

Table 2. The detailed generator architecture of StyleSwin-256 and StyleSwin-1024.

3. The Modeling Capacity of Double Attention

In order to prove the improved expressivity of the proposed double attention, we train an autoencoder for image reconstruction. Specifically, we adopt a conv-based encoder — a ResNet-50 pretrained from MoCo [6] such that both the low-level and high-level information are well preserved in the 16×16 feature map [12]. The latent feature map is further fed into the decoder for image reconstruction. The decoder adopts transformer blocks, which hierarchically upsamples the latent feature map and reconstructs the input. No style injection module is needed and we replace AdaIN with layer normalization. The decoder adopts either the vanilla Swin attention block or the proposed double attention. The autoencoders are trained with $\mathcal{L}_1$ loss. Figure 1 shows the training loss curve of the two autoencoders. One can see that the decoder with double attention shows faster convergence and yields lower reconstruction loss, indicating that the decoder that leverages enhanced receptive field shows stronger generative capacity.

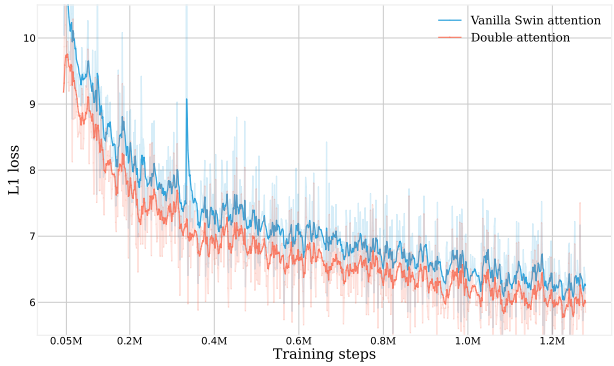

Figure 1. Image reconstruction training loss of autoencoders. The autoencoder adopts a fixed conv-based encoder and transformer-based decoder and is trained with $\mathcal{L}_1$ loss. The decoder with double attention shows improved modeling capacity over the vanilla Swin attention.

4. Additional Quantitative Evaluation

To further demonstrate StyleSwin's strong ability to model complex scenes and materials, we train our model on a subset of LSUN Car, which achieves comparable performance to state-of-the-art StyleGAN2. We also present additional quantitative evaluation results in terms of KID [1] and FID-Inf [2] on all evaluation datasets, comparing to StyleGAN2. The detailed measures are presented in Table 3 and Table 4.

Methods	FFHQ-256			Church-256			CelebAHQ-256			Car-256		
	FID	KID$\times 10^{-3}$	FID-Inf	FID	KID$\times 10^{-3}$	FID-Inf	FID	KID$\times 10^{-3}$	FID-Inf	FID	KID$\times 10^{-3}$	FID-Inf
StyleGAN2	3.62	1.45	1.37	3.86	1.71	1.53	-	-	-	**4.32**	1.63	**1.60**
StyleSwin	**2.81**	**0.54**	**0.83**	**2.95**	**1.02**	**1.44**	**3.25**	**0.61**	**1.36**	4.35	**1.53**	1.80

Table 3. Evaluation results comparing to StyleGAN2 on resolution 256 in terms of FID, KID and FID-Inf.

Methods	FFHQ-1024			CelebAHQ-1024		
	FID	KID$\times 10^{-3}$	FID-Inf	FID	KID$\times 10^{-3}$	FID-Inf
StyleGAN2[1]	**4.41**	**1.22**	**1.57**	5.17	1.71	1.53
StyleSwin	5.07	2.07	2.13	**4.43**	**1.42**	**2.08**

Table 4. Evaluation results comparing to StyleGAN2 on resolution 1024 in terms of FID, KID and FID-Inf. [1]We report the metrics of StyleGAN2 on FFHQ-1024 and that of StyleGAN on CelebA-HQ 1024.

5. More Qualitative Results

Latent code interpolation. To explore the property of the learned latent space of StyleSwin, we randomly sample two latent codes in the latent space and perform linear interpolation between them. As shown in Figure 2, our StyleSwin could produce smooth, meaningful image morphing with respect to different styles like gender, poses, and eyeglasses.

Additional image samples. We provide additional image samples generated by our StyleSwin. Figure 3 and Figure 4 show the impressive synthetic face images of FFHQ-1024 and CelebA-HQ 1024 with diverse viewpoints, backgrounds, and accessories, which illustrate the strong capacity of the proposed StyleSwin. Image samples of LSUN Church 256 and LSUN Car 256 are shown in Figure 5 and Figure 6, showing that our StyleSwin is capable to synthesize complex scenes with coherent structures and complicated materials with high-quality light effects.

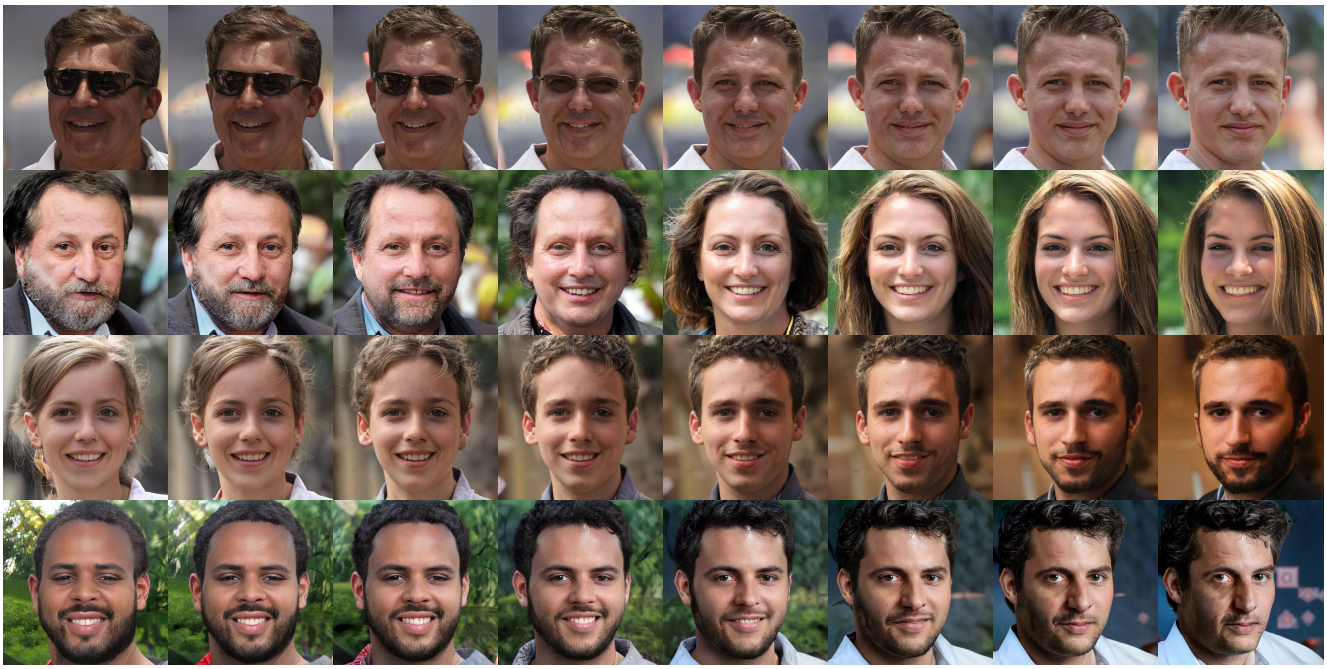

Figure 2. Latent interpolation results of the left-most and the right-most images on FFHQ 1024×1024.

6. Responsible AI Considerations

Our work does not directly modify the exiting images which may alter the identity or expression of the people. We discourage the use of our work in such applications as it is not designed to do so. We have quantitatively verified that the proposed method does not show evident disparity, on gender and ages as the model mostly follows the dataset distribution, however, we encourage additional care if you intend to use the system on certain demographic groups. We also encourage use of fair and representative data when training on customized data. We caution that the high-resolution images produced by our model may potentially be misused for impersonating humans and viable solutions so avoid this include adding tags or watermarks when distributing the generated photos.

7. Discussion of Limitation

Although, as stated in the main article, StyleSwin's theoretical FLOPs are smaller than StyleGAN2, there is a gap between the theoretical FLOPs and the throughput in practice. The throughput of StyleGAN2 and StyleSwin are 40.05 imgs/sec and 11.05 imgs/sec respectively on a single V100 GPU. This is primarily due to the fact that vision transformers have not been sufficiently optimized as ConvNets (e.g. using CuDNN), and we believe future optimization will democratize the usage of transformers as they exhibit lower theoretical FLOPs. Besides, bCR is not effective on 1024×1024, which we leave for further study.

Figure 3. Image samples of FFHQ 1024×1024.

Figure 4. Image samples of CelebA-HQ 1024×1024.

Figure 5. Image samples of LSUN Church 256×256.

Figure 6. Image samples of LSUN Car 256×256.

References

[1] Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. *arXiv preprint arXiv:1801.01401*, 2018. 3

[2] Min Jin Chong and David Forsyth. Effectively unbiased fid and inception score and where to find them. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6070–6079, 2020. 3

[3] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010. 1

[4] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014. 1

[5] Boris Hanin and David Rolnick. How to start training: The effect of initialization and architecture, 2018. 1

[6] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning, 2020. 2

[7] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018. 1

[8] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017. 1

[9] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2019. 1

[10] Lars M. Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *ICML*, 2018. 1

[11] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *ArXiv*, abs/1802.05957, 2018. 1

[12] Nanxuan Zhao, Zhirong Wu, Rynson WH Lau, and Stephen Lin. What makes instance discrimination good for transfer learning? *arXiv preprint arXiv:2006.06606*, 2020. 2

[13] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient gan training, 2020. 1

[14] Zhengli Zhao, Sameer Singh, Honglak Lee, Zizhao Zhang, Augustus Odena, and Han Zhang. Improved consistency regularization for gans, 2020. 1