SoftVQ-VAE: Efficient 1-Dimensional Continuous Tokenizer

Supplementary Material

A. Posterior of KL-VAE and VQ-VAE

In this section, we provide the detailed derivation of the KL-divergence of KL-VAE and VQ-VAE used in Eq. (1) and Eq. (2), respectively.

KL-VAE has the KL divergence of the posterior with a Gaussian prior:

$$\begin{aligned}&\mathcal{L}_{\text{kl}}\left(q_\phi(\mathbf{z})\|p(\mathbf{z})\right).\\&=\int q_\phi(\mathbf{z})\left(\log q_\phi(\mathbf{z})-\log p(\mathbf{z})\right)d\mathbf{z}\\&=\frac{1}{2}\sum_{j=1}^{D}\left(1+\log\left((\sigma_j)^2\right)-(\mu_j)^2-(\sigma_j)^2\right),\end{aligned}\tag{8}$$

where D is the latent space dimension.

VQ-VAE assumes a uniform prior over the total K codewords in the learnable codebook for the deterministic posterior, thus present a KL divergence as follows:

$$\begin{aligned}&\mathcal{L}_{\text{kl}}\left(q_\phi(\mathbf{z})\|p(\mathbf{z})\right).\\&=\int q_\phi(\mathbf{z})\left(\log q_\phi(\mathbf{z})-\log p(\mathbf{z})\right)d\mathbf{z}\\&=-(-\log K)\\&=\log K\end{aligned}\tag{9}$$

A.1. Comparison to VQ

We list the advantages of SoftVQ in Tab. 5 with empirical support, and will add in the revision.

Table 5. Advantages of SoftVQ over VQ.

Advantages	VQ	SoftVQ	Empirical Support
Grad. Broken	yes	no	No straight-through trick. SoftVQ-S-64 has a 1.41 rFID and 13.35 gFID.
Codebook Loss	yes	no	Not necessary. Adding codebook loss in SoftVQ-S-64 results in an 1.34 rFID of of but a worse gFID of 15.15.
Commit Loss	yes	no	Not necessary. Adding commit loss in SoftVQ-S-64 results in an rFID of 1.33 of but a worse gFID of 14.35.
Comp. Ratio	low	high	SoftVQ has much lower rFID<1.0 comapred to TiTok in Tab. 1.
Rep. Align.	difficult	easy	More discriminative features in main Fig. 4

While representation alignment can also applied to VQ, it fails to learn discriminative features with high compression ratio, mainly due to its broken gradient. Compared to SoftVQ, whose encoder and codebook's parameters are directly learned by the alignment loss, the gradient of alignment loss on VQ will first be straight-through from decoder input to encoder output, and then the codebook is updated according to the codebook loss, resulting in indirect learning of discriminative features.

We provide an additional codewords visualization in Fig. 5, where SoftVQ learns code embeddings uniformly across the entire distribution.

B. Additional Details of SoftVQ-VAE and its Variants

In this section, we present more details on the posterior of SoftVQ-VAE, its variant GMMVQ-VAE with the latent space as a GMM model, and the compatibility of SoftVQ-VAE with improvement techniques of VQVAE.

B.1. Posterior of SoftVQ-VAE

In SoftVQ, we similarly assume that the prior is a uniform distribution over the K learnable codewords as in VQ, except for we have the posterior as the softmax probability:

$$\begin{aligned}&\mathcal{L}_{\text{kl}}\left(q_\phi(\mathbf{z})\|p(\mathbf{z})\right).\\&=\int q_\phi(\mathbf{z})\left(\log q_\phi(\mathbf{z})-\log p(\mathbf{z})\right)d\mathbf{z}\\&=H(q_\phi(\mathbf{z}))-H(q_\phi(\mathbf{z}),p(\mathbf{z})),\end{aligned}\tag{10}$$

where $H(q_\phi(\mathbf{z})$ is the entropy for $H(q_\phi(\mathbf{z}),p(\mathbf{z}))$ is the cross-entropy between $q_\phi(\mathbf{z})$ and the uniform prior $p(\mathbf{z})\sim\mathcal{U}(0,K)$. In practice, $H(\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}[q_\phi(\mathbf{z})],p(\mathbf{z}))$ we compute the $\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}[q_\phi(\mathbf{z})]$ instead, where $\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}[q_\phi(\mathbf{z})]$ is computed on the averaged batch data during training.

B.2. GMMVQ-VAE

As discussed in the main paper, the latent space of SoftVQ can be interpreted as soft K-Means, and it can be further extended to a latent space of Gaussian Mixture Model. We present more details of this extension here.

In GMMVQ-VAE, the encoder predicts two things: the embedding $\hat{\mathbf{z}}$ and the weights of the Gaussian component $\omega(\hat{\mathbf{z}})$. We can then formulate the posterior as:

$$\begin{aligned}&\textit{posterior: } q_\phi(\mathbf{z}|\mathbf{x})=\text{Softmax}\left(-\omega(\hat{\mathbf{z}})\|\hat{\mathbf{z}}-\mathcal{C}\|_2\right)\\&\textit{latent: } \mathbf{z}=q_\phi(\mathbf{z}|\mathbf{x})\mathcal{C}\\&\textit{kl: } \mathcal{L}_{\text{kl}}=H(q_\phi(\mathbf{z}|\mathbf{x}))-H\left(\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}q_\phi(\mathbf{z}|\mathbf{x})\right).\end{aligned}\tag{11}$$

The difference between SoftVQ and GMMVQ in our formulation is the way in computing the posterior, *i.e.*, using fixed temperature parameter versus learning the data-dependent temperature parameters by the encoder. Note that

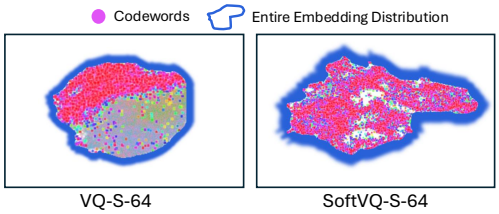

Figure 5. Codewords visualization

we still maintain the codebook and its codewords entries directly for the decoder input. It is possible to make the latent space formally a GMM, by treating the codewords as Gaussian means, and formulating the decoder input with the re-parametrization trick. However, we find that this formulation with re-parametrization will hinder the learning of the latent space. Thus, we adopted the simpler design to use the codebook directly for reconstruction, assuming fixed variance in the Gaussian components of GMM.

B.3. Compatibility of SoftVQ-VAE

Since SoftVQ-VAE maintains the learnable codebook, previous techniques to improve VQ-VAE are also compatible with it. Here, we follow ImageFolder [62] to show the combination of SoftVQ with product quantization [45] and residual quantization [55], as illustrated in Fig. 6.

Product Quantization (PQ) [45] divides the latent codes into G groups, with each group having its own codebook:

$$\begin{aligned} \mathbf{z} &= [\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, ..., \mathbf{z}^{(G)}] \\ q_\phi(\mathbf{z}^{(g)}|\mathbf{x}) &= \text{Softmax}(-\|\hat{\mathbf{z}}^{(g)} - \mathcal{C}^{(g)}\|_2/\tau). \end{aligned} \tag{12}$$

PQ can effectively increase the actual codebook size.

Residual Quantization (RQ) [55] applies multiple layers of quantization to the residual errors of the encoder output:

$$\begin{aligned} \mathbf{z}_l &= \mathbf{z}_{l-1} + \text{SoftVQ}(\mathbf{r}_{l-1}) \\ \mathbf{r}_l &= \mathbf{r}_{l-1} - \mathbf{z}_l \\ \mathbf{z}_0 &= 0 \\ \mathbf{r}_0 &= \hat{\mathbf{z}} \end{aligned} \tag{13}$$

where $\mathbf{r}$ is the residual and l is the layer index. RQ captures more fine-grained features and thus better reconstruction.

C. Additional Implementation Details

In this section, we provide more implementation details of the tokenizer training, the downstream generative models training, and the linear probing evaluation.

C.1. Implementation Details of SoftVQ-VAE

All tokenizer experiments (except the open-source ones) in this paper are trained using the same training recipe. We train the tokenizers on ImageNet [15] of resolution 256×256 and 512×512 for 250K iterations on 8 MI250 GPUs. Training for longer may lead to further improvement of reconstruction and potentially downstream generation performance [71, 108]. AdamW [68] optimizer is used with $\beta_1 = 0.9$, $\beta_2 = 0.95$, a weight decay of $1e$-4, a maximum learning rate of $1e$-4, and cosine annealing scheduler with a linear warmup of 5K steps. We use a constant batch size of 256 is used for all models. For the training objective, we set $\lambda_1 = 1.0$, $\lambda_1 = 0.2$, $\lambda_3 = 0.1$, and $\lambda_4 = 0.01$, following previous common practice. We additionally linearly warmup the loss weight of perceptual loss, *i.e.*, $\lambda_1 = 1.0$, for 10K iterations, which we find beneficial to train with fewer latent tokens.

Similarly to Tian et al. [99] and Li et al. [62], we found that the discriminator is very important for training the tokenizer. Instead of using the common PatchGAN and StyleGAN architecture, we adopted the frozen DINO-S [9, 79] discriminator with a similar architecture to StyleGAN [48, 49], as in [99]. In addition, we use DiffAug [121], consistency regularization [119], and LeCAM regularization [101] for discriminator training as in [99]. The loss weight of consistency regularization is set to 4.0 and LeCAM regularization is set to 0.001. We did not use the adaptive discriminator loss weight since it is too tricky to tune.

C.2. Implementation Details of DiT, SiT, and MAR

DiT. The training recipe of our DiT models mainly follows the original setup. Since we are using 1D latent tokens, we set the patch size of DiT models to 1 and use 1D absolute position encoding. We use a constant learning rate of $1e$-4 and a global batch size of 256 to train the DiT models. We use the cosine noise scheduler since it suits better for our case, but very similar generation performance were obtained in our early experiments with linear scheduler. To accelerate training, we additionally adopt mixed precision with bfloat16 and flash-attention. DiT-L models are trained for 400K iterations. In our main paper, we report the results of DiT-XL models for training 3M iterations. We report more and better results for longer training, *i.e.*, up to 4M iterations, in Appendix D.2. For conditional generation with CFG, we use 1.35 for DiT models trained on SoftVQ with 64 tokens and 1.45 for DiT models trained on SoftVQ with 32 tokens. These guidance scale values are obtained via grid search.

SiT. Similarly, we use the original setup to train the SiT models, with a constant learning rate of $1e$-4 and a global batch size of 256. We also use cosine scheduler in training of SiT models, as in our DiT models. The main results are reported with training for 3M iterations, and we include the training results for up to 4M iterations in Appendix D.2. For conditional generation with CFG, we use 1.75 for SiT models trained on SoftVQ with 64 tokens and 2.25 for models trained on SoftVQ with 32 tokens. Similarly to RPEA [116], we set the guidance interval to [0, 0.7] for CFG results [53]. These guidance scale values are obtained via grid search.

MAR. The MAR-H are trained using the AdamW optimizer for 500 epochs. The weight decay and momenta for AdamW are 0.02 and (0.9, 0.95). We also use a batch size of 2048 as in the original setup. Li et al. [60] used a linearly scaled learning rate of $8e$-4. However, we found that this learning rate constantly causes the NaN problems in loss. Thus we adopt a maximum learning rate of $2e$-4. We also train the models with a 100-epoch linear warmup of the learning rate, followed by a constant schedule. We suspect that the lower performance of our MAR models is caused by the

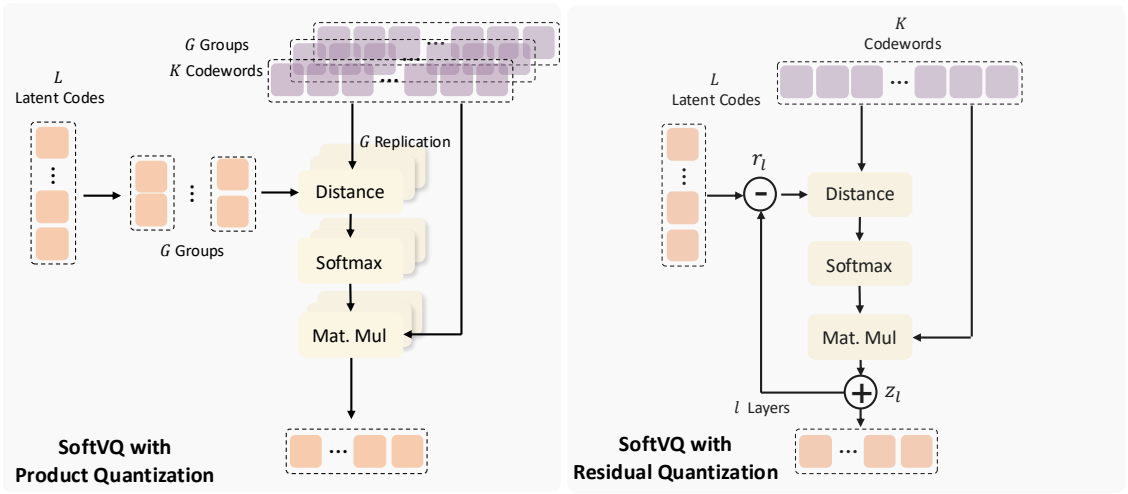

Figure 6. Illustration of product quantization (left) and residual quantization (right) with SoftVQ.

lower learning rate, which effectively results in fewer training iterations. More investigation on improving the MAR performance of SoftVQ is undergoing and will be updated in our future iterations. We use a CFG of 3.0 for conditional generation results.

We train all of our 256×256 generative models on 8 MI250 GPUs and 512×512 models on 8 MI300 GPUs.

C.3. Implementation Details of Linear Probing

We use the setup similar to that used in MAE [35], DAE [14], and REPA [3] to perform the linear probing evaluation in the latent tokenizer space and intermediate features of trained SiT models. Specifically, we train a parameter-free batch normalization layer and a linear layer for 90 epochs with a batch size of 16,384. We use the Lamb optimizer with cosine decay learning rate scheduler where the initial learning rate is set to 0.003.

D. Additional Results

In this section, we present our ablation study on the SoftVQ-VAE tokenizer, more results of the trained generative models including results from longer training, and more visualization of the reconstruction results from the tokenizer and the generative models.

D.1. Ablation Study

The full ablation results are shown in Tab. 6.

SoftVQ variants. We compare our baseline SoftVQ-S with several variants, including different quantization approaches. Adding product quantization (PQ) with group size G=2 and G=4 both improve the performance, reducing rFID from 1.33 to 1.19 and 1.03 respectively. Although residual quantization (RQ) alone increases rFID to 1.55, combining it with product quantization (R-PQ) yields better results with rFID of 1.17.

Table 6. Ablation study of SoftVQ-VAE

Variants						
Tokenizer	SoftVQ-S	+ PQ (G=2)	+ PQ (G=4)	+ RQ	+ R-PQ (G=4)	GMMVQ-S
rFID	1.33	1.19	1.03	1.55	1.17	1.12
Codebook Size						
K	512	1024	2048	4096	8192	16384
rFID	1.82	1.58	1.35	1.14	1.03	1.23
Latent Size						
L/D	64/16	64/32	64/64	32/32	32/64	32/128
rFID	2.31	1.03	0.63	1.61	1.24	0.79
Softmax Temp.						
τ	0.001	0.01	0.07	0.1	1.0	learnable
rFID	2.03	1.37	1.03	1.19	1.52	1.10

The GMMVQ-S variant also shows competitive performance with an rFID of 1.12.

Codebook size. We investigate the impact of codebook size K ranging from 512 to 16384. The results show that larger codebooks generally lead to better performance, with rFID decreasing from 1.82 (K=512) to 1.03 (K=8192). However, further increasing K to 16384 slightly degrades performance with an rFID of 1.23, suggesting that an overly large codebook may harm model stability.

Latent size. The latent dimensions L/D significantly affect the model performance. Starting from a small latent size of 64/16, increasing the dimensions initially helps, with 64/32 achieving the best rFID of 1.03. While further enlargement leads to better reconstruction performance, as seen in the 64/64 configuration with an rFID of 0.63, we find that the larger dimension of the latent space makes the generative models more challenging to learn.

Softmax temperature. The softmax temperature τ controls the sharpness of the posterior probability. We observe that very low temperatures τ =0.001 result in poor performance, while moderate values around 0.07 achieve optimal results. Making τ learnable produces competitive performance, suggesting that adaptive temperature could be a practical choice.

D.2. Main Results

We report the full results, including precision and recall, for the ImageNet 256×256 benchmark in Tab. 7 and 512×512 benchmark in Tab. 8, respectively. All main results reported are evaluated on SiT-XL and DiT-XL models trained for 3M iterations, and MAR models trained for 500 epochs. Noteworthy is that our models achieve performance comparable to state-of-the-art systems. More importantly, on 512×512 generation, SoftVQ with 32 tokens provides a **100**x speedup on the inference throughput.

An interesting finding is that, even though our models in general present the best conditional gFID without using CFG, their performance improvement with CFG becomes smaller, compared to SiT-XL/2 + REPA [116]. We leave the investigation of the reasons behind this observation for our future work.

We also provide results of longer training, *i.e.*, 4M iterations, and results along the training in Tab. 9. One can observe that the performance of our models does not saturate at 3M iterations of training, and longer training may provide further performance improvement.

D.3. More Comparison to TiTok

We provide a comparison between TiTok and SoftVQ in Tab. 10, including token numbers, model parameters, rFID of IN-1K and MSCOCO, linear probing (LP) accuracy, gFID and IS of SiT-L. We show that SoftVQ significantly outperforms TiTok with VQ and KL. Also, we need to note that while the model architectures are similar, training is different. TiTok is trained using 2-stage decoders with a pre-trained tokenizer, whereas our method is trained end-to-end with a single decoder. Also note that, although soft assignment of codewords may not be a new idea, our work is indeed the first to explore it with continuous tokenizer that achieves high compression ratio.

D.4. Reconstruction Visualization

We present a visualization of the SoftVQ-L reconstruction results with 32 and 64 tokens in Fig. 7 and Fig. 8, respectively.

D.5. Generation Visualization

More visualizations of SiT-XL trained on SoftVQ-L with 32 and 64 tokens are shown here.

Table 7. **System-level comparison** on ImageNet 256×256 conditional generation. We compare with both diffusion-based models and auto-regressive models with different types of tokenizers. Grey denotes SoftVQ. Our DiT/SiT results are reported with a total training of 3M iterations (compared to 4M of SiT/XL-2 + REPA and 7M for SiT/XL2 and DiT-XL/2), and MAR results are reported with a total training of 500 epochs with smaller learning rate (compared to 800 epochs of MAR-H). † indicates results with DPM-Solver and 50 steps.

Gen. Model	Tok. Model	# Tokens ↓	Tok. rFID ↓	Gflops ↓	Throughput (imgs/sec) ↑	w/o CFG				w/ CFG			
						gFID ↓	IS ↑	Prec. ↑	Recall ↑	gFID ↓	IS ↑	Prec. ↑	Recall ↑
Auto-regressive Models													
Taming-Trans. [23]	VQ	256	7.94	-	-	5.20	290.3	-	-	-	-	-	-
RQ-Trans. [55]	RQ	256	3.20	908.91	7.85	3.80	323.7	-	-	-	-	-	-
MaskGIT [10]	VQ	256	2.28	-	-	6.18	182.1	0.80	0.51	-	-	-	-
MAGE [59]	VQ	256	-	-	-	6.93	195.8	-	-	-	-	-	-
LlamaGen-3B [98]	VQ	256	2.08	781.56	2.90	-	-	-	-	3.06	279.7	0.80	0.58
TiTok-S-128 [115]	VQ	128	1.61	33.03	6.50	-	-	-	-	1.97	281.8	-	-
MAR-H [60]	KL	256	1.22	145.08	0.12	2.35	227.8	0.79	0.62	1.55	303.7	0.81	0.62
MAR-H	SoftVQ-L	32	0.61	67.93	2.19	3.83	211.2	0.77	0.61	2.54	273.6	0.78	0.61
	SoftVQ-BL	64	0.65	86.55	0.89	2.81	218.3	0.78	0.62	1.93	289.4	0.80	0.61
Diffusion-based Models													
LDM-4 [88]	KL	4096	0.27	157.92	0.37	10.56	103.5	0.71	0.62	3.60	247.7	0.87	0.48
U-ViT-H/2† [4]				128.89	0.98	-	-	-	-	2.29	263.9	0.82	0.57
MDTv2-XL/2 [29]				125.43	0.59	5.06	155.6	0.72	0.66	1.58	314.7	0.79	0.65
DiT-XL/2 [80]	KL	1024	0.62	80.73	0.51	9.62	121.5	0.67	0.67	2.27	278.2	0.83	0.53
SiT-XL/2 [72]				81.92	0.54	8.30	131.7	0.68	0.67	2.06	270.3	0.82	0.59
+ REPA [116]						5.90	157.8	0.70	0.69	1.42	305.7	0.80	0.65
DiT-XL	SoftVQ-B	32	0.89	14.52	8.94	9.83	113.8	0.70	0.61	3.91	264.2	0.81	0.54
	SoftVQ-BL		0.68		8.81	9.22	115.8	0.71	0.61	3.78	266.7	0.82	0.54
	SoftVQ-L		0.74		8.74	9.07	117.2	0.71	0.61	3.69	270.4	0.83	0.53
DiT-XL	SoftVQ-B	64	0.88	28.81	4.70	6.62	129.2	0.75	0.62	3.29	262.5	0.84	0.54
	SoftVQ-BL		0.65		4.59	6.53	131.9	0.75	0.62	3.11	268.3	0.84	0.56
	SoftVQ-L		0.61		4.51	5.83	141.3	0.75	0.62	2.93	268.5	0.84	0.55
SiT-XL	SoftVQ-B	32	0.89	14.73	10.28	7.99	129.3	0.70	0.63	2.51	301.3	0.76	0.62
	SoftVQ-BL		0.68		10.12	7.67	133.9	0.70	0.63	2.44	308.8	0.76	0.63
	SoftVQ-L		0.74		10.11	7.59	137.4	0.71	0.63	2.44	310.6	0.77	0.63
SiT-XL	SoftVQ-B	64	0.88	29.23	5.51	5.98	138.0	0.74	0.64	1.78	279.0	0.80	0.63
	SoftVQ-BL		0.65		5.42	5.80	143.5	0.74	0.64	1.88	287.9	0.80	0.63
	SoftVQ-L		0.61		5.39	5.35	151.2	0.74	0.64	1.86	293.6	0.81	0.63

Table 8. **System-level comparison** on ImageNet 512×512 conditional generation. We compare with both diffusion-based models and auto-regressive models with different types of tokenizers. Grey denotes SoftVQ. † indicates results with DPM-Solver and 50 steps.

Gen. Model	Tok. Model	# Tokens ↓	Tok. rFID ↓	Gflops ↓	Throughput (imgs/sec) ↑	w/o CFG				w/ CFG			
						gFID ↓	IS ↑	Prec. ↑	Recall ↑	gFID ↓	IS ↑	Prec. ↑	Recall ↑
Generative Adversarial Models													
BigGAN [10]	-	-	-	-	-	-	-	-	-	8.43	177.9	-	-
StyleGAN-XL [48]	-	-	-	-	-	-	-	-	-	2.41	267.7	-	-
Auto-regressive Models													
MaskGIT [10]	VQ	1024	1.97	-	-	7.32	156.0	-	-	-	-	-	
TiTok-B [115]	VQ	128	1.52	-	-	-	-	-	-	2.13	261.2	-	-
MAR-H	SoftVQ-BL	64	0.71	86.55	1.50	8.21	152.9	0.69	0.59	3.42	261.8	0.77	0.61
Diffusion-based Models													
ADM [16]	-	-	-	-	-	23.24	58.06	-	-	3.85	221.7	0.84	0.53
U-ViT-H/4† [4]				128.92	0.58	-	-	-	-	4.05	263.8	0.84	0.48
DiT-XL/2 [80]	KL	4096	0.62	373.34	0.10	9.62	121.5	-	-	3.04	240.8	0.84	0.54
SiT-XL/2 [72]				373.32	0.10	-	-	-	-	2.62	252.2	0.84	0.57
DiT-XL [11]	AE	256	0.22	80.75	1.02	9.56	-	-	-	2.84	-	-	-
UViT-H† [11]				128.90	6.14	9.83	-	-	-	2.53	-	-	-
UViT-H† [11]	AE	64	0.22	32.99	15.23	12.26	-	-	-	2.66	-	-	-
UViT-2B† [11]				104.18	9.44	6.50	-	-	-	2.25	-	-	-
SiT-XL	SoftVQ-L	32	0.64	14.73	10.12	10.17	119.2	0.65	0.59	4.23	218.0	0.83	0.52
	SoftVQ-BL	64	0.71	29.23	5.38	7.96	133.9	0.73	0.63	2.21	290.5	0.85	0.59

Table 9. Generation performance over training of DiT-XL and SiT-XL trained on SoftVQ-L with 32 and 64 tokens.

Model	Training Iter.	w/o CFG				w/ CFG			
		FID	IS	Prec.	Recall	FID	IS	Prec.	Recall
DiT/XL SoftVQ-L 32	3M	9.07	117.2	0.71	0.61	3.69	270.4	0.83	0.53
	4M	8.54	124.0	0.72	0.62	3.58	281.9	0.82	0.53
DiT/XL SoftVQ-L 64	3M	5.83	141.3	0.75	0.62	2.93	268.5	0.84	0.55
	4M	5.60	144.5	0.76	0.63	2.84	270.1	0.85	0.54
SiT/XL SoftVQ-L 32	400K	16.49	79.1	0.65	0.61	4.61	281.2	0.86	0.47
	1M	10.30	109.0	0.69	0.63	2.85	284.0	0.76	0.61
	2M	8.52	122.8	0.71	0.64	2.51	296.3	0.77	0.63
	3M	7.59	137.4	0.71	0.63	2.44	310.6	0.77	0.63
	4M	7.32	138.6	0.72	0.63	2.38	311.5	0.77	0.63
SiT/XL SoftVQ-L 64	400K	10.03	103.4	0.71	0.62	3.12	236.3	0.77	0.61
	1M	7.14	125.4	0.72	0.64	2.09	254.9	0.79	0.62
	2M	5.88	140.6	0.73	0.64	1.89	284.4	0.80	0.63
	3M	5.35	151.2	0.74	0.64	1.86	293.6	0.81	0.63
	4M	5.21	158.5	0.75	0.64	1.79	297.8	0.81	0.64

Figure 7. Reconstruction results of SoftVQ-L with 32 tokens.

Figure 8. Reconstruction results of SoftVQ-L with 64 tokens.

Figure 9. Uncurated 256×256 generation results of DiT-XL with SoftVQ-L 64 tokens. We use CFG with 4.0. Class label = ”balloon” (417).

Figure 10. Uncurated 256×256 generation results of DiT-XL with SoftVQ-L 64 tokens. We use CFG with 4.0. Class label = “baseball” (429).

Figure 11. Uncurated 256×256 generation results of DiT-XL with SoftVQ-L 64 tokens. We use CFG with 4.0. Class label = “dog sled” (537).

Figure 12. Uncurated 256×256 generation results of DiT-XL with SoftVQ-L 64 tokens. We use CFG with 4.0. Class label = “space shuttle” (812).

Figure 13. Uncurated 256×256 generation results of DiT-XL with SoftVQ-L 32 tokens. We use CFG with 4.0. Class label = “panda” (388).

Figure 14. Uncurated 256×256 generation results of DiT-XL with SoftVQ-L 32 tokens. We use CFG with 4.0. Class label = “geyser” (974).

Figure 15. Uncurated 256×256 generation results of SiT-XL with SoftVQ-L 64 tokens. We use CFG with 4.0. Class label = “golden retriever” (207).

Figure 16. Uncurated 256×256 generation results of SiT-XL with SoftVQ-L 64 tokens. We use CFG with 4.0. Class label = “arctic wolf” (270).

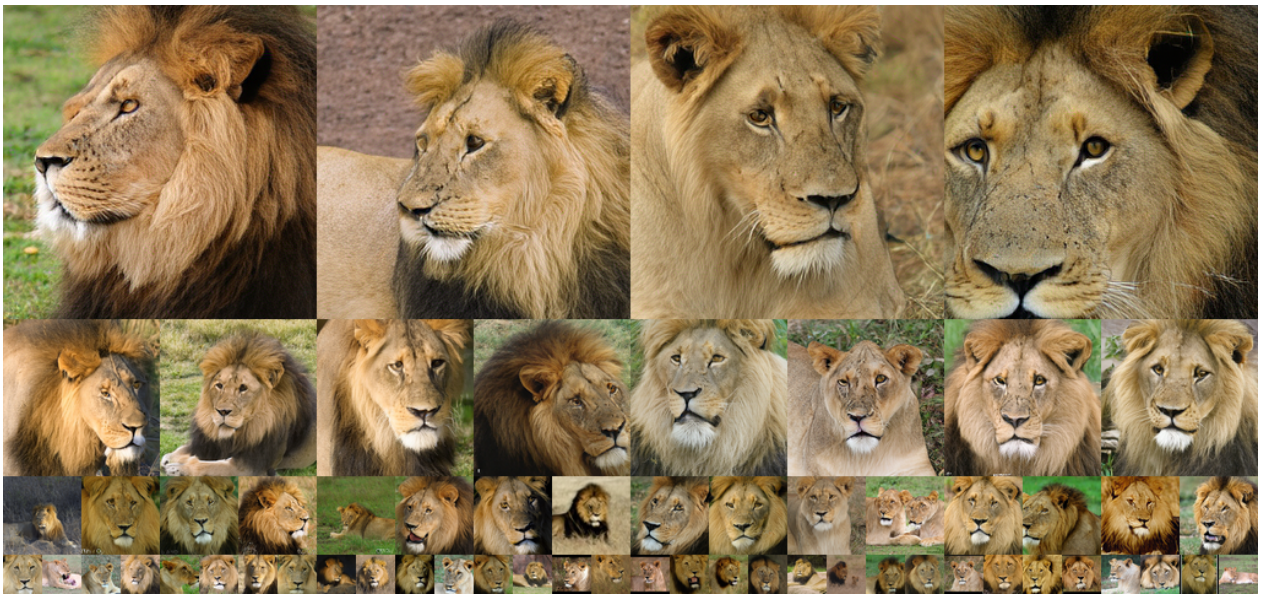

Figure 17. Uncurated 256×256 generation results of SiT-XL with SoftVQ-L 64 tokens. We use CFG with 4.0. Class label = “lion” (291).

Figure 18. Uncurated 256×256 generation results of SiT-XL with SoftVQ-L 64 tokens. We use CFG with 4.0. Class label = “loggerhead sea turtle” (33).

Figure 19. Uncurated 256×256 generation results of SiT-XL with SoftVQ-L 64 tokens. We use CFG with 4.0. Class label = “fire truck” (555).

Figure 20. Uncurated 256×256 generation results of SiT-XL with SoftVQ-L 64 tokens. We use CFG with 4.0. Class label = “laptop” (620).

Figure 21. Uncurated 256×256 generation results of SiT-XL with SoftVQ-L 64 tokens. We use CFG with 4.0. Class label = “macaw” (88).

Figure 22. Uncurated 256×256 generation results of SiT-XL with SoftVQ-L 64 tokens. We use CFG with 4.0. Class label = “sulphur-crested cockatoo” (89).

Figure 23. Uncurated 256×256 generation results of SiT-XL with SoftVQ-L 64 tokens. We use CFG with 4.0. Class label = “ice cream” (928).

Figure 24. Uncurated 256×256 generation results of SiT-XL with SoftVQ-L 64 tokens. We use CFG with 4.0. Class label = “cheeseburger” (933).

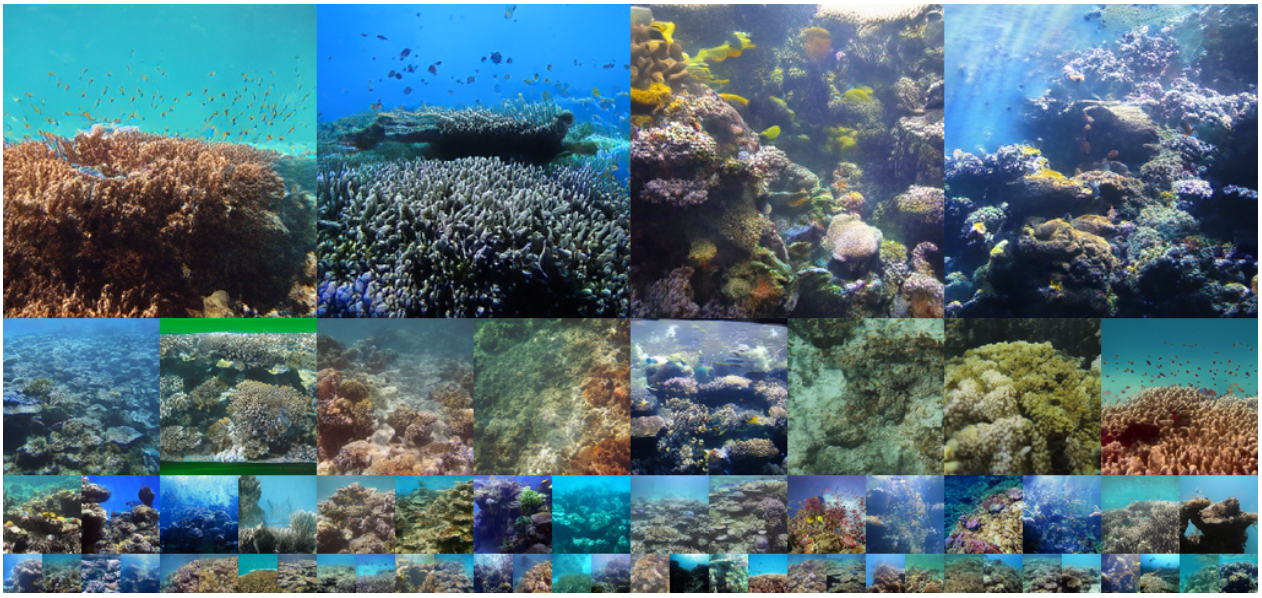

Figure 25. Uncurated 256×256 generation results of SiT-XL with SoftVQ-L 32 tokens. We use CFG with 4.0. Class label = “coral reef” (973).

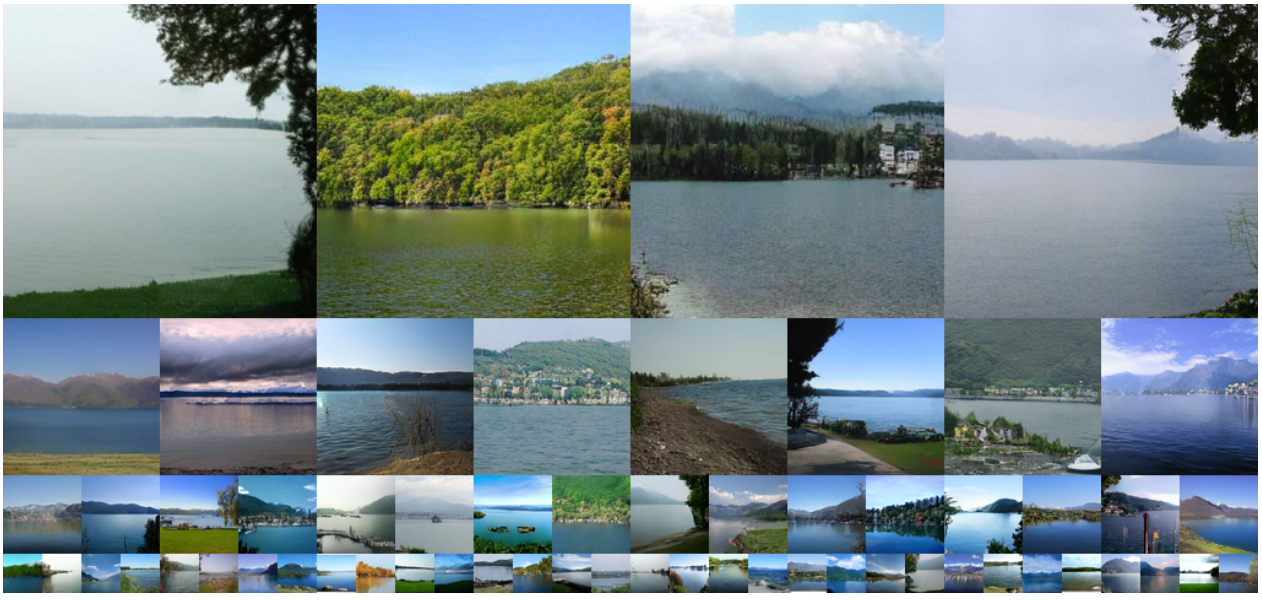

Figure 26. Uncurated 256×256 generation results of SiT-XL with SoftVQ-L 32 tokens. We use CFG with 4.0. Class label = “lake shore” (975).

Figure 27. Uncurated 512×512 generation results of SiT-XL with SoftVQ-L 64 tokens. We use CFG with 2.0. Class label = “cliff drop-off” (972).

Figure 28. Uncurated 512×512 generation results of SiT-XL with SoftVQ-L 64 tokens. We use CFG with 2.0. Class label = “volcano” (980).

Table 10. Comparison between TiTok and SoftVQ.

Tokenizer	# Param.	rFID (IN-1K)	rFID (COCO)	LP (IN-1K)	gFID	IS
TiTok-B-64	176	1.70	11.20	39.2	25.12	53.1
TiTok-BL-KL-64	389	1.25	8.94	11.4	23.35	52.7
TiTok-BL-VQ-64	390	2.60	11.24	9.3	19.23	61.8
SoftVQ-B-64	**173**	**0.89**	**5.16**	42.3	**10.13**	**103.4**
TiTok-L-32	614	2.21	14.41	48.9	26.84	49.5
TiTok-LL-KL-32	614	1.61	10.45	12.3	24.65	51.97
SoftVQ-L-32	**608**	**0.61**	**5.51**	59.4	**9.47**	**107.2**