Scalable Image Tokenization with Index Backpropagation Quantization

Fengyuan Shi[1,3*]   **Zhuoyan Luo**[2,3*]   **Yixiao Ge**[3†✉]   **Yujiu Yang**[2]   **Ying Shan**[3]   **Limin Wang**[1✉]

[*]**equal contribution**   [†] **project lead**   [✉] **corresponding author**

[1]Nanjing University   [2]Tsinghua University   [3]ARC Lab, Tencent PCG

Figure 1. **Reconstruction and generation samples of IBQ.** We show 1024×1024 reconstructed samples (top) and 256×256 generated samples (middle and bottom).

Abstract

Existing vector quantization (VQ) methods struggle with scalability, largely attributed to the instability of the codebook that undergoes partial updates during training. The codebook is prone to collapse as utilization decreases, due to the progressively widening distribution gap between non-activated codes and visual features. To solve the problem, we propose Index Backpropagation Quantization (IBQ), a new VQ method for the joint optimization of all codebook embeddings and the visual encoder. Applying a straight-through estimator on the one-hot categorical distribution between the encoded feature and codebook, all codes are differentiable and maintain a consistent latent space with the visual encoder. IBQ enables scalable training of visual tokenizers and, for the first time, achieves a large-scale codebook (2^{18}) with high dimension (256) and high utilization. Experiments on the standard ImageNet benchmark demonstrate the scalability and superiority of IBQ, achieving competitive results on reconstruction and the application of autoregressive visual generation.

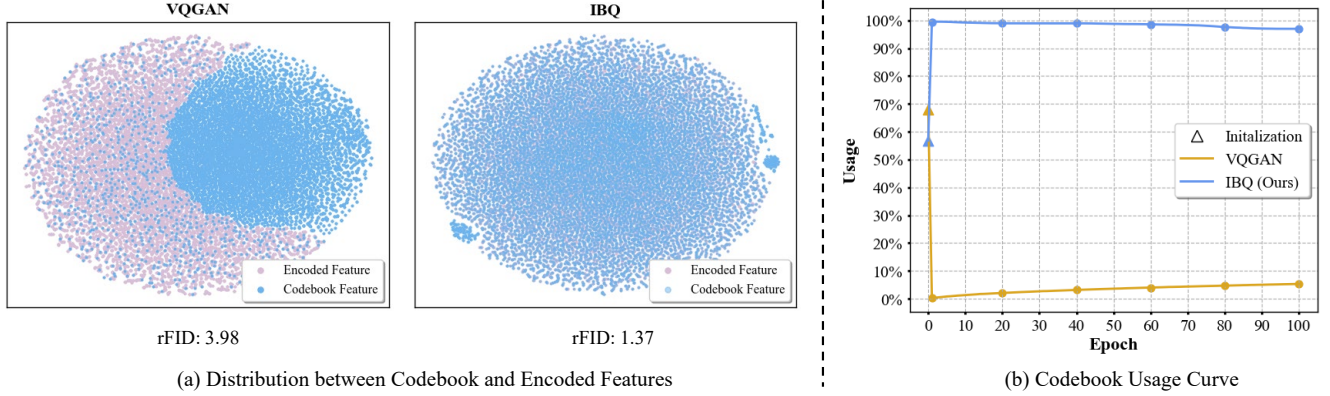| VQGAN	IBQ	
rFID: 3.98	rFID: 1.37	
(a) Distribution between Codebook and Encoded Features		(b) Codebook Usage Curve

Figure 2. **Effects of Distribution Gap on Codebook Usage.** (a) T-SNE of the codebook (16,384 codebook size and 256 dimension) and sampled encoder features. (b) Codebook usage curve. The partial-update strategy adopted by VQGAN broadens the distribution gap between encoder features and non-activated codes, while those of IBQ based on all-codes updating are evenly mixed, maintaining a high codebook usage (∼96%) throughout the training. Fully leveraging the codebook significantly improves the reconstruction quality.

1. Introduction

Discrete tokenizer plays a pivotal role in processing complex data across various modalities, such as text [5, 22, 32], images [2, 9, 34], and audio [1, 4]. By transforming raw data into discrete tokens, models can effectively handle diverse data types within a unified framework [5, 35], simplifying the integration of multimodal information and facilitating native large multimodal models [30, 36].

In the image domain, pioneering works like VQGAN [9] employ vector quantization (VQ) to learn visual tokenizers, enabling effective data compression and reconstruction. VQ tokenizers are deemed as the key component in applications such as autoregressive image generation [9, 29, 31, 34] and representation learning [2, 18]. However, a notable challenge with VQ-based methods is the information loss during quantization, leading to inferior reconstruction performance compared to continuous representation models like Variational Autoencoders (VAEs) [14, 25]. Intuitively, scaling visual tokenizers by increasing the codebook size and embedding dimension could help mitigate the information loss associated with discrete tokens, thereby bridging the gap between discrete and continuous representations. However, it is noteworthy that current visual tokenizers [29, 37] have not demonstrated such scaling properties.

Empirical research [9, 29] has revealed that current VQ methods struggle with scalability due to the inherent tendency of the codebook to collapse. This arises because these methods only optimize a limited number of the selected codes during each backpropagation. Such a widely-adopted partial update strategy gradually broadens the distribution gap between non-activated codes and the visual encoder's representation space, making the non-activated codes further less likely to be selected. As shown in Fig. 2, VQGAN [9] almost fails when scaling both codebook size (*i.e.*, 16,384) and embedding dimension (*i.e.*, 256) simul-

taneously. Only a small amount of codes share the same distribution with the visual encoder, and the codebook usage degrades from 68% to 0.002% after training one epoch.

To tackle the challenge, we introduce a new VQ method, namely, Index Backpropagation Quantization (**IBQ**). It globally updates the entire codebook in each backward process to ensure consistency with the distribution of the visual encoder. In such a way, all codes have the same probability of being selected, resulting in a high utilization of the codebook throughout the training process. Specifically, rather than directly applying the straight-through estimator [3] to the selected codes, we employ this reparameterization approach on the categorical distribution between visual features and all codebook embeddings, thereby rendering all codes differentiable. As shown in Fig. 2, the sampled visual features and the codebook embeddings from IBQ are evenly mixed. IBQ keeps a high codebook usage (∼ 96%) throughout the training process. Fully utilizing the codebook effectively enhances the representation capacity, as demonstrated by the superior reconstruction of IBQ (1.37 rFID) compared to VQGAN (3.98 rFID).

We conduct a comprehensive study on the scaling behavior of IBQ tokenizers along three axes: codebook size, code dimension, and model size. We observe significant gains in reconstruction quality or codebook usage when scaling up tokenizers. To our knowledge, IBQ is the pioneering work to train an extremely large codebook (*i.e.*, 262, 144) with a relatively large code dimension (*i.e.*, 256). This achievement leads to state-of-the-art reconstruction quality, reaching an rFID of 1.00. We further demonstrate the effectiveness of IBQ tokenizers in autoregressive image generation by integrating them with vanilla transformers of varying scales, ranging from 300M to 2.1B parameters, achieving competitive performance. Although IBQ is also compatible with other advanced autoregressive models, the paper does not focus on them, leaving them for future research.

In summary, our contributions are threefold:

- We propose a simple yet effective vector quantization method, dubbed Index Backpropagation Quantization (IBQ), for training scalable visual tokenizers.
- We study the scaling properties of IBQ by increasing codebook size, code dimension, and model size. IBQ for the first time trains a super large codebook (2^{18}) with a large dimension (256) and high usage, achieving state-of-the-art reconstruction performance.
- We validate the effectiveness of IBQ tokenizers in visual generation by equipping them with vanilla autoregressive models ranging from 300M to 2.1B, remarkably outperforming competing methods, *e.g.*, LlamaGen [29], and Open-MAGVIT2 [21].

2. Related Work

2.1. Vector Quantization

At the core of visual tokenizers is vector quantization, which maps the visual signals into discrete tokens. VQ-VAE [34] proposes an encoder-quantizer-decoder structure with a learnable codebook as the discrete representation space. VQ-VAE2 [24] introduces multi-scale hierarchical VQ-VAE to enhance local features. VQGAN [9] further uses adversarial loss and perceptual loss for good perceptual quality. RQ-VAE [17] and DQ-VAE [12] improve VQGAN by residual quantization and dynamic quantization, respectively. To improve codebook utilization for large-size codebooks, some works try to decrease code dimension [29, 37]. Following this observation, MAGVIT-v2 [38] reduces the code dimension to zero, and expands the codebook size to 2^{18} with Lookup-Free Quantization. Instead of joint optimization of the model and codebook, VQGAN-LC [42] extends the codebook size to 100,000 using a frozen codebook with a trainable projector. However, it introduces a bottleneck that constrains the tokenizer capacity.

Existing VQ methods suffer from codebook collapse when scaling up tokenizers and typically use small-size codebooks or low-dimensional code embeddings, limiting the representational capacity. In contrast, our proposed IBQ shows consistent improvements when scaling up codebook size, code dimension and model size.

2.2. Tokenized Visual Generation

Tokenizers map continuous visual signals into a discrete token sequence. For subsequent visual generation, there are two approaches, including non-autoregressive (NAR) and autoregressive (AR) generation. NAR [6, 38] usually adopts BERT-style transformers to predict masked tokens. For inference, these methods generate all tokens of an image simultaneously, and iteratively refine the generated images conditioned on the previous generation. In contrast, AR models perform next-token prediction in a raster-scan

Algorithm 1 Pseudocode of IBQ in a PyTorch-like style

```
def IBQ(z, codebook):
    '''
    z: visual feature map (B * h * w, D)
        B: batch size
        h: height of feature map
        w: width of feature map
        D: feature dimension
    codebook: (K, D)
        K: codebook size
        D: code dimension
    '''
    logits = mm(z, codebook.T) # (B * h * w, K)
    Ind_soft = softmax(logits, dim=1) # (B * h * w, K)
    _, indices = soft_one_hot.max(dim=1)
    Ind_hard = onehot(indices) # (B * h * w, K)
    Ind = Ind_hard - Ind_soft.detach() + Ind_soft
    z_q = mm(Ind, codebook) # (B * h * w, D)
    return z_q
```

mm: matrix multiplication; onehot: transfer index into one-hot vector.

manner. VQGAN [9] adopts GPT2-medium architecture, while LlamaGen [29] employs Llama [32] for scalable image generation. VAR [31] extends "next-token prediction" to "next-scale prediction" and introduces adaptive normalization (AdaLN [23]) to improve generation quality. Open-MAGVIT2 [21] proposes asymmetric token factorization for super-large codebook learning.

In this paper, we adopt vanilla autoregressive models to validate the effectiveness of IBQ tokenizers in visual generation, excluding masked modeling or multi-scale structures for simplicity. Notably, IBQ is compatible with advanced generative models, which can further unlock the tokenizer potential. We leave this exploration for future work.

3. Method

3.1. Preliminary: Vector Quantization

Vector quantization (VQ) maps continuous visual signals into discrete tokens with a fixed-size codebook $\mathcal{C} \in \mathbb{R}^{K \times D}$, where K is the codebook size and D is the code dimension. Given an image $\mathcal{I} \in \mathbb{R}^{H \times W \times 3}$, VQ first utilizes an encoder to project the image into the feature map $\mathcal{Z} \in \mathbb{R}^{h \times w \times D}$, where $h = H/p$, $w = W/p$, and p is the downsample ratio. The feature map is then quantized into $\mathcal{Q} \in \mathbb{R}^{h \times w \times D}$ discrete representations using the codebook. Finally, the decoder reconstructs the image given the quantized features.

Previous methods quantize each visual feature $z \in \mathbb{R}^D$ by selecting the nearest code from the codebook based on Euclidean distance [9, 34]. Since the $\arg\min$ operation in quantization is non-differentiable, they apply the straight-through estimator on the selected codes to copy the gradients from the decoder to the encoder, to optimize the encoder and decoder simultaneously. The quantization process can be formulated as:

$$q = \arg\min_{\mathcal{C}_k \in \mathcal{C}} ||z - \mathcal{C}_k|| \in \mathbb{R}^D, \qquad (1)$$

$$z_q = z + \text{sg}[q - z], \qquad (2)$$

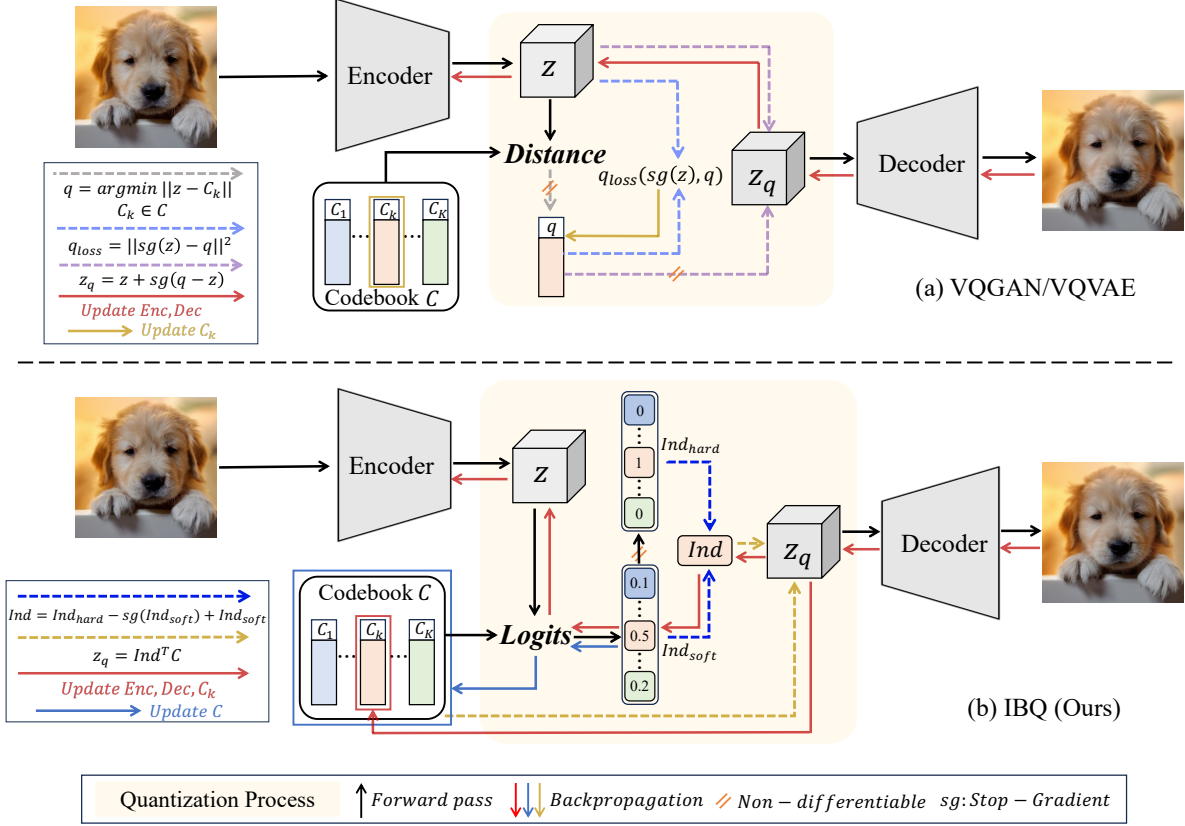where $\text{sg}[\cdot]$ is stop-gradient operation.

Figure 3. **Gradient flow of different VQ methods.** VQGAN/VQVAE only update the selected codes in each backward process. IBQ updates all codes simultaneously by transferring the gradients of soft one-hot categorical distribution to hard one-hot index.

This partial updating strategy (*i.e.*, only selected codes are optimized) adopted by these methods progressively widens the distribution gap between visual features and non-activated codes. It incurs the instability during training due to the codebook collapse, which hampers the scalability of the visual tokenizer.

3.2. Index Backpropagation Quantization

Quantization. To ensure the consistent distribution between the codebook and encoded features through the training, we introduce an all-codes updating method, Index Backpropagation Quantization (IBQ). The core idea of IBQ is to pass gradients to all codebook embeddings, rather than the selected ones only. Algorithm 1 provides the pseudo-code of IBQ.

Specifically, we first perform dot product between the given visual feature z and all code embeddings as logits and get probabilities (soft one-hot) by softmax function.

$$\text{logits} = [z^T \mathcal{C}_1, z^T \mathcal{C}_2, \cdots, z^T \mathcal{C}_K]^T \in \mathbb{R}^K, \quad (3)$$

$$\text{Ind}_\text{soft} = \text{softmax}(\text{logits}), \quad (4)$$

$$\text{Ind}_\text{hard} = \text{One-Hot}(\text{argmax}(\text{Ind}_\text{soft})). \quad (5)$$

We then copy the gradients from soft one-hot categorical

distribution to hard one-hot index:

$$\text{Ind} = \text{Ind}_\text{hard} - \text{sg}[\text{Ind}_\text{soft}] + \text{Ind}_\text{soft}. \quad (6)$$

Given the index, the quantized feature can be computed as:

$$z_q = \text{Ind}^T \mathcal{C}. \quad (7)$$

In this way, we can pass the gradients to all codes of the codebook via index. By Index Backpropagation Quantization, the distribution of the whole codebook and encoded features remains consistent throughout completed training, thus gaining a high codebook utilization.

Training Losses. Similar to VQGAN [9], the tokenizer is optimized with a combination of losses:

$$\mathcal{L} = \mathcal{L}_R + \mathcal{L}_Q + \mathcal{L}_P + \mathcal{L}_G + \mathcal{L}_E, \quad (8)$$

where $\mathcal{L}_R$ is reconstruction loss of image pixels, $\mathcal{L}_Q$ is quantization loss between the selected code embeddings and encoded features, $\mathcal{L}_P$ is perceptual loss from LPIPS [40], $\mathcal{L}_G$ is adversarial loss with PatchGAN discriminator [13] to enhance the image quality, and $\mathcal{L}_E$ is entropy penalty to encourage codebook utilization [38].

To better explain how IBQ keeps the consistent distribution between the encoder features and the whole codebook,

we provide a gradient analysis. Considering the quantization loss $\mathcal{L}_Q = ||z - z_q||^2$,

$$\frac{\partial \mathcal{L}_Q}{\partial \mathcal{C}_k} = -2\text{Ind}_k(z - z_q) = -2p_k(z - z_q), \quad (9)$$

$$p_k = \frac{\exp(z^T \mathcal{C}_k)}{\sum_{j=1}^K \exp(z^T \mathcal{C}_j)}. \quad (10)$$

The softmax probabilities p_k ensure that each $\mathcal{C}_k$ is updated based on its similarity to the encoder feature z, and $z - z_q$ shifts $\mathcal{C}_k$ toward dominant regions of the feature distribution $P_Z(z)$. Random batch sampling covers the whole encoder latent space, gradually aligning the entire codebook $\mathcal{C}$ with the distribution $P_Z(z)$ of encoder features over time.

We further introduce double quantization loss, to force the selected code embeddings and given encoded visual features towards each other.

$$z_q' = \text{Ind}_{\text{hard}}^T \mathcal{C}, \quad (11)$$

$$\mathcal{L}_Q = ||z_q - z||^2 + \\ ||\text{sg}[z] - z_q'||^2 + \beta ||z - \text{sg}[z_q']||^2. \quad (12)$$

Discussion with other VQ methods. As shown in Fig. 3, existing VQ methods (*e.g.*, VQ-VAE [34] and VQGAN [9]) update only a few selected codes within each backward process, progressively widening the gap between non-activated codes and encoded features, which leads to codebook collapse. This issue worsens as code dimension and codebook size increase. Instead of applying the straight-through estimator [3] on the selected codes, we employ it on the categorical distribution between visual features and all codebook embeddings to enable gradients backward to all codes. This promotes distribution consistency between the codebook and encoded features throughout training, allowing IBQ to scale up to extremely large codebook size with high code dimension and utilization.

3.3. Vanilla Autoregressive Transformer

After tokenization, the visual feature is quantized into discrete representations which are subsequently flattened in a raster-scan manner for visual generation. Given the discrete token index sequence $\mathcal{X} = \{x_i\}_{i=1}^T$, where $T = h' \times w'$, we employ an autoregressive transformer to model the sequence dependency through next-token prediction. Specifically, the optimization process is to maximize the log-likelihood:

$$p(x_1, \cdots, x_T | c) = \prod_{t=1}^T p(x_t | x_1, \cdots, x_{t-1}, c), \quad (13)$$

where c is the condition such as class label.

Note that, since our focus is on the visual tokenizer, we adopt the vanilla architecture of autoregressive transformers akin to Llama [32] with AdaLN [23] for visual generation. More details can be deferred to the supplementary.

4. Experiment

4.1. Datasets and Metrics

Both the visual tokenizers and autoregressive transformers are trained on 256×256 ImageNet [7]. For reconstruction, we measure reconstruction-FID (rFID [10]), codebook utilization, and LPIPS [40] on the ImageNet 50k validation set. For generation, we evaluate image quality using generation FID (gFID), Inception Score [26], and Precision/Recall [16], following ADM evaluator [8].

4.2. Implementations Details

Visual Reconstruction Setup. We adopt the same model architecture proposed in VQGAN [9]. The visual tokenizer is trained with the following settings: an initial $1e - 4$ learning rate with 0.01 multi-step decay mechanism, an Adam Optimizer [15] with $\beta_1 = 0.5$, $\beta_2 = 0.9$, a total 256 batch size with 330 epochs, a combination of reconstruction, GAN [13], perceptual [40], commitment [9], entropy [38], double quantization losses, and LeCAM regularization [33] for training stability. Unless otherwise specified, we use a codebook size of 16,384, a code dimension of 256, and 4 ResBlocks as our default tokenizer setting.

Visual Generation Setup. We use vanilla Autoregressive models ranging from 300M to 2.1B to validate the effectiveness of IBQ tokenizers in visual generation, adopting a Llama-based architecture with RoPE [28], SwiGLU [27], and RMSNorm [41]. AdaLN [23] is also incorporated for improved visual synthesis quality. The class embedding serves as both the start token and AdaLN condition. IBQ with width w, depth d and head h follows the scaling rules proposed in [29, 31], where $w = 64d, h = d$. All models are trained with similar settings: a base learning rate of $1e - 4$ per 256 batch size, an AdamW optimizer [20] with $\beta_1 = 0.9$, $\beta_2 = 0.95$, weight decay = $5e - 2$, a total 768 batch size and $300 \sim 450$ training epochs corresponding the model size, gradient clipping of 1.0, and 0.1 dropout rate for input embedding, FFN module and conditional embedding.

4.3. Main Results

Visual Reconstruction. Tab. 1 shows the quantitative reconstruction comparison between IBQ and prevalent visual tokenizers. Existing VQ methods show a significant drop in codebook usage when scaling codebook size (*e.g.*, VQGAN [9] has a 44% usage for 1024 codebook size, while a 5.9% usage for 16,384 codebook size.), and code dimension (*e.g.*, LlamaGen [29] has a 97% usage for 8-dimension codes, while a 0.29% usage for 256-dimension codes.) Therefore, the actual representational capacity is limited by the codebook collapse.

In contrast, IBQ's joint optimization of codebook embeddings and the visual encoder maintains distribution consistency, enabling stable training of large-scale codebook

Method	Token Type	Tokens	Ratio	Train Resolution	Codebook Size	Codebook Dim	rFID↓	LPIPS↓	Codebook Usage↑
VQGAN [9]	2D	16×16	16	256×256	1,024	256	7.94	—	44%
VQGAN [9]	2D	16×16	16	256×256	16,384	256	4.98	0.2843	5.9%
VQGAN* [9]	2D	16×16	16	256×256	16,384	256	3.98	0.2873	5.3%
SD-VQGAN [25]	2D	16×16	16	256×256	16,384	8	5.15	—	—
MaskGIT [6]	2D	16×16	16	256×256	1,024	256	2.28	—	—
LlamaGen [29]	2D	16×16	16	256×256	16,384	256	9.21	—	0.29%
LlamaGen [29]	2D	16×16	16	256×256	16,384	8	2.19	0.2281	97%
VQGAN-LC [42]	2D	16×16	16	256×256	16,384	8	3.01	0.2358	99%
VQGAN-LC [42]	2D	16×16	16	256×256	100,000	8	2.62	0.2212	99%
Open-MAGVIT2 [21]	2D	16×16	16	256×256	16,384	0	1.58	0.2261	100%
Open-MAGVIT2 [21]	2D	16×16	16	256×256	262,144	0	1.17	0.2038	100%
IBQ (Ours)	2D	16×16	16	256×256	16,384	256	1.37	0.2235	96%
IBQ (Ours)	2D	16×16	16	256×256	262,144	256	**1.00**	**0.2030**	84%
Titok-L [39]	1D	32	—	256×256	4,096	16	2.21	—	—
Titok-B [39]	1D	64	—	256×256	4,096	16	1.70	—	—
Titok-S [39]	1D	128	—	256×256	4,096	16	1.71	—	—

Table 1. **Reconstruction performance of different tokenizers on 256×256 ImageNet 50k validation set.** * reproduced VQGAN.

IBQ	Double Quant.	Deeper Model	rFID↓	LPIPS↓	Usage↑
			3.98	0.2873	5.3%
✓			1.67	0.2340	98%
✓	✓		1.55	0.2311	97%
✓	✓	✓	1.37	0.2235	96%

Table 2. **Effectiveness of designed modules.**

Codebook Size	rFID↓	LPIPS↓	Usage↑
1,024	2.24	0.2580	99%
8,192	1.87	0.2437	98%
16,384	1.37	0.2235	96%
262,122	1.00	0.2030	84%

Table 3. **Impact of codebook size.**

Codebook Dim	rFID↓	LPIPS↓	Usage↑
32	2.04	0.2408	92%
64	1.39	0.2281	69%
128	1.38	0.2255	77%
256	1.37	0.2235	96%

Table 4. **Effect of codebook dim.**

Num Resblock	rFID↓	LPIPS↓	Usage↑
1	1.80	0.2377	99%
2	1.55	0.2311	97%
4	1.37	0.2235	96%

Table 5. **Benefit of larger model size.**

Method	Codebook size	Codebook dim	Transformer scale	rFID↓	LPIPS↓	gFID↓	IS↑
LFQ	16,384	0	342M	1.58	0.2261	3.40	228.03
IBQ	16,384	256	342M	1.37	0.2235	2.88	254.73

Table 6. **Performance comparison with LFQ.**

with high utilization. Specifically, IBQ with 16,384 codebook size and 256 code dimension achieves 1.37 rFID, outperforming other VQ methods at the same downsampling rate and codebook size. Increasing the codebook size to 262,144, IBQ achieves state-of-the-art reconstruction with 1.00 rFID, surpassing Open-MAGVIT2 [21]. A qualitative comparison in Fig. 4 shows IBQ's superior visual quality in complex scenarios such as faces and characters. Note that, we observe that incorporating additional facial data yields consistent improvements (see supplementary materials).

Visual Generation. In Tab. 7, we compare IBQ with other generative models, including Diffusion models, AR models, and variants of AR models (VAR [31] and MAR [19]) on class-conditional image generation. With powerful IBQ tokenizers, our models show consistent improvements when scaling up the model size (from 300M to 2.1B), and outperform all previous vanilla autoregressive models at different scales of model size. Moreover, IBQ outperforms the diffusion-based model DiT [23], and achieves comparable results with the variants of AR models.

These AR model variants focus on the architecture designs of transformers in the second stage, while our work is devoted to better visual tokenizers in the first stage. Therefore, we believe that with our stronger tokenizers, the AR models and their variants can be boosted further.

4.4. Scaling Up IBQ Tokenizers

Existing VQ methods struggle to scale up due to the codebook collapse. For example, LlamaGen [29] sees a significant drop in usage and rFID when increasing the code dimension from 8 to 256 (97% → 0.29%, 2.19 rFID → 9.21 rFID), as shown in Tab. 1. This is due to their partial updates during training, which progressively widens the distribution gap between non-activated codes and encoded features.

Scaling Up Tokenizers Improves Reconstruction. IBQ tokenizers show promising scaling capacity for reconstruction in three aspects: **1) Codebook Size**: As shown in Tab. 3, reconstruction quality improves significantly as the codebook size increases from 1,024 to 16,384, with high utilization and consistent visual soundness even at 262,144

Figure 4. **Qualitative Reconstruction Comparison.** We compare IBQ with LlamaGen and Open-MAGVIT2 tokenizer.
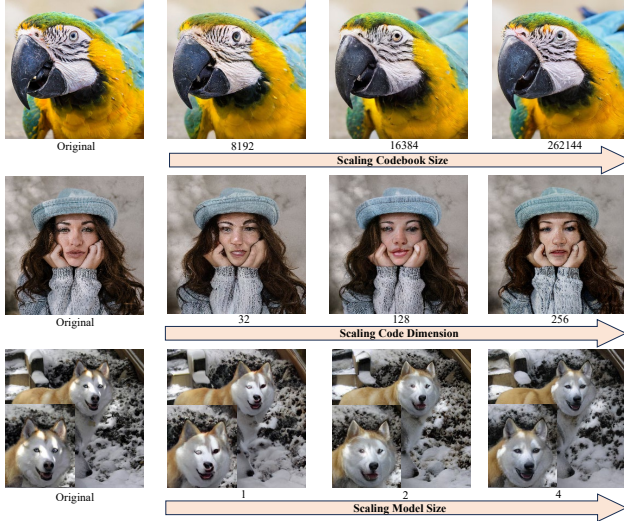

Figure 5. **Scaling up visual tokenizers (*e.g.*, codebook size, code dimension and model size) improves visual soundness of reconstruction.**

codes. **2) Code Dimension**: interestingly, we observe a notable increase in codebook usage when scaling code dimension in Tab. 4. We assume that low-dimensional codes are less discriminative and tend to be clustered, indicating that representative codes are more likely to be selected under our global updating strategy. In contrast, high-dimensional codes are highly informative due to their sparsity in the representation space, allowing for more even selection during training, which ensures high utilization with better performance. **3) Model Size**: Tab. 5 reveals that by increasing the number of ResBlock both in both the encoder and decoder,
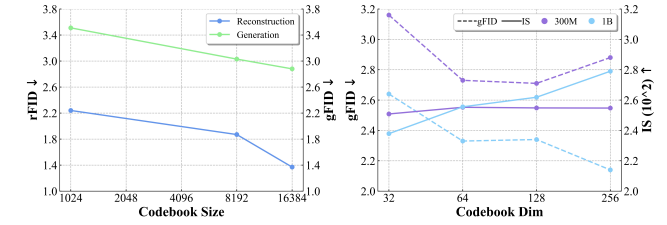

Figure 6. **Scaling up IBQ tokenizers enables better generation, especially with larger autoregressive models (*e.g.*, 1B param.).**

a higher reconstruction performance can be guaranteed.

With these factors, we realize a super large codebook of 262,144 codebook size and 256 dimensions with high codebook usage (84%), achieving the state-of-the-art reconstruction performance (1.00 rFID). To better illustrate the scaling properties, we also provide visualizations in Fig. 5.

Scaling Up Tokenizers Improves Generation. Scaling up IBQ tokenizers also enhances generation quality. As shown in Fig. 6, increasing the codebook size significantly improves reconstruction and generation FID, with a similar trend observed when scaling code dimensions. Moreover, with larger autoregressive models (*e.g.*, 1B parameters), the improvement in generation quality becomes more remarkable, suggesting that scaling up generative models can further unlock the potential of IBQ tokenizers.

4.5. Ablation Studies

Key Designs. To validate the effectiveness of our method, we conduct ablation studies on several key designs, as shown in Tab. 2. The re-implemented VQGAN performance is 3.98 rFID and 5.3% codebook utilization. Dif-

Type	Model	#Para.	FID↓	IS↑	Precision↑	Recall↑
Diffusion	ADM [8]	554M	10.94	101.0	0.69	0.63
	CDM [11]	—	4.88	158.7	—	—
	LDM-4 [25]	400M	3.60	247.7	—	—
	DiT-XL/2 [23]	675M	2.27	278.2	0.83	0.57
VAR	VAR-d16 [31]	310M	3.30	274.4	0.84	0.51
	VAR-d20 [31]	600M	2.57	302.6	0.83	0.56
	VAR-d24 [31]	1.0B	2.09	312.9	0.82	0.59
	VAR-d30 [31]	2.0B	1.92	323.1	0.82	0.59
MAR	MAR-B [19]	208M	2.31	281.7	0.82	0.57
	MAR-L [19]	479M	1.78	296.0	0.81	0.60
	MAR-H [19]	943M	**1.55**	303.7	0.81	**0.62**
Vanilla AR	VQGAN [9]	227M	18.65	80.4	0.78	0.26
	VQGAN [9]	1.4B	15.78	74.3	—	—
	VQGAN-re [9]	1.4B	5.20	280.3	—	—
	ViT-VQGAN [37]	1.7B	4.17	175.1	—	—
	ViT-VQGAN-re [37]	1.7B	3.04	227.4	—	—
	RQTran. [17]	3.8B	7.55	134.0	—	—
	RQTran.-re [17]	3.8B	3.80	**323.7**	—	—
	LlamaGen-L [29]	343M	3.80	248.28	0.83	0.51
	LlamaGen-XL [29]	775M	3.39	227.08	0.81	0.54
	LlamaGen-XXL [29]	1.4B	3.09	253.61	0.83	0.53
	LlamaGen-3B [29]	3.1B	3.06	279.72	0.84	0.53
	LlamaGen-L* [29]	343M	3.07	256.06	0.83	0.52
	LlamaGen-XL* [29]	775M	2.62	244.08	0.80	0.57
	LlamaGen-XXL* [29]	1.4B	2.34	253.90	0.80	0.59
	LlamaGen-3B* [29]	3.1B	2.18	263.33	0.81	0.58
	Open-MAGVIT2-B [21]	343M	3.08	258.26	**0.85**	0.51
	Open-MAGVIT2-L [21]	804M	2.51	271.70	0.84	0.54
	Open-MAGVIT2-XL [21]	1.5B	2.33	271.77	0.84	0.54
Vanilla AR	IBQ-B (Ours)	342M	2.88	254.73	0.84	0.51
	IBQ-L (Ours)	649M	2.45	267.48	0.83	0.52
	IBQ-XL (Ours)	1.1B	2.14	278.99	0.83	0.56
	IBQ-XXL (Ours)	2.1B	2.05	286.73	0.83	0.57

Table 7. **Class-conditional generation on 256 × 256 ImageNet.** ∗ specifies the generated images are 384 × 384 and are resized to 256×256 for evaluation. The evaluation protocol and implementation are the same as ADM [8].

ferent from previous methods, the replacement from VQ to IBQ achieves consistent distribution between encoded features and the whole codebook by rendering all code differentiable, which brings a clear improvement of both codebook usage (5.3%→ 98%) and reconstruction quality (3.98 rFID→1.67 rFID). By incorporating double quantization loss to force the selected code embeddings and encoded visual features toward each other, IBQ guarantees more precise quantization. Following MAGVIT-v2 [38], we enlarge the model size for better compacity, and the reconstruction performance gets improved correspondingly.

Comparison with LFQ. For fair comparisons, we adopt LFQ [21] with 16,384 codes and replace its asymmetric token factorization with our vanilla transformer architecture. Tab. 6 shows that IBQ outperforms LFQ in both reconstruction and generation, which demonstrates increasing code dimension can improve the reconstruction ability of the visual tokenizer and further boost the visual generation.

5. Conclusion

In this paper, we identify the bottleneck in scaling tokenizers (*e.g.*, codebook size), stemming from the partial-update strategy in current VQ methods, which progressively enlarge the distribution gap between encoded features and non-activated codes, eventually leading to codebook collapse. To address this challenge, we propose a simple yet effective vector quantization method, termed as Index Backpropagation Quantization (**IBQ**), for scalable tokenizer training, which updates all codes by applying the straight-through estimator on the categorical distribution over visual features and all codebook embeddings, thereby maintaining consistent distribution between the entire codebook and encoded features. Experiments on ImageNet demonstrate that IBQ enables a high-utilization, large-scale visual tokenizer with improved performance in both reconstruction (1.00 rFID) and generation (2.05 gFID).

References

[1] Alexei Baevski, Steffen Schneider, and Michael Auli. vq-wav2vec: Self-supervised learning of discrete speech representations. *arXiv preprint arXiv:1910.05453*, 2019. 2

[2] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021. 2

[3] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 2, 5

[4] Zalán Borsos, Raphaël Marinier, Damien Vincent, Eugene Kharitonov, Olivier Pietquin, Matt Sharifi, Dominik Roblek, Olivier Teboul, David Grangier, Marco Tagliasacchi, et al. Audiolm: a language modeling approach to audio generation. *IEEE/ACM transactions on audio, speech, and language processing*, 31:2523–2533, 2023. 2

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *NeurIPS*, pages 1877–1901, 2020. 2

[6] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. Maskgit: Masked generative image transformer. In *CVPR*, pages 11305–11315, 2022. 3, 6

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. 5

[8] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *NeurIPS*, 34:8780–8794, 2021. 5, 8

[9] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, pages 12873–12883, 2021. 2, 3, 4, 5, 6, 8

[10] 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. In *NeurIPS*, 2017. 5

[11] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. 23(1):2249–2281, 2022. 8

[12] Mengqi Huang, Zhendong Mao, Zhuowei Chen, and Yongdong Zhang. Towards accurate image coding: Improved autoregressive image generation with dynamic vector quantization. In *CVPR*, pages 22596–22605, 2023. 3

[13] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, pages 5967–5976, 2017. 4, 5

[14] Diederik P Kingma. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2

[15] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[16] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. In *NeurIPS*, pages 3929–3938, 2019. 5

[17] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *CVPR*, pages 11513–11522, 2022. 3, 8

[18] Tianhong Li, Huiwen Chang, Shlok Mishra, Han Zhang, Dina Katabi, and Dilip Krishnan. Mage: Masked generative encoder to unify representation learning and image synthesis. In *CVPR*, pages 2142–2152, 2023. 2

[19] Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. *arXiv preprint arXiv:2406.11838*, 2024. 6, 8

[20] I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 5

[21] Zhuoyan Luo, Fengyuan Shi, Yixiao Ge, Yujiu Yang, Limin Wang, and Ying Shan. Open-magvit2: An open-source project toward democratizing auto-regressive visual generation. *arXiv preprint arXiv:2409.04410*, 2024. 3, 6, 8

[22] OpenAI. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 2

[23] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *CVPR*, pages 4195–4205, 2023. 3, 5, 6, 8

[24] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. In *NeurIPS*, 2019. 3

[25] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, pages 10684–10695, 2022. 2, 6, 8

[26] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NeurIPS*, 2016. 5

[27] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020. 5

[28] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2024. 5

[29] 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. 2, 3, 5, 6, 8

[30] Chameleon Team. Chameleon: Mixed-modal early-fusion foundation models. *arXiv preprint arXiv:2405.09818*, 2024. 2

[31] Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *arXiv preprint arXiv:2404.02905*, 2024. 2, 3, 5, 6, 8

[32] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. 2, 3, 5

[33] Hung-Yu Tseng, Lu Jiang, Ce Liu, Ming-Hsuan Yang, and Weilong Yang. Regularizing generative adversarial networks under limited data. In *CVPR*, pages 7921–7931, 2021. 5

[34] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *NeurIPS*, 2017. 2, 3, 5

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017. 2

[36] 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. 2

[37] 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 *ICLR*, 2022. 2, 3, 8

[38] Lijun Yu, Jose 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 *ICLR*, 2024. 3, 4, 5, 8

[39] Qihang Yu, Mark Weber, Xueqing Deng, Xiaohui Shen, Daniel Cremers, and Liang-Chieh Chen. An image is worth 32 tokens for reconstruction and generation. *arXiv preprint arXiv:2406.07550*, 2024. 6

[40] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, pages 586–595, 2018. 4, 5

[41] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona T. Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. OPT: open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022. 5

[42] Lei Zhu, Fangyun Wei, Yanye Lu, and Dong Chen. Scaling the codebook size of vqgan to 100,000 with a utilization rate of 99%. *arXiv preprint arXiv:2406.11837*, 2024. 3, 6