Improving Flexible Image Tokenizers for Autoregressive Image Generation

Zixuan Fu[1] **Lanqing Guo**[2] **Chong Wang**[1] **Binbin Song**[3] **Ding Liu**[4] **Bihan Wen**[1*]
[1]Nanyang Technological University [2]The University of Texas at Austin
[3]Harbin Institute of Technology [4]Meta AI
{zixuan006,wang1711,bihan.wen}@ntu.edu.sg, lanqing.guo@austin.utexas.edu
binbinsong@hit.edu.cn, Liudingdavy@gmail.com

Abstract

Flexible image tokenizers aim to represent an image using an ordered 1D variable-length token sequence. This flexible tokenization is typically achieved through nested dropout, where a portion of trailing tokens is randomly truncated during training, and the image is reconstructed using the remaining preceding sequence. However, this tail-truncation strategy inherently concentrates the image information in the early tokens, limiting the effectiveness of downstream AutoRegressive (AR) image generation as the token length increases. To overcome these limitations, we propose **ReTok**, a flexible tokenizer with Redundant Token Padding and Hierarchical Semantic Regularization, designed to fully exploit all tokens for enhanced latent modeling. Specifically, we introduce **Redundant Token Padding** to activate tail tokens more frequently, thereby alleviating information over-concentration in early tokens. In addition, we apply **Hierarchical Semantic Regularization** to align the decoding features of earlier tokens with those from a pre-trained vision foundation model, while progressively reducing the regularization strength toward the tail to allow finer low-level detail reconstruction. Extensive experiments demonstrate the effectiveness of ReTok: on ImageNet 256×256, our method achieves superior generation performance compared with both flexible and fixed-length tokenizers. Code is available at: https://github.com/zfu006/ReTok

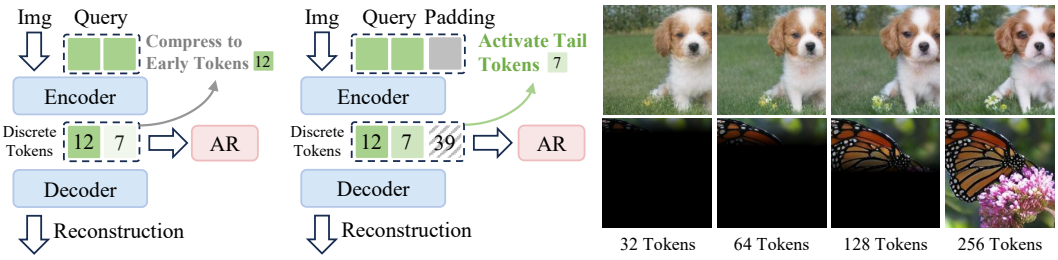
1 Introduction

Autoregressive (AR) models have demonstrated remarkable capability in image generation [12, 43, 45, 49, 60], owing to their inherent flexibility, scalability [16, 9, 3, 37], and potential for extension into unified multimodal frameworks [8, 44, 55, 50]. Typically, AR image generators rely on a visual tokenizer to compress images from the pixel space into a compact, discrete latent space, where the image distribution is modeled through next-token prediction. As a result, the visual tokenizer is crucial for downstream AR modeling, and advances in tokenizer design have greatly boosted generation quality [56, 60, 47, 12, 5, 53, 61].

Despite this progress, most visual tokenizers encode images into fixed-grid 1D or 2D representations, which fail to capture the heterogeneous complexity of natural images and constrain the flexibility of AR models. To overcome these limitations, **flexible tokenizers** are proposed to represent an image as a 1D variable-length token sequence [1, 31, 11, 57, 19, 26, 33, 54]. During training, these tokenizers adopt nested dropout [38], where trailing tokens are randomly truncated, and the image is reconstructed using the remaining prefix tokens. This enables AR models to generate variable-length token sequences that can be decoded into plausible images via the tokenizer's decoder. However, despite their flexibility, we observe that AR models built upon such tokenizers fail to

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) Previous approach *(b) ReTok (Ours)* *(c) ReTok (top) vs. Fixed-grid tokenizer (bottom)*

Figure 1: Overview of our method. (a) Previous flexible tokenizers (e.g., FlexTok [1] and One-D-Piece [31]) with naive nested dropout compress information into early tokens, compromising generation quality. (b) Our method with redundant token padding activates tail tokens, consistently improving generation quality as the sequence extends. (c) Comparison of ReTok with fixed-length tokenizers [43] in Autoregressive (AR) generation. Unlike fixed-length tokenizers that require generating a fixed-length sequence before decoding, ReTok allows AR models to produce variable-length sequences.

consistently improve generation quality as the sequence length increases. Moreover, their generation performance remains significantly lower than that of AR models trained with fixed-length tokenizers under the same token budget. As illustrated in Figure 2, the generation quality (measured by gFID) shows negligible improvement even when increasing the token count (e.g., from 128 to 256 tokens), revealing a fundamental bottleneck in current flexible tokenization strategies. In this paper, we propose **ReTok**, a 1D flexible tokenizer equipped with two novel training strategies that effectively address the limitations of existing flexible tokenizers. Specifically, we introduce **redundant token padding**, which appends additional tokens to the sequence tail to increase the activation frequency of trailing tokens during nested dropout, thereby promoting more balanced information distribution across the sequence. We further employ **hierarchical semantic regularization**, which aligns the decoding features of earlier tokens with high-level semantic representations from a pre-trained vision foundation model, e.g., DINOv2 [32], while progressively decaying the constraint toward the later tokens to enable the reconstruction of fine-grained visual details. Together, these designs substantially alleviate the generation bottleneck of flexible tokenizers and enhance the downstream AR image generation performance, advancing the practical adoption of flexible tokenization.

Our main contributions are summarized as follows:

- We propose **ReTok**, a novel 1D visual tokenizer that significantly mitigates the generation bottleneck of existing flexible tokenizers and enables high-quality AR generation.
- We introduce **redundant token padding** and **hierarchical semantic regularization**, which allow the tokenizer to fully exploit every token in the sequence and achieve consistent improvements in generation quality with longer token lengths.
- We conduct extensive experiments demonstrating that ReTok achieves superior generation performance among flexible tokenizers and attains comparable quality to state-of-the-art fixed-length 1D tokenizers.

2 Related Work

2.1 Image Tokenizer

Fixed-grid 1D and 2D Image Tokenizers. The objective of image tokenizers is to compress images into a compact latent space, which can then be modeled using generative models. For autoregressive image modeling, discrete tokenizers are widely adopted. VQVAE [47] first introduces vector quantization for discrete image modeling, while VQGAN [12] further improves the perceptual quality of reconstructed images by incorporating the adversarial and perceptual losses [15, 21]. Recent works have further advanced the development of discrete tokenizers through various improvements, such as replacing convolutional architectures with Vision Transformers (ViTs) [59, 4], scaling up the codebook size [66, 35, 27], refining quantization strategies [30, 66, 64, 60, 53], and introducing multi-scale residual quantization [45, 17], among others.

While previous tokenizers encode images into a 2D grid with spatial structures, recent 1D tokenizers further eliminate this inductive bias by encoding images into a 1D sequence [6, 7, 61, 56, 65]. TiTok [61], a ViT-based 1D tokenizer, initializes query tokens at the encoder. These query tokens and image patch embeddings are then jointly fed into the ViT encoder to learn the image latent representation. After that, quantization is applied to the query tokens, which are subsequently concatenated with the 2D mask tokens for reconstructing images at the decoder. The advantage of TiTok is its flexibility, as the number of query tokens can be controlled to balance the compression ratio and generation quality. GigaTok [56] further improves the 1D tokenizer by scaling the model size with representation alignment [62], which significantly enhances the generation quality of downstream AR models.

Flexible Tokenizers. In contrast to fixed-grid tokenizers, flexible tokenizers aim to encode images with variable-length token sequences [31, 1, 57, 19, 26, 11, 51, 13, 33, 54]. FlexTok [1] applies nested dropout [38] during tokenizer training, achieving coarse-to-fine image reconstruction from 1 to 256 tokens. However, FlexTok achieves the best gFID at 32 tokens, and generation quality decreases as more tokens are generated, indicating the under-utilization of tail tokens. Meanwhile, One-D-Piece [31] adopts a similar dropout training strategy. However, it focuses on image reconstruction and lacks the analysis of the downstream autoregressive image generation. Instead of representing the image at the original resolution with variable-length tokens, SpectralAR [19] and DetailFlow [26] introduce hierarchical reconstruction, where the early tokens reconstruct either low-resolution or low-frequency components of the image, while tail tokens complement fine-grained details. Consequently, using generated early tokens can only reconstruct low-resolution and blurry images, which undermines the flexibility of the AR models. Other flexible tokenizers, such as ViLex [51], are designed for diffusion models, while ElasticTok [57] and ALIT [11] mainly evaluate for image reconstruction.

2.2 Representation Alignment for Generation

Representation alignment [62] is initially designed for diffusion models [34, 28, 48, 14], by aligning the diffusion models' intermediate features with vision foundation model features [32, 46]. Recently, VA-VAE [58], GigaTok [56], MAETok [6], and ImageFolder [25] also inject semantic guidance from vision foundation models to regularize tokenizer training. Representation alignment significantly improves convergence speed and generation quality of the downstream diffusion and AR models.

3 Method

In this section, we first introduce the background of 1D and flexible tokenizers. We then discuss the generation bottleneck for flexible tokenizers using naive nested dropout. Finally, we present ReTok, which incorporates proposed redundant token padding and hierarchical semantic regularization.

3.1 Preliminary

1D Discrete Tokenizers. 1D discrete tokenizers, such as TiTok [61], aim to compress the image into the compact 1D sequence instead of the 2D grid. Given an image $\mathbf{X} \in \mathbb{R}^{H \times W \times 3}$, it is first patchified or fed into convolutional layers to obtain its patch embeddings $\mathbf{P} \in \mathbb{R}^{\frac{H}{f} \times \frac{W}{f} \times D}$, where f denotes the downsampling ratio and D is the embedding dimension. To construct the 1D image representations, a set of N learnable query tokens $\mathbf{Q} \in \mathbb{R}^{N \times D}$ is initialized. A ViT encoder [10] then takes as input the concatenation of the query tokens $\mathbf{Q}$ and image embeddings $\mathbf{P}$. The encoder only outputs query token embeddings, which are then quantized into the discrete tokens $\mathbf{Z} \in \mathbb{R}^{N \times d}$ using a quantizer:

$$\mathbf{Z} = \mathcal{Q}\big(\mathcal{E}\big([\mathbf{P};\, \mathbf{Q}]\big)\big)\,, \tag{1}$$

where $\mathcal{E}$ and $\mathcal{Q}$ are the encoder and quantizer, and $[\,\cdot;\cdot\,]$ denotes concatenation. For reconstruction, learnable mask tokens $\mathbf{M} \in \mathbb{R}^{\frac{H}{f} \times \frac{W}{f} \times D}$ and quantized query tokens are fed to a ViT decoder $\mathcal{D}$ to recover the 2D image: $\hat{\mathbf{X}} = \mathcal{D}([\mathrm{MLP}(\mathbf{Z});\, \mathbf{M}])$. $\mathbf{Z}$ is projected to embedding dimension D by MLP.

Flexible Tokenizers with Nested Dropout. Previous works [1, 31, 26, 57, 19] have proposed to apply nested dropout to train flexible tokenizers. During training, the tail tokens in the quantized token sequence $\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_N]$ are randomly dropped, resulting in a truncated token sequence:

$$\mathbf{Z}' = [\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_k], \tag{2}$$

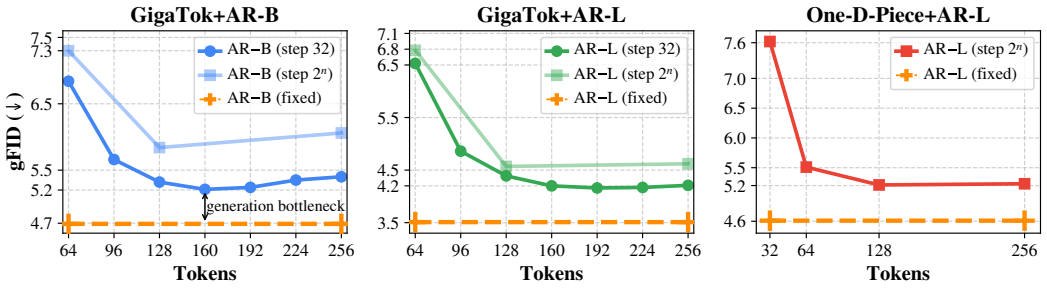


Figure 2: Illustration of the generation bottleneck in flexible tokenizers. We train AR models and evaluate generation quality at varying token lengths. Results indicate a significant decrease in downstream generation compared to fixed-length tokenizers. Details are in Section 3.2.

where $k \leq N$ denotes the number of tokens that are retained, while $N - k$ tail tokens are masked out. In FlexTok [1] and One-D-Piece [31], k is randomly sampled from $\{1, 2, 4, 8, ..., N\}$, while in DetailFlow [26], k is sampled from $\{8, 16, 24, ..., N\}$ with an interval of 8. The decoder aims to reconstruct the original image by using the truncated token sequence:

$$\hat{\mathbf{X}} = \mathcal{D}([\text{MLP}(\mathbf{Z}'); \mathbf{M}]). \tag{3}$$

Using nested dropout, the tokenizer learns to encode images as ordered 1D coarse-to-fine sequences.

Training Tokenizers with Semantic Regularization. Existing works propose training visual tokenizers along with representation alignment [56, 58, 62, 25, 6] to improve the downstream generation performance. We follow GigaTok [56], which applies the semantic regularization to align the tokenizer's decoder features with DINOv2 [32] image features from the same image:

$$\mathcal{L}_{reg} = -\cos(\text{MLP}(f_l^{\text{dec}}), f^{\text{DINO}}), \tag{4}$$

where f_l^{dec} denotes intermediate features from the tokenizer's l-th ViT decoder layer, f^{DINO} represents semantic features from the pre-trained DINOv2-B encoder on the same input image, and MLP projects f_l^{dec} to match the channel dimension of f^{DINO}. The full training objective of the tokenizer is the combination of the image reconstruction loss $\mathcal{L}_{rec}$ and the semantic regularization loss $\mathcal{L}_{reg}$:

$$\mathcal{L}_{total} = \mathcal{L}_{rec} + \lambda \mathcal{L}_{reg}. \tag{5}$$

Here, we follow the reconstruction loss defined in VQGAN [12], containing pixel-level reconstruction loss, perceptual loss [21], adversarial loss [15, 20], and VQ codebook loss.

3.2 Generation Bottleneck for Flexible Tokenizers

Even though there exist a few flexible tokenizers, limited work systematically analyzes their downstream autoregressive image generation performance, especially compared to the fixed-length counterparts. We conduct our preliminary experiments based on GigaTok [56], a fixed-length 1D image tokenizer that achieves state-of-the-art image generation. Following GigaTok, the token length N is set to 256, while we fine-tune GigaTok with 50 epochs by using the naive nested dropout to obtain its flexible versions. We conduct with two tokenizers, denoted as tokenizer (step 2^n) and tokenizer (step 32), where the retained tokens k are randomly selected from $\{32, 64, 128, 256\}$ and $\{32, 64, 96, ..., 256\}$ with an interval of 32, respectively. We then train the corresponding AR models[2] on ImageNet for 120 epochs and evaluates the generation FID (gFID) on its validation set. Moreover, we evaluate One-D-Piece [31], a pre-trained flexible tokenizer, and additionally train a version without nested dropout to serve as a fixed-length counterpart for comparison. Our findings are as follows:

Finding 1. **Image generation quality decreases when flexible tokenizers are trained with the naive nested dropout.** We highlight the generation bottleneck of flexible tokenizers - AR models trained under flexible tokenizers perform worse than those trained on fixed-length tokenizers. In Figure 2, we illustrate the results of different AR models under various token lengths. Compared to

[2]The AR model we utilize for evaluation is the LlamaGen-B (111M) and LlamaGen-L (343M) [43]. GigaTok-S-S serves as the baseline tokenizer.

the fixed-length baseline, all AR models with different parameters exhibit a significant decrease in generation quality. At the length 256, AR-B (step 32) and AR-B (step 2^n) with GigaTok achieve 5.4 and 6.1 in gFID, respectively, whereas the fixed-length AR model reaches 4.69[3]. This performance drop underscores the limitations of naive nested dropout.

Finding 2. **Generating more tokens in the tail won't improve generation quality.** As shown in Figure 2, generating more tokens does not guarantee better quality and can even lead to degraded results; e.g., gFID of 256 tokens is worse than that at 160 tokens for AR-B (step 32) with GigaTok. Similar phenomena have been observed in AR-L. We attribute this to naive nested dropout, where the tokenizer compresses most image information into early tokens. This training strategy fails to exploit the tail tokens and impedes the further improvement of generation quality with increasing tokens.

3.3 ReTok

ReTok follows GigaTok's architecture [56]. Image patch embeddings $\mathbf{P}$ are obtained via a stack of convolution layers. The ViT encoder and ViT decoder then learn the compact discrete 1D image representation. Finally, embeddings are mapped back to the pixel space by convolution layers for image reconstruction. In the following part, we introduce key improvements for flexible tokenizers.

Redundant Token Padding. Training flexible tokenizers with naive nested dropout compresses most image information into early tokens, which incurs the generation bottleneck for autoregressive models. To overcome this problem, we propose redundant token padding by concatenating additional query tokens at the tail of the original sequence:

$$[\mathbf{Z}; \mathbf{Z}_{pad}] = \mathcal{Q}(\mathcal{E}([\mathbf{P}; \mathbf{Q}; \mathbf{Q}_{pad}])), \tag{6}$$

where $\mathbf{Q}_{pad} \in \mathbb{R}^{M \times D}$ denotes the M padding tokens and $\mathbf{Z}_{pad}$ is the corresponding discrete tokens. The full token sequence $\mathbf{Z}_{full}$ is described as:

$$[\mathbf{Z}; \mathbf{Z}_{pad}] = [\mathbf{z}_1, ..., \mathbf{z}_N, \mathbf{z}_{N+1}, ..., \mathbf{z}_{N+M}]. \tag{7}$$

We further perform nested dropout on the concatenated token sequence, where the number of retained tokens satisfies $k \leq N + M$. The truncated token sequence is fed to the decoder for reconstruction following Equation (3). During the tokenizer training, the original token sequence $\mathbf{Z}$ becomes the "early" tokens in the current full token sequence, where the tokenizer aims to compress the most image information into it. This training strategy activates the tail tokens in the original sequence. Since the original token sequence $\mathbf{Z}$ learns to represent the image during training, we solely use it and discard the encoded padding tokens $\mathbf{Z}_{pad}$ for downstream autoregressive generation. We illustrate the overview of our redundant token padding in Figure 1.

Hierarchical Semantic Regularization. Previous works [2] have shown that 1D tokenizers with high compression ratios (e.g., TiTok [61] with 32 tokens) effectively learn semantic and high-level image representations. Inspired by their work, we propose hierarchical semantic regularization to enhance the semantic representation of early tokens while enabling the tail tokens for pixel-level reconstruction. We follow the training objective defined in Equation (5), but make the regularization weight λ a function of sequence length k:

$$\mathcal{L}_{total} = \mathcal{L}_{rec} + \lambda(k)\mathcal{L}_{reg}, \tag{8}$$

where $\lambda(k)$ decreases linearly as the retained sequence length k increases. With Equation (8), early tokens emphasize feature-level alignment with the semantic feature of DINOv2, while progressively enabling the subsequent tokens to complement the low-level structures and details of the image.

Decoder Fine-tuning. Since early tokens are constrained with high semantic regularization during training, we further fine-tune the decoder to improve the reconstruction performance of early tokens. We freeze the well-trained encoder and the quantizer, while easing the semantic constraint by setting $\lambda(k)$ to a small constant for all sequence lengths. As we show in experiments, fine-tuning the decoder improves the quality of generated images for short token sequences (e.g., 32 or 64 tokens).

3.4 Application - Efficient Test-Time Scaling in AR Generation

Since ReTok can decode images with varying lengths and consistently improves their quality as more tokens are generated, it naturally enables **efficient test-time scaling** (TTS) for **AR image generation**.

[3]We search the optimal CFG for AR models at the full length (256 tokens) and evaluate the gFID for the shorter length under the same CFG.

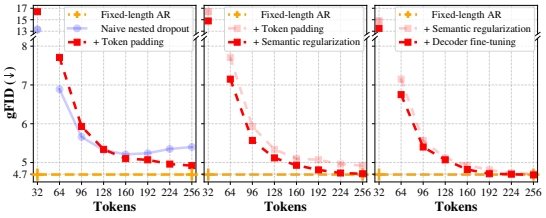
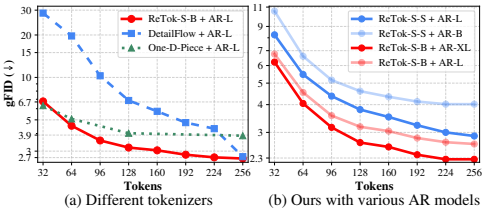


Figure 3: Ablation study on the improvement of ReTok. We evaluate the effectiveness of our proposed methods (redundant token padding, hierarchical semantic regularization, and decoder fine-tuning) on the ImageNet validation set.



(a) Different tokenizers (b) Ours with various AR models

Figure 4: Analysis of image generation at different token lengths. (a) Downstream image generation comparison of ReTok, DetailFlow [26], and One-D-Piece [31]. (b) Generation performance of ReTok with various AR models.

For example, in a simple Best-of-N (BoN) setting, early tokens in ReTok already produce images with decent quality and preserve the global structure of the final image. Therefore, we can generate N candidates using short prefixes (e.g., 64 tokens), select the best one with a verifier, and continue decoding the selected candidate, rather than generating N full 256-token candidates, thereby reducing inference-time computation. Detailed discussion is provided in Section 4.4.

4 Experiments

4.1 Experiment Settings

Implementation Details. Following previous works [1, 26, 31, 56], we set ReTok's original query token length N to 256. The tokenizer can encode an image from 32 to 256 tokens, with step size 32. We further extend the sequence length up to 480 by padding with 224 redundant query tokens, which enables the number of retained tokens k to be randomly selected from $\{32, 64, ..., 256, 288, ..., 480\}$ when applying nested dropout during tokenizer training. This keeps the "average" retained sequence length around 256. Following the architecture design of GigaTok [56], we train two versions of the tokenizer, ReTok-S-S (136M) and ReTok-S-B (232M), with a codebook size of 16384. Unless otherwise specified, we initialize our tokenizers with pre-trained GigaTok weights. We train the ReTok-S-S for 200 epochs and ReTok-S-B for 250 epochs, while all decoders of tokenizers are further fine-tuned for 50 epochs. The weight of semantic regularization $\lambda(k)$ for ReTok-S-S and ReTok-S-B in Equation (8) decreases linearly from 2.0 and 2.5 to 0.5, respectively, for sequence lengths between 32 and 256, and is set to 0.5 for lengths beyond 256. For fine-tuning the decoder, we fix the semantic constraint to 0.5.

For downstream image generation, we discard the redundant discrete tokens and keep the token length to 256. Our autoregressive models are based on LlamaGen [43]. For the ReTok-S-S and ReTok-S-B, we train the LlamaGen-B (111M)/LlamaGen-L (343M) and LlamaGen-L (343M)/LlamaGen-XL (775M) variants, respectively. The AR models in our main experiments are trained for 300 epochs, following the training recipes defined in [43]. During inference, a step-function Classifier-Free Guidance (CFG) schedule is employed, where the first 18% of tokens are generated without CFG (CFG = 1) to enhance diversity, and the remaining tokens use CFG to improve generation quality. We search for the optimal CFG for each AR model during evaluation. All tokenizers and AR models are trained on ImageNet [40] with images of size 256×256, and evaluated on the ImageNet validation set.

Metrics. We apply Fréchet Inception Distance (FID) [18], Inception Score (IS) [41], Precision and Recall [22] to evaluate the image generation performance. We also report Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM) [52], Learned Perceptual Image Patch Similarity (LPIPS) [63], and reconstruction FID (rFID) results for assessing the tokenizer's reconstruction performance.

4.2 Roadmap to Improve ReTok

We evaluate each part of the proposed method. (1) We start from GigaTok-S-S and train the fixed-length AR model (111M) for 120 epochs. The AR model achieves a gFID of 4.69, serving as our

Table 1: Comparison of class-conditional image generation on ImageNet 256×256. † denotes the model generates images at 336×336 resolution, and are resized to 256×256 for evaluation. $\star$ indicates models that are our implementation. We report the optimal gFID achieved across scenarios with and without Classifier-Free Guidance. **Bold** and underline indicate the first and second best methods within flexible tokenizers.

Type	Tokenizer	Param.	rFID↓	Generator	Param.	Type	# Tokens	gFID↓	gIS↑	Precision↑	Recall↑
Continuous modeling											
				DiT-XL/2 [34]	675M	Diff	1024	2.27	278.2	0.83	0.57
2D	SD-VAE [39]	84M	0.62	SiT-XL/2 [28]	675M	Diff	1024	2.06	270.3	0.82	0.59
				SiT-XL/2+REPA [62]	675M	Diff	1024	1.42	305.7	0.80	0.65
2D	VA-VAE [58]	70M	0.28	LightningDiT [58]	675M	Diff	256	1.35	295.3	0.79	0.65
2D	VAE [24]	66M	0.53	MAR-B [24]	208M	AR+Diff	256	2.31	281.7	0.82	0.57
Discrete modeling											
2D	VQGAN [5]	66M	2.28	MaskGIT [5]	227M	Mask	256	6.18	182.1	0.80	0.51
2D	VAR [45]	109M	0.90	VAR-d16 [45]	310M	VAR	680	3.30	274.1	0.84	0.51
				VAR-d20 [45]	600M	VAR	680	2.57	302.6	0.83	0.56
2D	ImageFolder [25]	176M	0.80	ImageFolder-VAR [25]	362M	VAR	286	2.60	295.0	0.75	0.63
2D	LlamaGen [43]	72M	2.19	LlamaGen-L [43]	343M	AR	256	3.81	248.3	0.83	0.52
1D	VFMTok† [65]	–	0.89	LlamaGen-L† [43]	343M	AR	256	2.11	230.1	0.82	0.60
1D	VFMTok [65]	–	1.02	LlamaGen-L [43]	343M	AR	256	2.79	276.0	–	–
1D	TiTok-S [61]	72M	1.71	MaskGIT-UViT-L [5]	287M	Mask	128	1.87	281.8	–	–
1D	TiTok-L [61]	641M	2.21	MaskGIT-ViT [5]	177M	Mask	32	2.77	199.8	–	–
1D	GigaTok-S-S [56]	136M	1.01	LlamaGen-B [43]	111M	AR	256	4.05	240.6	0.81	0.51
				LlamaGen-L⋆ [43]	343M	AR	256	2.86	261.2	0.81	0.57
1D	GigaTok-S-B [56]	232M	0.89	LlamaGen-L⋆ [43]	343M	AR	256	2.71	246.3	0.81	0.58
Flex	FlexTok [1]	~ 2.5B	1.08	LlamaGen [1]	1.33B	AR+Diff	32	**1.86**	–	–	–
							256	~ 2.5	–	–	–
Flex	One-D-Piece [31]	641M	1.08	LlamaGen-B⋆ [43]	86M	AR	256	6.49	194.3	**0.82**	0.43
				LlamaGen-L⋆ [43]	318M	AR	256	3.86	231.7	0.81	0.51
Flex	DetailFlow-32 [26]	270M	0.80	LlamaGen-L [43]	326M	AR	256	2.75	250.8	0.81	0.58
Flex	DetailFlow-64 [26]	270M	0.55	LlamaGen-L [43]	326M	AR	512	2.62	245.3	0.80	**0.60**
Flex	SpectralAR [19]	–	4.03	SpectralAR-d16 [19]	310M	AR	64	3.02	282.2	0.81	0.55
				SpectralAR-d20 [19]	600M	AR	64	2.49	**305.4**	–	–
Flex	**ReTok-S-S**	136M	1.09	LlamaGen-B [43]	111M	AR	256	4.02	245.2	0.80	0.50
				LlamaGen-L [43]	343M	AR	256	2.92	243.5	0.81	0.57
Flex	**ReTok-S-B**	232M	1.01	LlamaGen-L [43]	343M	AR	256	2.66	231.7	**0.82**	0.57
				LlamaGen-XL [43]	775M	AR	256	2.27	245.9	**0.82**	**0.60**

reference. (2) Building upon this baseline, we train the flexible tokenizer with naive nested dropout for 50 epochs, where the retained sequence length k is selected from $\{32, 64, 96, ..., 256\}$. The downstream AR model uses the same configuration as above. We search for the optimal CFG at full length (e.g., 256 tokens) and apply it to shorter sequences. The training objective follows Equation (5) for all sequence lengths with $\lambda = 0.5$. (3) In Figure 3 left, the AR model achieves the best gFID of 5.21 at 160 tokens and a worse gFID of 5.40 at 256 tokens, showing a significant decrease compared to the fixed-length counterpart. (4) We then adopt the proposed redundant token padding strategy to train the flexible tokenizer. Figure 3 left shows that the downstream AR model achieves the best gFID of 4.92 at 256 tokens. (5) However, generation quality decreases at early tokens. We therefore propose hierarchical semantic regularization to enhance their semantic representation. This technique improves gFID of 32 tokens by 1.61 while attaining a gFID of 4.71 at 256 tokens, comparable to the baseline. (6) Finally, we relax the semantic constraint and fine-tune the decoder for better image reconstruction. Figure 3 right shows that decoder fine-tuning refines image generation at short token lengths while maintaining quality for long sequences.

4.3 Main Results

Class-conditional Image Generation. We first evaluate the image generation performance at the full length on ImageNet. As shown in Table 1, our ReTok achieves performance comparable to that of our GigaTok baseline [56], demonstrating its effectiveness in addressing the generation bottleneck for flexible tokenizers. Compared to other flexible tokenizers, our method presents a significant gain over One-D-Piece [31] on downstream image generation. Moreover, ReTok outperforms DetailFlow [26] under similar parameters. Figure 5 shows several generated images by using 256 tokens.

We further compare the generation results of different flexible tokenizers using the same LlamaGen-L across various token lengths. For DetailFlow [26], since it generates images at varying resolutions for different token lengths, we resized all low-resolution outputs to 256×256 before measuring gFID. Figure 4 (a) presents that our method outperforms the One-D-Piece across all token lengths except at 32 tokens. Notably, One-D-Piece shows negligible gains from 128 to 256 tokens. In contrast,

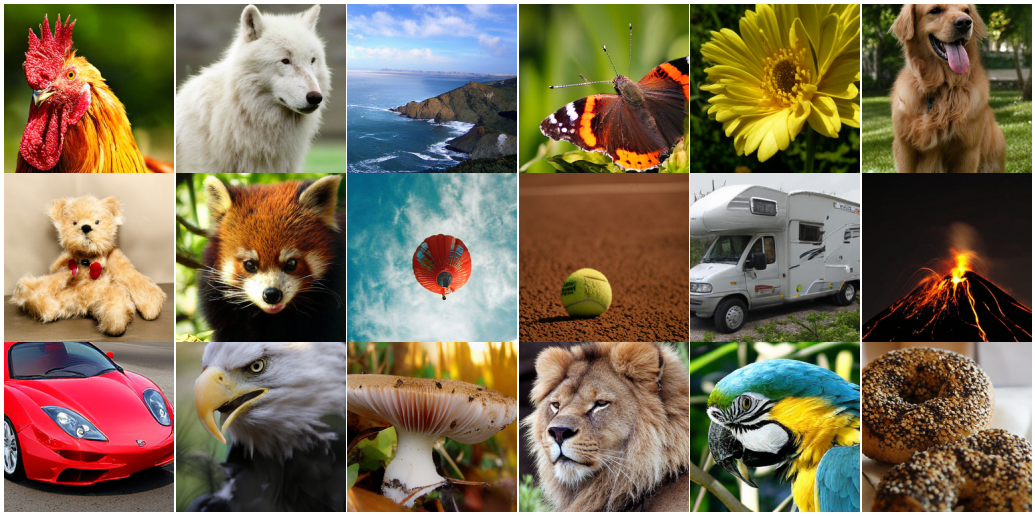


Figure 5: Examples of generated images on ImageNet 256×256 from the ReTok-S-B with LlamaGen-XL models using 256 tokens. The classifier-free guidance is set to 4.0.

Table 2: Ablation of semantic regularization. The hierarchical regularization improves generation quality while maintaining decent reconstruction.

Sem. Reg. $\lambda(k)$	Tokens	rFID↓	LPIPS↓	gFID↓
0.5	32	9.92	0.411	16.40
	256	1.15	0.232	4.92
2.0–0.5	32	9.02	0.415	14.79
	256	1.18	0.234	4.68
2.0	256	1.24	0.241	4.72

Table 3: Quantitative results of Test-Time Scaling (TTS) on ReTok-S-S. We compare the generation performance without TTS (search size = 1) and with TTS (search size = 4). The CFG is set to 1 during the generation process.

Method	Search size	FID↓	IS↑	Precision↑	Recall↑	CLIP score↑
AR-B	1	8.46	105.6	0.74	0.59	28.30
AR-B	4	4.69	194.0	0.83	0.51	29.59
AR-L	1	3.99	159.3	0.77	0.61	28.84
AR-L	4	3.67	280.4	0.85	0.53	29.91

our method improves the gFID from 3.18 to 2.66 between 128 and 256 tokens. We also evaluate the generation quality of our method with different AR models in Figure 4 (b). The results reveal consistent improvements with the increasing number of tokens.

Image Reconstruction. We demonstrate the image reconstruction quality of our method in Table 1, where our ReTok-S-S and ReTok-S-B achieve rFID scores of 1.09 and 1.01 with 256 tokens, respectively. Our tokenizers slightly decrease compared to GigaTok, which we attribute to the use of nested dropout. Nevertheless, ReTok-S-B outperforms methods One-D-Piece and FlexTok, demonstrating its overall effectiveness in reconstruction. Detailed reconstruction results can be found in the appendix.

4.4 Results of Test-Time Scaling

To verify test-time scaling (TTS) in ReTok, we conduct experiments on AR-B and AR-L models trained with ReTok-S-S (CFG=1). Following [29], we implement Best-of-N (BoN) selection. We generate 64 early tokens and reconstruct them for verification. To select the best candidate, we use the pre-trained linear classification head of DINOv2 [32] and choose the candidate with the highest logit for the target class. The selected image is used to decode subsequent tokens (see Figure 6).

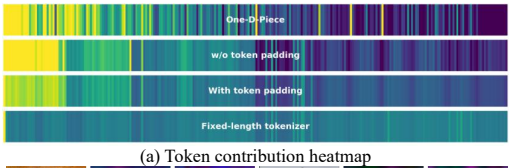
Experimental results in Table 3 show that TTS improves generation quality across FID, IS, and CLIP scores [36] (using prompt: *"a photo of {class label}"*) at search size $N = 4$. However, we find recall decreases, which suggests TTS trades sample diversity for semantic consistency, consistent with [29].

4.5 More Analysis and Ablation Study

Redundant Token Padding is the Key to Activate Tail Tokens. As illustrated in Figure 3, token padding improves overall generation quality as more tail tokens are generated. We further analyze token contribution following One-D-Piece [31]. We first reconstruct an image $\hat{\mathbf{X}}$ with the tokenizer,



Figure 6: Visualizing Test-Time Scaling (TTS) in AR generation. Candidates (64 tokens) are first generated and verified, where the selected image is used to further decode up to 256 tokens.



(a) Token contribution heatmap



(b) Visualization of latent features

Figure 7: Ablation study of ReTok. (a) Analysis of token contribution (S-S tokenizer). (b) Visualization of latent features from the tokenizer decoder at different token lengths.

Table 4: Design choices of ReTok (S-S tokenizer). (a) Effect of the number of padding tokens. The reconstruction and generation performance are evaluated on the full token (256) length. (b) Effect of the initial weight of the hierarchical semantic regularization. We vary the semantic weight $\lambda(32)$ while keeping $\lambda(256)$ at 0.5. (c) Impact of the step size of nested dropout.

Pad. Tokens	rFID↓	LPIPS↓	gFID↓
64	1.21	0.238	4.74
224	1.18	0.234	4.68
384	1.34	0.245	5.03

(a) Number of padding tokens

λ_{start}	rFID↓	LPIPS↓	gFID↓
5	1.27	0.242	4.93
2	1.18	0.234	4.68
1	1.13	0.232	4.75

(b) Initial weight λ_{start}

Step Size	Tokens	rFID↓	LPIPS↓	gFID↓
16	32	9.56	0.422	15.79
	256	1.22	0.234	4.71
32	32	9.02	0.415	14.79
	256	1.18	0.234	4.68

(c) Dropout step size

and measure a token's contribution by the L1 distance $\|\hat{\mathbf{X}} - \hat{\mathbf{X}}'\|$ between the original reconstruction and perturbed version $\hat{\mathbf{X}}'$, where token $\mathbf{z}_i$ at position i is replaced by a random token. We compute the mean L1 distance for each position i over the ImageNet validation set, then apply softmax to obtain a normalized contribution distribution. We visualize the token contribution heatmap in Figure 7 (a), comparing One-D-Piece, ReTok without token padding, ReTok with token padding, and GigaTok. As expected, both One-D-Piece and ReTok without token padding show highly concentrated contributions at head tokens (yellow), indicating their dominant role in reconstruction, while tail tokens contribute negligibly. In contrast, our method effectively activates tail tokens, yielding a more uniform contribution distribution across all tokens, similar to the fixed-length tokenizer.

Early Tokens Deserve High Semantic Regularization. We justify the proposed hierarchical semantic regularization in Table 2. We first compare our tokenizer trained with hierarchical semantic regularization against a baseline without it, where the baseline regularization weight is set to a low constant (0.5) for all tokens (see details in Section 4.2). Hierarchical semantic regularization significantly improves overall generation quality across all token lengths without compromising reconstruction performance. Furthermore, we consider an extreme case by setting a large fixed regularization weight of 2.0, identical to the weight applied at 32 tokens in the hierarchical version. However, the results show severe degradation in reconstruction and no improvement in generation.

Visualize Latent Features of Tokenizer. We visualize latent features from the first ViT decoder layer using PCA with 3 components. PCA is computed on one class and applied to reduce features to 3 dimensions for visualization. We compare feature maps at 32 and 256 tokens in Figure 7 (b). The results show that early tokens capture the global shape of the main object, while the full sequence adds detailed structure and texture.

Design and Training Choices for ReTok. We determine key design choices for ReTok. (1) Number of padding tokens. By default, we pad 224 redundant tokens. In Table 4 (a), we find padding more tokens (384) or fewer tokens (64) leads to worse reconstruction and generation results. (2) Initial weight of hierarchical semantic regularization. As shown in Table 4 (b), a large initial weight for 32 tokens degrades both generation and reconstruction of the tokenizer, while a small weight slightly leads to a performance drop on gFID. (3) Step size of nested dropout. We compare two settings for sampling k during nested dropout: step size 32 ($k \in \{32, 64, ..., 256\}$) and step size 16

Table 5: Additional design choices of ReTok. (a) Effect of tokenizer initialization. (b) Comparison of padding and prefix-length reweighting. We compare redundant token padding with no padding using $p = P(k = 256) \in \{0.3, 0.5\}$, with the remaining lengths sampled uniformly. (c) Effect of the representation-alignment teacher. Experiments in (a,b) use ReTok-S-S with AR-B trained for 120 epochs, while (c) follows the full ReTok-S-B training protocol with AR-L trained for 300 epochs.

Init.	Epochs	gFID↓
GigaTok	50	**4.68**
Scratch	250	4.82
Scratch	350	4.71

(a) Tokenizer initialization

Sampling	64	96	128	160	192	224	256
$p = 0.3$	**6.97**	**5.68**	5.23	5.10	4.92	4.99	5.03
$p = 0.5$	8.91	6.63	5.72	5.39	5.11	5.02	4.74
Padding	7.55	5.76	**5.12**	**4.93**	**4.81**	**4.73**	**4.68**

(b) Padding vs. reweighting

Teacher	gFID↓
DINOv2-B	**2.66**
DINOv2-L	2.74
DINOv3-L	2.82

(c) Teacher encoder

($k \in \{32, 48, ...256\}$). We observe in Table 4 (c) that using a shorter step size (step size 16) performs comparable performance at full length, while degrading both reconstruction and generation at 32 tokens. (4) GigaTok Initialization. Training from scratch for 350 epochs, including a 50-epoch warm-up without token dropout, achieves 4.71 gFID versus 4.68 with GigaTok initialization and 50 epochs of training (Table 5 (a)), showing that pre-trained initialization is not essential. We use pre-trained GigaTok to accelerate convergence. (5) Redundant token padding vs. reweighting. Without padding, we set $p = P(k = 256)$ to 0.3 or 0.5 and sample the remaining lengths uniformly. Increasing p improves full-length generation but degrades short-prefix generation quality; redundant token padding outperforms $p = 0.5$ at all reported lengths (Table 5 (b)). (6) Alignment teacher. Under the same full ReTok-S-B and AR-L training protocol, DINOv2-L and DINOv3-L [42] do not improve upon DINOv2-B (Table 5 (c)). We therefore use DINOv2-B as the teacher encoder.

5 Conclusion

In this paper, we systematically analyze the generation bottleneck of current flexible tokenizers when applying naive nested dropout. To address these issues, we present ReTok, a 1D visual tokenizer that allows flexible AR generation to achieve consistent improvement in generation as the token sequence extends. We propose redundant token padding, hierarchical semantic regularization, and decoder fine-tuning to exploit full sequence length for better latent modeling. We conduct extensive experiments to evaluate the effectiveness of our method. On ImageNet 256×256, our ReTok-S-B with AR-XL achieves 2.27 gFID, demonstrating its superior performance compared to other flexible and fixed-length tokenizers. Discussions of implementation details, visual examples, and limitations are presented in the appendix.

Acknowledgments and Disclosure of Funding

This work was supported by the Rapid-Rich Object Search (ROSE) Lab, School of Electrical and Electronic Engineering, Nanyang Technological University (NTU), Singapore.

References

[1] Roman Bachmann, Jesse Allardice, David Mizrahi, Enrico Fini, Oğuzhan Fatih Kar, Elmira Amirloo, Alaaeldin El-Nouby, Amir Zamir, and Afshin Dehghan. Flextok: Resampling images into 1d token sequences of flexible length. In *Forty-second International Conference on Machine Learning*, 2025.

[2] Lukas Lao Beyer, Tianhong Li, Xinlei Chen, Sertac Karaman, and Kaiming He. Highly compressed tokenizer can generate without training. In *Forty-second International Conference on Machine Learning*, 2025.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[4] Shiyue Cao, Yueqin Yin, Lianghua Huang, Yu Liu, Xin Zhao, Deli Zhao, and Kaigi Huang. Efficient-vqgan: Towards high-resolution image generation with efficient vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7368–7377, 2023.

[5] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11315–11325, 2022.

[6] Hao Chen, Yujin Han, Fangyi Chen, Xiang Li, Yidong Wang, Jindong Wang, Ze Wang, Zicheng Liu, Difan Zou, and Bhiksha Raj. Masked autoencoders are effective tokenizers for diffusion models. In *Forty-second International Conference on Machine Learning*, 2025.

[7] Hao Chen, Ze Wang, Xiang Li, Ximeng Sun, Fangyi Chen, Jiang Liu, Jindong Wang, Bhiksha Raj, Zicheng Liu, and Emad Barsoum. Softvq-vae: Efficient 1-dimensional continuous tokenizer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 28358–28370, 2025.

[8] Xiaokang Chen, Zhiyu Wu, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, and Chong Ruan. Janus-pro: Unified multimodal understanding and generation with data and model scaling. *arXiv preprint arXiv:2501.17811*, 2025.

[9] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations*, 2021.

[11] Shivam Duggal, Phillip Isola, Antonio Torralba, and William T. Freeman. Adaptive length image tokenization via recurrent allocation. In *The Thirteenth International Conference on Learning Representations*, 2025.

[12] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021.

[13] Carlos Esteves, Mohammed Suhail, and Ameesh Makadia. Spectral image tokenizer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17181–17190, 2025.

[14] Zixuan Fu, Chong Wang, Lanqing Guo, Kailai Zhou, Jiahao Nie, and Bihan Wen. A frozen pixel-space diffusion model can guide itself with its own samples. *arXiv preprint arXiv:2607.29122*, 2026.

[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[16] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[17] Jian Han, Jinlai Liu, Yi Jiang, Bin Yan, Yuqi Zhang, Zehuan Yuan, Bingyue Peng, and Xiaobing Liu. Infinity: Scaling bitwise autoregressive modeling for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15733–15744, 2025.

[18] 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. *Advances in neural information processing systems*, 30, 2017.

[19] Yuanhui Huang, Weiliang Chen, Wenzhao Zheng, Yueqi Duan, Jie Zhou, and Jiwen Lu. Spectralar: Spectral autoregressive visual generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15842–15852, 2025.

[20] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[21] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

[22] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *Advances in neural information processing systems*, 32, 2019.

[23] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023.

[24] Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. *Advances in Neural Information Processing Systems*, 37:56424–56445, 2024.

[25] Xiang Li, Kai Qiu, Hao Chen, Jason Kuen, Jiuxiang Gu, Bhiksha Raj, and Zhe Lin. Imagefolder: Autoregressive image generation with folded tokens. In *The Thirteenth International Conference on Learning Representations*, 2025.

[26] Yiheng Liu, Liao Qu, Huichao Zhang, Xu Wang, Yi Jiang, Yiming Gao, Hu Ye, Xian Li, Shuai Wang, Daniel K Du, et al. Detailflow: 1d coarse-to-fine autoregressive image generation via next-detail prediction. *arXiv preprint arXiv:2505.21473*, 2025.

[27] Chuofan Ma, Yi Jiang, Junfeng Wu, Jihan Yang, Xin Yu, Zehuan Yuan, Bingyue Peng, and Xiaojuan Qi. UniTok: A unified tokenizer for visual generation and understanding. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

[28] Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *European Conference on Computer Vision*, pages 23–40. Springer, 2024.

[29] Nanye Ma, Shangyuan Tong, Haolin Jia, Hexiang Hu, Yu-Chuan Su, Mingda Zhang, Xuan Yang, Yandong Li, Tommi Jaakkola, Xuhui Jia, et al. Inference-time scaling for diffusion models beyond scaling denoising steps. *arXiv preprint arXiv:2501.09732*, 2025.

[30] Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: VQ-VAE made simple. In *The Twelfth International Conference on Learning Representations*, 2024.

[31] Keita Miwa, Kento Sasaki, Hidehisa Arai, Tsubasa Takahashi, and Yu Yamaguchi. One-d-piece: Image tokenizer meets quality-controllable compression. *arXiv preprint arXiv:2501.10064*, 2025.

[32] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[33] Maitreya Patel, Jingtao Li, Weiming Zhuang, Yezhou Yang, and Lingjuan Lv. Vibetoken: Scaling 1d image tokenizers and autoregressive models for dynamic resolution generations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2058–2068, 2026.

[34] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[35] Liao Qu, Huichao Zhang, Yiheng Liu, Xu Wang, Yi Jiang, Yiming Gao, Hu Ye, Daniel K. Du, Zehuan Yuan, and Xinglong Wu. Tokenflow: Unified image tokenizer for multimodal understanding and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2545–2555, 2025.

[36] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[37] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[38] Oren Rippel, Michael Gelbart, and Ryan Adams. Learning ordered representations with nested dropout. In *International Conference on Machine Learning*, pages 1746–1754. PMLR, 2014.

[39] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[40] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[41] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

[42] Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

[43] Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. Autoregressive model beats diffusion: Llama for scalable image generation. *arXiv preprint arXiv:2406.06525*, 2024.

[44] Chameleon Team. Chameleon: Mixed-modal early-fusion foundation models. *arXiv preprint arXiv:2405.09818*, 2024.

[45] Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *Advances in neural information processing systems*, 37:84839–84865, 2024.

[46] Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025.

[47] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[48] Chong Wang, Zixuan Fu, Shiqi Huang, Siyuan Yang, Hao Cheng, and Bihan Wen. Persistence forcing: Exploiting feature specialization in pixel-space diffusion. *arXiv preprint arXiv:2609.36014*, 2026.

[49] Junke Wang, Zhi Tian, Xun Wang, Xinyu Zhang, Weilin Huang, Zuxuan Wu, and Yu-Gang Jiang. Simplear: Pushing the frontier of autoregressive visual generation through pretraining, sft, and rl. *arXiv preprint arXiv:2504.11455*, 2025.

[50] Xinlong Wang, Xiaosong Zhang, Zhengxiong Luo, Quan Sun, Yufeng Cui, Jinsheng Wang, Fan Zhang, Yueze Wang, Zhen Li, Qiying Yu, et al. Emu3: Next-token prediction is all you need. *arXiv preprint arXiv:2409.18869*, 2024.

[51] XuDong Wang, Xingyi Zhou, Alireza Fathi, Trevor Darrell, and Cordelia Schmid. Visual lexicon: Rich image features in language space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19736–19747, 2025.

[52] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[53] Mark Weber, Lijun Yu, Qihang Yu, Xueqing Deng, Xiaohui Shen, Daniel Cremers, and Liang-Chieh Chen. Maskbit: Embedding-free image generation via bit tokens. *Trans. Mach. Learn. Res.*, 2024, 2024.

[54] Xin Wen, Bingchen Zhao, Ismail Elezi, Jiankang Deng, and Xiaojuan Qi. " principal components" enable a new language of images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16641–16651, 2025.

[55] Yecheng Wu, Zhuoyang Zhang, Junyu Chen, Haotian Tang, Dacheng Li, Yunhao Fang, Ligeng Zhu, Enze Xie, Hongxu Yin, Li Yi, Song Han, and Yao Lu. VILA-U: a unified foundation model integrating visual understanding and generation. In *The Thirteenth International Conference on Learning Representations*, 2025.

[56] Tianwei Xiong, Jun Hao Liew, Zilong Huang, Jiashi Feng, and Xihui Liu. Gigatok: Scaling visual tokenizers to 3 billion parameters for autoregressive image generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 18770–18780, 2025.

[57] Wilson Yan, Volodymyr Mnih, Aleksandra Faust, Matei Zaharia, Pieter Abbeel, and Hao Liu. Elastictok: Adaptive tokenization for image and video. In *The Thirteenth International Conference on Learning Representations*, 2025.

[58] Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15703–15712, 2025.

[59] Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. Vector-quantized image modeling with improved VQGAN. In *The Tenth International Conference on Learning Representations*, 2022.

[60] Lijun Yu, José Lezama, Nitesh Bharadwaj Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Agrim Gupta, Xiuye Gu, Alexander G. Hauptmann, Boqing Gong, Ming-Hsuan Yang, Irfan Essa, David A. Ross, and Lu Jiang. Language model beats diffusion - tokenizer is key to visual generation. In *The Twelfth International Conference on Learning Representations*, 2024.

[61] Qihang Yu, Mark Weber, Xueqing Deng, Xiaohui Shen, Daniel Cremers, and Liang-Chieh Chen. An image is worth 32 tokens for reconstruction and generation. *Advances in Neural Information Processing Systems*, 37:128940–128966, 2024.

[62] Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. In *The Thirteenth International Conference on Learning Representations*, 2025.

[63] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[64] Yue Zhao, Yuanjun Xiong, and Philipp Krähenbühl. Image and video tokenization with binary spherical quantization. In *The Thirteenth International Conference on Learning Representations*, 2025.

[65] Anlin Zheng, Xin Wen, Xuanyang Zhang, Chuofan Ma, Tiancai Wang, Gang Yu, Xiangyu Zhang, and Xiaojuan Qi. Vision foundation models as effective visual tokenizers for autoregressive generation. In *Advances in Neural Information Processing Systems*, 2025.

[66] Lei Zhu, Fangyun Wei, Yanye Lu, and Dong Chen. Scaling the codebook size of VQ-GAN to 100, 000 with a utilization rate of 99%. In *Advances in Neural Information Processing Systems*, 2024.

A Technical appendices and supplementary material

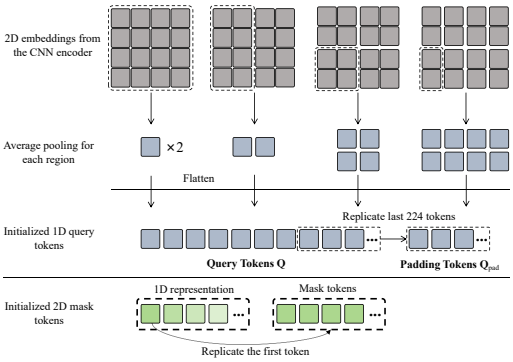


Figure 8: Illustration of initializing query tokens and mask tokens.

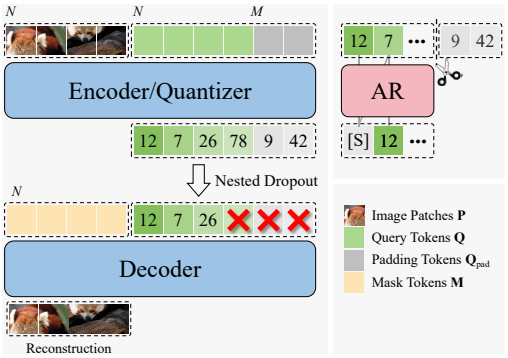


Figure 9: Illustration of ReTok.

A.1 Implementation Details of ReTok

We follow GigaTok [56], which adopts a hybrid architecture comprising a CNN encoder, a ViT-based Q-Former encoder [23], a ViT-based Q-Former decoder, and a CNN decoder. First, the CNN encoder takes the 256×256 image as input and downsamples it 16 times to obtain the image embedding $\mathbf{P}$ ($L = 256$). To extract 1D representations, we employ a Q-Former with 256 query tokens $\mathbf{Q}$, which are initialized by a multi-level average pooling strategy (see Figure 8). For padding tokens $\mathbf{Q}_{pad}$, we replicate the last 224 query tokens, resulting in a total of 480 tokens. These query tokens $[\mathbf{Q}, \mathbf{Q}_{pad}]$ and image embeddings $\mathbf{P}$ are fed into the ViT encoder, which consists of alternating self-attention and cross-attention blocks (where image embeddings serve as Keys/Values). We use the absolute positional embeddings for both query tokens and image embeddings. Finally, the truncated quantized tokens $\mathbf{Z}'$ are concatenated with the mask tokens $\mathbf{M}$ and processed by the ViT decoder, followed by a CNN decoder for image reconstruction. Notably, the mask tokens are initialized by replicating the latent representation of the first discrete token $\mathbf{Z}'$. We present the illustration and configurations of our ReTok in Figure 9 and Table 6. For autoregressive models, we apply LlamaGen [43] with absolute positional embeddings to model the latent distribution. To determine the optimal Classifier-Free Guidance (CFG) scale for gFID, we start from the CFG=1.0 with a step of 0.25.

A.2 Full Results

We present the quantitative results of generation (gFID) in Table 7 and diverse generated samples in Figure 10 by using 256 tokens. Meanwhile, progressive generation results are also presented in Figure 11. Some images generated with fewer tokens exhibit artifacts, which are mitigated as the number of tokens increases. This indicates that image complexity varies and requires different token lengths for effective representation; specifically, complex images require more tokens to achieve high-quality generation. We also evaluate the full results for image reconstruction by using ReTok in Table 8 and Table 9. Since our tokenizers are trained with the token padding, we also present the image reconstruction performance using more than 256 tokens.

A.3 Additional Ablation Study

Solely Using Hierarchical Semantic Regularization is Not Enough. We conduct the experiment training ReTok-S-S without the redundant token padding. As shown in Table 10, relying solely on semantic regularization yields suboptimal performance and suffers from a generation bottleneck. For example, the model achieves better generation quality with fewer tokens (e.g., 192 and 224) than with the full 256 tokens.

Table 6: Configurations of ReTok.

Configuration	**ReTok-S-S**	**ReTok-S-B**
Model		
Parameters	136M	232M
Codebook Size	16384	
Latent Dim	8	
Num. Tokens	256	
Pad. Tokens	224	
Training		
Training Epochs	200	250
Batch Size	128	
Retained Sequence	{32,64,...,480}	
$\lambda(k)$	2-0.5	2.5-0.5
Training Optimizer		
Optimizer	AdamW	
Learning Rate	1e-4	
Beta	$\beta_1 = 0.9, \beta_2 = 0.95$	
Scheduler	Cosine Decay	
End Learning Rate	1e-5	
Warmup Iterations	0	
Fine-tuning		
Fine-Tuning Epochs	50	
Batch Size	128	
Retained Sequence	{32,64,...,480}	
$\lambda(k)$	0.5	
Fine-tuning Optimizer	Same as the training	

Table 7: Generation performance (gFID) of AR models at different tokens.

Tokenizer	Generator	CFG	256	224	192	160	128	96	64	32
ReTok-S-S	LlamaGen-B	5.75	4.02	4.02	4.13	4.34	4.6	5.16	6.61	10.55
ReTok-S-S	LlamaGen-L	2.25	2.92	3.0	3.23	3.52	3.8	4.37	5.47	8.25
ReTok-S-B	LlamaGen-L	1.75	2.66	2.71	2.83	3.04	3.18	3.57	4.54	6.76
ReTok-S-B	LlamaGen-XL	1.5	2.27	2.27	2.38	2.58	2.7	3.16	4.05	6.22

Table 8: Reconstruction results of ReTok-S-S at different tokens.

Tokens	rFID↓	PSNR↑	SSIM↑	LPIPS↓
480	0.79	21.32	0.702	0.199
448	0.79	21.32	0.701	0.199
416	0.79	21.32	0.702	0.199
384	0.88	21.06	0.689	0.208
352	0.96	20.80	0.678	0.217
320	0.98	20.71	0.675	0.218
288	1.01	20.63	0.670	0.223
256	1.09	20.32	0.657	0.232
224	1.16	19.98	0.643	0.244
192	1.38	19.51	0.624	0.260
160	1.67	19.05	0.604	0.275
128	1.87	18.72	0.589	0.292
96	2.39	18.0	0.559	0.320
64	3.30	17.21	0.525	0.353
32	6.10	15.89	0.466	0.411

Table 9: Reconstruction results of ReTok-S-B at different tokens.

Tokens	rFID↓	PSNR↑	SSIM↑	LPIPS↓
480	0.79	21.57	0.707	0.193
448	0.79	21.57	0.707	0.193
416	0.8	21.51	0.706	0.194
384	0.82	21.37	0.702	0.197
352	0.85	21.24	0.697	0.201
320	0.9	21.06	0.689	0.207
288	0.94	20.84	0.68	0.213
256	1.01	20.57	0.668	0.222
224	1.08	20.19	0.652	0.234
192	1.23	19.72	0.633	0.249
160	1.42	19.28	0.614	0.263
128	1.56	18.89	0.596	0.279
96	1.92	18.20	0.567	0.305
64	2.66	17.43	0.532	0.340
32	4.72	16.04	0.469	0.398

Table 10: Ablation study on the role of hierarchical semantic regularization (S-S tokenizer). We report gFID↓ for ReTok with and without token padding.

Tokens	256	224	192
With token padding	4.68	4.73	4.81
W/o token padding	5.08	4.96	4.93

Table 11: Minimum number of starting tokens (S-S tokenizer).

Tokens	rFID↓	LPIPS↓	gFID↓
32	9.02	0.415	14.79
16	17.36	0.507	24.60

Minimum Numbers of Starting Tokens. Although our tokenizer supports a minimum of 32 tokens, we investigated starting from even fewer, such as 16 tokens. Unfortunately, we found that 16 tokens are insufficient to effectively represent an image for both reconstruction and generation (Table 11). Consequently, we set 32 tokens as our starting point.

A.4 Limitations

Our ReTok effectively addresses the generation bottleneck caused by naive nested dropout in flexible tokenizers. However, our tokenizer mainly focuses on the 256×256 generation, while its extension to higher image resolution is unclear. We leave this for future work. Meanwhile, our tokenizer is designed for images. For video tokenization, flexible tokenizers are also highly desirable due to the temporal redundancy of the videos.



Figure 10: Examples of generated images on ImageNet 256×256 from the ReTok-S-B + LlamaGen-XL models using 256 tokens. The classifier-free guidance is set to 4.0.

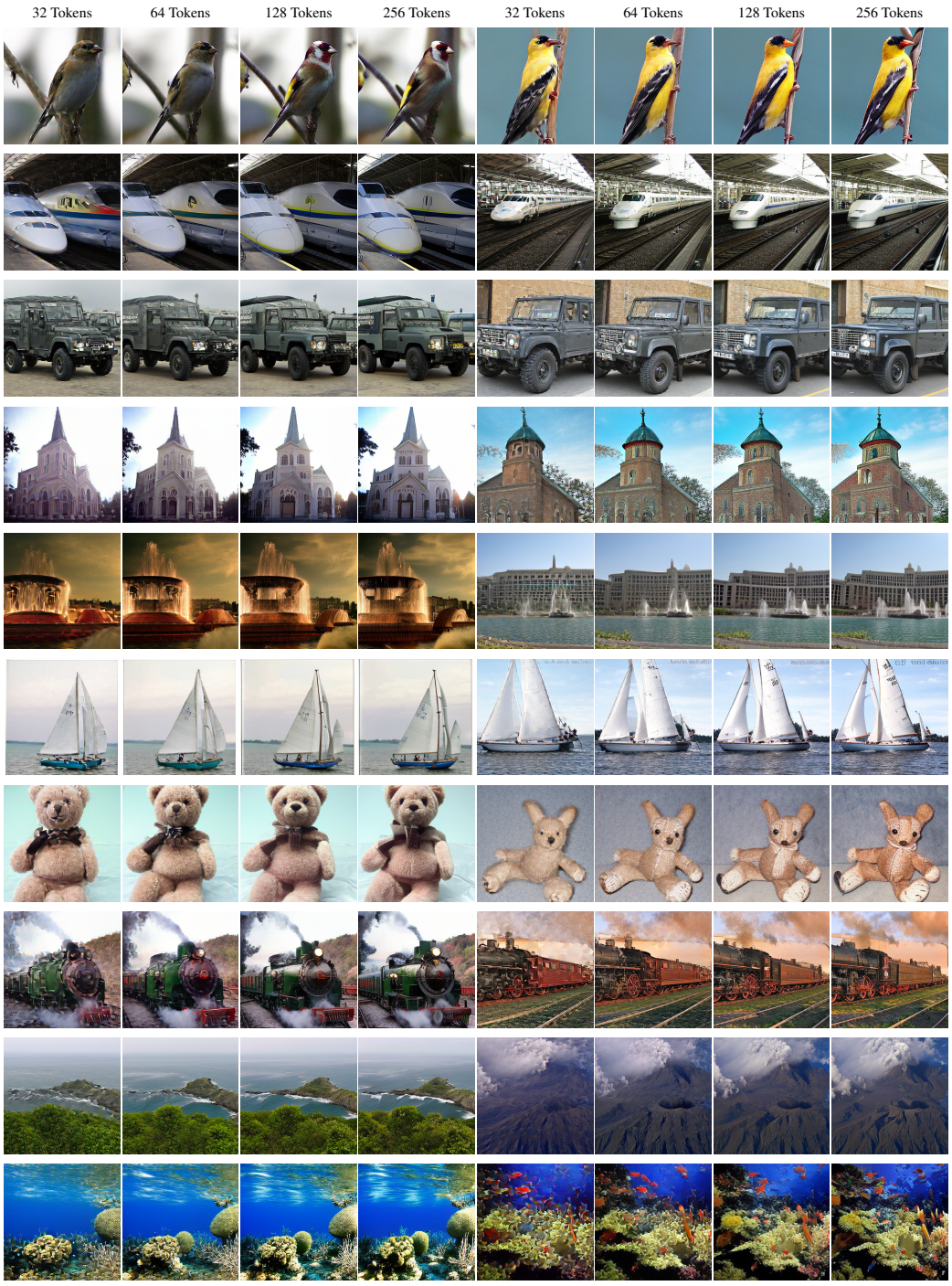


Figure 11: Examples of progressive generation on ImageNet 256×256 from the ReTok-S-B + LlamaGen-XL models. Complex scenes require more tokens, while a small number of tokens is sufficient for simple scenes.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction summarize the paper's main contributions and scope. The corresponding method and experiments are provided in Section 3 and Section 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations are discussed in Section A.4

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: this paper does not present theoretical results, theorems, or formal proofs. Our contributions are methodological and empirical, focusing on flexible image tokenizer design and downstream generation evaluation.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide the key experimental settings, including model configurations, training schedules, tokenizer settings, evaluation metrics, and datasets in Section 4 and Section A.1. These details are sufficient to reproduce the main results supporting our claims.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: We use ImageNet as the training and evaluation dataset, which is obtainable from the official source subject to its license. Code is not included in the submission, but we provide detailed implementation settings in Section 4.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide sufficient details in the paper to understand the reported results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We do not report error bars for the main experiments due to the high computational cost of training image tokenizers and AR models. We ensure fair comparison by using consistent training, evaluation protocols, and standard ImageNet validation metrics.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: We do not provide a complete accounting of compute resources for each experiment. However, we describe the model configurations with training and evaluation settings needed to reproduce the main results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics and believe this work conforms to it. The paper focuses on standard image generation benchmarks, without involving human subjects, private data, or high-risk deployment.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

Justification: We do not include a broader impacts discussion. The work focuses on methodological improvements for image tokenization and generation.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not release new datasets or high-risk pretrained generative models.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We properly cite the datasets, models, and prior methods used in this work, and follow their corresponding licenses and terms of use.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: This submission does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this work does not involve LLMs as important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.