Incorporating Reinforced Adversarial Learning in Autoregressive Image Generation
-Supplementary Material-

1. Additional Image Samples

We provide additional image samples in Figure 1 and Figure 2 with the same models used in Figure 5 and Figure 6 of the main paper. The images have resolution of 128×128, decoded from latent code sizes of 16×16 and 32×32.

We further provide quantitative results in Table 1 for CelebA [2] and in Table 2 for LSUN-bedroom [5] datasets with different settings. Lastly, we also share an experiment on CelebA-HQ [1] dataset for image resolution of 512×512 in Table 3 using latent code sizes of 32×32 and 64×64. For all experiments, we finetune the pixelcnn model with the MLE loss with the same number of iterations as our RL trained model, then compare which one is better.

2. Architecture Details

We provide the details of PixelCNN [4] and VQVAE [3] architectures we used in our experiments in Tables 4, 5, 6, 7. 8, 9 and 10.

Figure 1: Additional image samples generated by MLE-trained model and our proposed reinforced adversarial learning on the CelebA dataset. The generated images have resolution of 128×128, and they are decoded from 16×16 and 32×32 latent codes.

Figure 2: Additional image samples generated by MLE-trained model and our proposed reinforced adversarial learning on the LSUN-bedroom dataset. The generated images have resolution of 128×128, and they are decoded from 16×16 and 32×32 latent codes.

3. Sampling Speed

We report FID for a partial generation and full generation under the same training time in the main paper, Table 3(a). More detailed training durations are reported in Table 11, computed with a single GPU. For 128×128 res., partial generation can sample a batch of 16 images in 61 seconds on average compared to 102 seconds with full generation. For 256×256 res., partial generation can sample a batch of 16 images in 123 seconds on average compared to 170 seconds with full generation. As can be seen, the partial generation significantly improves the training time. Due to smaller code sizes, the model is also less exhaustive to train as lesser GPU memory is allocated.

4. Scaling to Higher Resolutions

The main reason we do not have such successful results is due to computational limitations as discussed in the paper. Another reason is the VQ-VAE reconstruction and even calculating FIDs for real images vs. reconstructed real images, FIDs are very high as can be seen in Table 12. As we limit the encoded code size to effectively use PixelCNNs, FIDs reduce due to the increased compression. It is noteworthy that the ImageNet classifier is also very sensitive to small distortions caused by vector quantization as also discussed in VQ-VAE-2.

The superior performance of VQ-VAE-2 is also enabled by rejection sampling. Rejection sampling is an external method to filter out bad quality samples by using a classifier network that is trained on ImageNet and can be applied to ANY generative network. However, the rejection sampling method is a temporary solution, which makes it hard to use for real-world cases. By introducing Reinforced Adversarial Learning, our aim is to penalize bad quality samples and permanently improve PixenCNNs sample generation quality.

Another issue with rejection sampling is sampling costs. The sampling time for autoregressive models is already a problem and including rejection sampling would make this process even slower. For 256×256 resolution, as reported in VQ-VAE-2 supplementary materials, the sampling speed per image is 3 minutes even with caching and use of large batches. Therefore, VQ-VAE-2 requires many GPU hours, which is very costly. In the paper, we mention the disadvantages of rejection sampling such as increased sampling time, reduced diversity and unavailability of class scores.

Table 1: Additional experiment for CelebA dataset. FID calculations are made compared to reconstructed real images.

Image Size — Latent Code Size	128×128 — 16×16
MLE	18.01
Single Reward	16.91

Table 2: Additional experiments for LSUN-bedroom dataset. FID calculations are made compared to reconstructed real images.

Image Size — Latent Code Size	64×64 — 8×8	64×64 — 16×16	128×128 — 16×16
MLE	19.95	15.90	23.53
Single Reward	19.07	11.47	18.96

Table 3: Additional experiment for CelebA-HQ dataset. FID calculations are made compared to reconstructed real images.

Image Size — Latent Code Size	512×512 — top: 32×32, bottom: 64×64
MLE	34.93
Single Reward	32.14
Intermediate Reward	31.84

Table 4: Architecture details for CelebA, 64×64 image resolution experiments.

(a) PixelCNN Prior - Oracle Network

Input size	8×8
Batch size	1024
Hidden units	256
Residual units	256
Layers	5
Attention layers	0
Conv Filter size	5
Dropout	0.1
Output stack layers	5
Training steps	90039

(b) PixelCNN Prior - Generator Network

Input size	8×8
Batch size	1024
Hidden units	128
Residual units	128
Layers	5
Attention layers	0
Conv Filter size	5
Dropout	0.1
Output stack layers	5
Training steps	19531

(c) VQ-VAE: Encoder, Decoder

Input size	64×64
Latent Layers	8×8
β	0.25
Batch size	128
Hidden units	128
Residual units	64
Layers	3
Codebook size	128
Codebook dimension	64
Encoder filter size	3
Decoder filter size	4
Training steps	156250

Table 5: Architecture details for LSUN-bedroom, 64×64 image resolution experiments with latent size of 8×8.

(a) PixelCNN Prior

Input size	8×8
Batch size	1536
Hidden units	512
Residual units	512
Layers	20
Attention layers	0
Conv Filter size	5
Dropout	0.1
Output stack layers	5
Training steps	97656

(b) VQ-VAE: Encoder, Decoder

Input size	64×64
Latent Layers	8×8
β	0.25
Batch size	128
Hidden units	128
Residual units	64
Layers	3
Codebook size	512
Codebook dimension	64
Encoder filter size	3
Decoder filter size	4
Training steps	234375

Table 6: Architecture details for LSUN-bedroom, 64×64 image resolution experiments with latent size of 16×16.

(a) PixelCNN Prior

Input size	16×16
Batch size	480
Hidden units	512
Residual units	512
Layers	20
Attention layers	8
Conv Filter size	5
Dropout	0.1
Output stack layers	20
Training steps	31250

(b) VQ-VAE: Encoder, Decoder

Input size	64×64
Latent Layers	16×16
β	0.25
Batch size	128
Hidden units	128
Residual units	64
Layers	2
Codebook size	512
Codebook dimension	64
Encoder filter size	3
Decoder filter size	4
Training steps	234375

Table 7: Architecture details for CelebA and LSUN-bedroom with 128×128 resolution with a single prior PixelCNNs.

(a) PixelCNN Prior

Input size	16×16
Batch size	480
Hidden units	512
Residual units	512
Layers	20
Attention layers	8
Conv Filter size	5
Dropout	0.1
Output stack layers	20
Training steps	41666

(b) VQ-VAE: Encoder, Decoder

Input size	128×128
Latent Layers	16×16
β	0.25
Batch size	128
Hidden units	128
Residual units	64
Layers	3
Codebook size	512
Codebook dimension	64
Encoder filter size	3
Decoder filter size	4
Training steps	156250

Table 8: Architecture details for CelebA, LSUN-bedroom, 128×128 experiments with two PixelCNN priors.

(a) PixelCNN Top-Prior Network

Input size	16×16
Batch size	480
Hidden units	512
Residual units	512
Layers	20
Attention layers	8
Conv Filter size	5
Dropout	0.1
Output stack layers	20
Training steps	166664

(b) PixelCNN Bottom-Prior Network

Input size	32×32
Batch size	128
Hidden units	512
Residual units	512
Layers	20
Attention layers	0
Conv Filter size	5
Dropout	0.1
Conditioning Output stack layers	20
Training steps	156250

(c) VQ-VAE: Encoder, Decoder

Input size	128×128
Latent Layers	16×16, 32×32
β	0.25
Batch size	128
Hidden units	128
Residual units	64
Layers	2
Codebook size	512
Codebook dimension	64
Encoder filter size	3
Decoder filter size	4
Training steps	156250

Table 9: Architecture details for CelebA, 256×256 experiments with two PixelCNN priors.

(a) PixelCNN Top-Prior Network

Input size	32×32
Batch size	128
Hidden units	512
Residual units	512
Layers	20
Attention layers	0
Conv Filter size	5
Dropout	0.1
Output stack layers	20
Training steps	187500

(b) PixelCNN Bottom-Prior Network

Input size	64×64
Batch size	128
Hidden units	128
Residual units	128
Layers	16
Attention layers	0
Conv Filter size	5
Dropout	0.1
Conditioning Output stack layers	20
Training steps	125000

(c) VQ-VAE: Encoder, Decoder

Input size	256×256
Latent Layers	32×32, 64×64
β	0.25
Batch size	128
Hidden units	128
Residual units	64
Layers	2
Codebook size	512
Codebook dimension	64
Encoder filter size	3
Decoder filter size	4
Training steps	156250

Table 10: Architecture details for CelebA-HQ, 512×512 experiments with two PixelCNN priors.

(a) PixelCNN Top-Prior Network

Input size	32×32
Batch size	128
Hidden units	512
Residual units	512
Layers	20
Attention layers	0
Conv Filter size	5
Dropout	0.1
Output stack layers	20
Training steps	20625

(b) PixelCNN Bottom-Prior Network

Input size	64×64
Batch size	128
Hidden units	128
Residual units	128
Layers	16
Attention layers	0
Conv Filter size	5
Dropout	0.1
Conditioning Output stack layers	20
Training steps	42187

(c) VQ-VAE: Encoder, Decoder

Input size	512×512
Latent Layers	32×32, 64×64
β	0.25
Batch size	128
Hidden units	128
Residual units	64
Layers	3
Codebook size	512
Codebook dimension	64
Encoder filter size	3
Decoder filter size	4
Training steps	23437

Table 11: Sampling speed comparisons for full generation vs. partial generation with a batch size of 16.

Image Size — Latent Code Size	128×128 — $16 \times 16 + 32 \times 32$	256×256 — $32 \times 32 + 64 \times 64$
Full Generation	102 s	170 s
Partial Generation	61 s	123 s

Table 12: FIDs for real images vs. reconstructed real images.

Dataset	Celeba	Celeba	LSUN-bedroom
Image Size	128 x 128	256 x 256	128 x 128
Latent Code Size	16 x 16 + 32 x 32	32 x 32 + 64 x 64	16 x 16 + 32 x 32
FID: Real image vs. Reconstructed Real Image	24.52	7.47	28.54

References

[1] Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196 (2017) 1

[2] Liu, Z., Luo, P., Wang, X., Tang, X.: Deep learning face attributes in the wild. In: ICCV (December 2015) 1

[3] van den Oord, A., Vinyals, O., et al.: Neural discrete representation learning. In: Advances in Neural Information Processing Systems. pp. 6306–6315 (2017) 1

[4] Oord, A.v.d., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., Kavukcuoglu, K.: Conditional image generation with pixelcnn decoders. In: NeurIPS. pp. 4797–4805 (2016) 1

[5] Yu, F., Zhang, Y., Song, S., Seff, A., Xiao, J.: Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. arXiv preprint arXiv:1506.03365 (2015) 1