Supplementary Materials for ECCV 2022 paper PixelFolder: An Efficient Progressive Pixel Synthesis Network for Image Generation

Jing He[1], Yiyi Zhou[1⋆], Qi Zhang[2], Jun Peng[1], Yunhang Shen[2], Xiaoshuai Sun[1,3], Chao Chen[2], and Rongrong Ji[1,3]

[1]Media Analytics and Computing Lab, Department of Artificial Intelligence, School of Informatics, Xiamen University. [2]Youtu Lab, Tencent. [3]Institute of Artificial Intelligence, Xiamen University.
{blinghe, pengjun}@stu.xmu.edu.cn, {zhouyiyi, xssun, rrji}@xmu.edu.cn, {merazhang, aaronccchen}@tencent.com, shenyunhang01@gmail.com

A More Experiments

A.1 Comparison of additional metrics for image quality

To further validate our PixelFolder in terms of image quality, we report two new indicators, namely Inception Score (IS) and Perceptual Path Length (PPL). The IS was shown to correlate well with the human scoring for judging the realism of generated images on CIFAR-10 dataset [6], the higher IS score indicators the better image quality. The PPL indicates the smoothness of the mapping from a latent space to the output images, which is very important for high quality image generation [3]. The smooth mapping will have a small PPL. In Tab. A, the overall performance of PixelFolder is still better.

Method	FFHQ, 256× 256		LSUN Church, 256 × 256	
	IS ↑	PPL ↓	IS ↑	PPL ↓
INR-GAN	4.51±0.08	**62**	2.69±0.02	529
CIPS	4.75±0.06	273	2.73±0.02	3132
StyleGAN2	4.86±0.04	209	2.78±0.02	350
PixelFolder	**5.04±0.07**	197	**2.79±0.03**	**307**

Table A: The comparison of inception score (IS) and perceptual path length (PPL) on FFHQ and LSUN Church. The higher IS score and the lower PPL score indicate the better performance in terms of image quality. These results further validate the superior of the proposed PixelFolder.

⋆ Corresponding Author.

Method	FFHQ, 512 × 512		FFHQ, 1024 × 1024	
	#Params↓	FID ↓	#Params↓	FID↓
INR-GAN	112.43	-	117.30	16.32
CIPS	145.12	-	574.32	-
StyleGAN2	30.28	4.15	30.37	**2.84**
PixelFolder	**20.84**	**4.08**	**21.14**	2.98

Table B: Comparison of high-resolution image generation. The proposed PixelFolder has better performance with fewer parameters than other methods.

Method	Cat	Bedroom
INR-GAN	-	3.40
CIPS	12.86	-
StyleGAN2	9.14	**2.65**
PixelFolder	**8.41**	3.71

Table C: Comparison of FID scores on LSUN Cat and LSUN Bedroom.

Method	FFHQ	Church
UDM(RVE)+ST [a]	5.54	-
UnleashTR [b]	6.11	4.07
StyleGAN2	3.83	3.86
PixelFolder	**3.77**	**2.46**

Table D: Comparison with diffusion models in terms of FID score.

A.2 Comparison of high-resolution image generation

In Tab. B, we compare the performance of high-resolution image generation between the proposed PixelFolder and other SOTA methods, *i.e.* 512 × 512 and 1024 × 1024. From these results, we observe that PixelFolder still has competitive performance in terms of model efficiency and image quality. Compared to StyleGAN2, PixelFolder obtains competitive FID values with only two-thirds of its parameters, and especially at 512 × 512 resolutions where PixelFolder obtains better performance. Note that the proposed PixelFolder is directly implemented without any tricks like adding additional noises. INR-GAN and CIPS are out of GPU memory to train.

A.3 Comparison on more datasets

We conduct additional experiments on two popular datasets as reported in Tab. C, *i.e.* LSUN cat and LSUN bedroom. The proposed PixelFolder still obtains competitive performance, which further demonstrates the generalizability of our PixelFolder.

A.4 Comparison with more typical methods

In addition to some GAN-based approaches [1,4], we further compare PixelFolder with the recently popular diffusion models [5,2] in Tab. D, where the merits of our method still can be seen.

B Experiment details

To ablate the pixel folding operations, we compare them with the alternatives of *down-sampling* and *deconvolution* (DeConv) in Tab.3 of the main paper. We can see that both alternative models, *i.e.*, Fold+DeConv and Down-Sampling+DeConv, have more parameters than Fold+Unfold (base). The main reason for the parameter gap is that the pixel unfolding operation reduces the number of channels of the hidden tensors, thus the corresponding convolution kernel has fewer parameters (see blue font in Tab. E).

In order to keep the shape of the hidden tensor with red color consistent, we further modify DeConv_0 and DeConv_1, *i.e.*, the kernels are replaced with $3 \times 3 \times 512 \times 128$ and $3 \times 3 \times 128 \times 512$, respectively. The results of the shape-consistent alternatives are shown in Tab. F. Although Fold+DeConv-sc and Down-Sampling+DeConv-sc guarantee the consistency of the shape of pixel tensors, their performance lags far behind that of Fold+Unfold (base), which further demonstrates the effectiveness of pixel folding operations.

Layers	Fold+UnFold (base)	Fold+DeConv	DownS.+DeConv
Initialization	$16 \times 16 \times 512$	$16 \times 16 \times 512$	$16 \times 16 \times 512$
projection	$16 \times 16 \times 32$	$16 \times 16 \times 32$	-
Folding$\times$2/DownS.	$4 \times 4 \times 512$	$4 \times 4 \times 512$	$4 \times 4 \times 512$
ModConv_0/DeConv_0	$4 \times 4 \times 512$ $(3 \times 3 \times 512 \times 512)$	$8 \times 8 \times 512$ $(3 \times 3 \times 512 \times 512)$	$8 \times 8 \times 512$ $(3 \times 3 \times 512 \times 512)$
Unfolding_0	$8 \times 8 \times 128$	-	-
ModConv_1	$8 \times 8 \times 512$ $(3 \times 3 \times 128 \times 512)$	$8 \times 8 \times 512$ $(3 \times 3 \times 512 \times 512)$	$8 \times 8 \times 512$ $(3 \times 3 \times 512 \times 512)$
ModConv_2/DeConv_1	$8 \times 8 \times 512$ $(3 \times 3 \times 512 \times 512)$	$16 \times 16 \times 512$ $(3 \times 3 \times 512 \times 512)$	$16 \times 16 \times 512$ $(3 \times 3 \times 512 \times 512)$
Unfolding_1	$16 \times 16 \times 128$	-	-
ModConv_3	$16 \times 16 \times 512$ $(3 \times 3 \times 128 \times 512)$	$16 \times 16 \times 512$ $(3 \times 3 \times 512 \times 512)$	$16 \times 16 \times 512$ $(3 \times 3 \times 512 \times 512)$
ToRGB	$16 \times 16 \times 3$ $(1 \times 1 \times 3 \times 512)$	$16 \times 16 \times 3$ $(1 \times 1 \times 3 \times 512)$	$16 \times 16 \times 3$ $(1 \times 1 \times 3 \times 512)$

Table E: The tensor shapes of the output of layers in the first row. $(k \times k \times c_{in} \times c_{out})$ represents the kernel size of corresponding convolution layer. The red font indicates the tensor shape before feeding into ModConv or DeConv, and the blue font indicates the kernel size of ModConv or DeConv with red tensor as input.

Settings	#Parm (M) ↓	GMACs ↓	FID ↓	Precision ↑	Recall ↑
Fold+Unfold (base)	**20.84**	**23.78**	**5.49**	**0.679**	**0.514**
Fold+DeConv	29.41	86.53	5.60	0.667	0.371
Fold+DeConv-sc	12.66	12.12	8.41	0.643	0.366
Down-Sampling+DeConv	29.21	89.38	5.53	0.679	0.456
Down-Sampling+DeConv-sc	12.46	12.97	8.36	0.670	0.442

Table F: Additional ablation study on FFHQ. The models of all settings are trained with 200k steps for a quick comparison. "sc" is short for "shape-consistent". The results further prove the advantages of pixel folding operations.

C Additional samples

In Fig. A, we provide more interpolations on LSUN Church to further demonstrate the generalization capability of PixelFolder. We also provide additional samples on different datasets in Fig. B. Similar to StyleGANv1v2 [3,4] and CIPS [1], our model also has the capability of style mixing controlled by stage-wise latent codes as illustrated in Fig. C.

Fig. A: Interpolations by PixelFolder on LSUN Church. The factor α of the interpolations is set to smaller than that of Fig.4 of the main paper. These interpolations further demonstrate the generalization capability of PixelFolder when the input noise fluctuates slightly.

(a) FFHQ. (b) LSUN Church.

(c) LSUN Bedroom. (d) LSUN Cat.

Fig. B: Additional samples of PixelFolder on different datasets, *i.e.*, FFHQ, LSUN Church, LSUN Bedroom and LSUN Cat.

Fig. C: Stage-wise style mixing of our PixelFolder trained on FFHQ. The left two columns are the images generated by latent code z_1 and z_2, respectively. The three columns on the right are the images generated by replacing z_1 at stage1, stage2 and stage3 with the corresponding latent code z_2.

References

1. Anokhin, I., Demochkin, K., Khakhulin, T., Sterkin, G., Lempitsky, V., Korzhenkov, D.: Image generators with conditionally-independent pixel synthesis. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. pp. 14278–14287 (2021)
2. Bond-Taylor, S., Hessey, P., Sasaki, H., Breckon, T.P., Willcocks, C.G.: Unleashing transformers: Parallel token prediction with discrete absorbing diffusion for fast high-resolution image generation from vector-quantized codes. arXiv preprint arXiv:2111.12701 (2021)
3. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. pp. 4401–4410 (2019)
4. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of stylegan. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. pp. 8110–8119 (2020)
5. Kim, H., Choi, Y., Kim, J., Yoo, S., Uh, Y.: Exploiting spatial dimensions of latent in gan for real-time image editing. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. pp. 852–861 (2021)
6. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training gans. Advances in neural information processing systems **29** (2016)