StableVQ: Practical Guidelines for Stable Vector-Quantized Tokenizer Training

Bao Tang[1,2,§] **Jiahao Guo**[1,2,§] **Haoxiang Cao**[2,3,§]
Wenyu Liu[1] **Changqian Yu**[2,†] **Kun Gai**[2] **Xinggang Wang**[1,†]

[1]Huazhong University of Science and Technology [2]KlingAI Research
[3]South China Normal University

Abstract

Vector Quantization (VQ) is fundamental to discrete visual tokenizers that power modern autoregressive and masked image generation models. While recent shared-projection codebook methods have substantially advanced codebook utilization, training stability remains a critical and underexplored challenge. We argue that the root cause lies in the *entanglement* of the Encoder–Decoder and Codebook training: because neither module can reliably fulfill its own responsibility in isolation, the system can only function when the two subsystems happen to cooperate—a fragile condition that breaks down precisely when training is most stressed. We propose **StableVQ**, which revisits the proper learning objective of each module and resolves the problems that arise when each is trained to fulfill its own role independently. Concretely, (1) **Dynamic STE** corrects the instability in the Encoder's learning objective, enabling it to robustly optimize the reconstruction space under discrete regularization even when codebook utilization is low. (2) **Region VQ Loss** reconceives the Codebook's learning objective so that it can independently guarantee full tracking of the encoder output distribution, without relying on encoder oscillations to drive activation. (3) **Decoupled Schedule** recognizes that the distinct responsibilities of the Encoder–Decoder and the Codebook demand distinct optimization dynamics, and assigns each an independent learning rate schedule to ensure robust system-level behavior. Built on top of shared-projection codebooks, StableVQ is lightweight and introduces no learnable parameters. Experiments on ImageNet demonstrate consistent improvements in training stability, codebook utilization, and reconstruction quality across diverse codebook sizes and initialization settings.

1 Introduction

Visual tokenization has become a foundational component of modern generative vision systems [30, 6, 27, 28]. By mapping continuous image features to sequences of discrete tokens via a learned codebook, VQ-VAEs [30] enable autoregressive transformers [27, 28, 32], masked generative models [1], and multimodal language models to operate over compact, structured visual representations. The expressiveness of the resulting vocabulary directly determines the upper bound on downstream generation quality.

A persistent obstacle in VQ training is **codebook collapse**, where most code vectors are never assigned, severely underutilizing model capacity. Recent *shared-projection* methods [11, 36, 2] address this by reparameterizing codebook entries as $\tilde{\mathbf{e}}_k = f_\theta(\mathbf{e}_k)$ via a shared differentiable function, so that gradients propagate across the entire code distribution. This reframes VQ training as a *distribution alignment* problem between the projected code distribution and the encoder output distribution, substantially advancing codebook utilization.

[§]Work done during internship at KlingAI Research.
[†]Corresponding authors: X. Wang (xgwang@hust.edu.cn) & C. Yu (y-changqian@outlook.com).

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

Yet **training stability remains a critical and underexplored challenge**. In practice, convergence is sensitive to initialization; codebooks may stagnate in low-utilization phases; and abrupt utilization collapse can occur mid-training, particularly at scale. We argue that these failure modes are not incidental but symptomatic of a deeper structural issue: the Encoder–Decoder and the Codebook are entangled in their training, such that neither can reliably fulfill its own responsibility in isolation. This entanglement masks the latent dysfunction of each module, leaving underlying issues unresolved and making the overall system contingent on fragile inter-module cooperation rather than principled individual competence. In Section 4.1, we provide a principled analysis of the latent problems in existing VQ tokenizer training that this entanglement conceals.

We propose **StableVQ**, a set of lightweight interventions that enables principled, stable VQ training without relying on fragile inter-module cooperation. Our contributions are as follows:

- **A separation-of-concerns analysis of VQ tokenizer training.** We revisit the proper responsibility of each module in VQ training and show that inter-module entanglement has long concealed latent dysfunctions in each. This analysis reframes training instability as a failure of modular responsibility rather than a fundamental limitation of the quantization paradigm.
- **StableVQ: principled and lightweight interventions for stable training.** Guided by the above analysis, we propose three targeted, parameter-free components that enable each module to fulfill its own responsibility independently. Together, they achieve principled, stable VQ training across diverse codebook sizes and initialization settings, without relying on heuristic design choices.
- **A more accessible performance ceiling for VQ tokenizers.** By eliminating the dependence on heuristic initialization and shared-projection architecture design, StableVQ allows the full expressive potential of VQ tokenizers to be realized without optimization artifacts standing in the way. A single linear projection suffices to reach state-of-the-art quality across diverse settings, and the principled modular stability established by our framework offers a solid theoretical footing for extending VQ training reliably to more demanding scenarios.

2 Related Work

We provide a brief overview of related work here, with a more comprehensive version in Appendix A. Vector-quantized representation learning was introduced by VQ-VAE [30], which maps continuous encoder features to discrete code indices through nearest-neighbor lookup in a learned codebook. Subsequent tokenizers improve reconstruction quality and token capacity through hierarchical latents [22], perceptual and adversarial objectives [6], residual or multi-stage quantization [13], and ViT-based architectures [31]. A central challenge in these systems is codebook collapse and low utilization, especially as codebook size or embedding dimension increases. Existing remedies include code reset or replacement [34], low-dimensional embeddings and normalization [31], relaxed or soft-assignment paths [21, 12, 25], and scalar or binary quantization alternatives [18, 32]. More recent shared-projection methods such as VQ-STE++, SimVQ, and FVQ [11, 36, 2] reparameterize code vectors through a shared function, substantially improving utilization with little architectural overhead. StableVQ builds on this line and studies the remaining training-instability problem.

3 Background

3.1 Vector Quantization

Given an encoder E and a decoder D, a VQ-VAE [30] maps an input image $\mathbf{x}$ to a spatial feature map $\mathbf{z} = E(\mathbf{x}) \in \mathbb{R}^{H \times W \times d}$. Each spatial feature $\mathbf{z}_{ij} \in \mathbb{R}^d$ is quantized by nearest-neighbor lookup in a codebook $\mathcal{C} = \{\mathbf{e}_k\}_{k=1}^K$:

$$k_{ij}^* = \arg\min_{k \in [K]} \|\mathbf{z}_{ij} - \mathbf{e}_k\|_2^2, \qquad \hat{\mathbf{z}}_{ij} = \mathbf{e}_{k_{ij}^*}. \tag{1}$$

The training objective decomposes into three terms:

$$\mathcal{L} = \mathcal{L}_{\text{recon}} + \beta \underbrace{\|\mathbf{z} - \text{sg}[\hat{\mathbf{z}}]\|^2}_{\text{commitment loss}} + \underbrace{\|\text{sg}[\mathbf{z}] - \hat{\mathbf{z}}\|^2}_{\text{VQ loss}}, \tag{2}$$

where sg$[\cdot]$ is the stop-gradient operator. The commitment loss constrains encoder outputs to remain near their assigned codes; the VQ loss drives codebook entries toward the encoder output distribution; and the end-to-end gradient from the reconstruction loss is passed back to the Encoder via the Straight-Through Estimator (STE), which approximates $\partial\mathcal{L}/\partial\mathbf{z}_{ij} \approx \partial\mathcal{L}/\partial\hat{\mathbf{z}}_{ij}$.

3.2 Codebook with a Shared Projection

In the standard formulation, each code $\mathbf{e}_k$ is an independent parameter, so the VQ loss gradient is nonzero only for selected codes—a property we term *gradient sparsity*. As codebook size K grows, the fraction of codes updated per step diminishes, exacerbating collapse risk.

A recent line of work addresses gradient sparsity via a *shared projection*: codes are reparameterized as $\tilde{\mathbf{e}}_k = f_\theta(\mathbf{e}_k)$, where f_θ is a differentiable function shared across all codes:

$$k^*_{ij} = \arg\min_{k\in[K]} \|\mathbf{z}_{ij} - f_\theta(\mathbf{e}_k)\|_2^2. \tag{3}$$

Because f_θ is shared, gradients from any selected code propagate to influence all $\{\mathbf{e}_k\}$, transforming VQ training into an alignment problem between the projected code distribution $\mathcal{P}_\mathcal{C} = \{f_\theta(\mathbf{e}_k)\}_{k=1}^K$ and the encoder output distribution $\mathcal{P}_\mathcal{Z} = \{E(\mathbf{z}_{ij})\}$.

Representative instantiations of this paradigm include the affine reparameterization approach of [11], which applies a shared learnable scale-and-shift transformation to the code vectors, rescaling each quantized embedding as $q = c_{\text{mean}} + c_{\text{std}} \odot \hat{q}$ where c_{mean} and c_{std} are codebook-wide affine parameters; SimVQ [36], which defines f_θ as a learnable linear layer W acting on a fixed latent basis, so that each code vector is produced as $c_k = q_k W$, optimizing the entire linear space spanned by the codebook rather than individual code vectors; and FVQ [2], which employs a more expressive nonlinear projector to remap code vectors, enabling full codebook utilization. These methods substantially improve codebook utilization, but training stability remains unsolved, as shown below.

4 Method

4.1 Failure Modes of VQ Training

Limitations of shared-projection methods. Although shared-projection methods substantially mitigate gradient sparsity, they do not fully resolve the challenge of codebook distribution alignment. First, gradient propagation through f_θ influences all codes indirectly: the signal reaching inactive codes is diffuse and undirected, providing no guarantee that they converge toward the correct regions of the token distribution. Second, this indirect influence is subject to decay: once a small subset of codes covers the token distribution well enough to minimize the VQ loss, the training signal driving the remaining codes becomes negligible. Furthermore, shared-projection methods operate purely on the codebook side and do not account for potential instabilities in the Encoder's optimization during training—a separate source of fragility that can compound with codebook misalignment and destabilize the system as a whole.

Challenging failure modes. Wasserstein VQ [7] analyzes static relationships between token and code distributions. We further examine characteristic failure modes observed during training, analyzing how the token–code distributional relationship evolves in each case and what drives this evolution (Figure 1). **(a) Code scale $\ll$ Token scale**: when the code distribution occupies a smaller scale region than the token distribution, the Codebook receives only sparse optimization targets due to low utilization, while the token distribution fluctuates unpredictably under the competing gradients of the STE-passed reconstruction signal and the commitment loss. The system thus falls into *prolonged low utilization*; when these fluctuations are severe enough, commitment loss spikes can escalate to NaN gradients before the codebook ever reaches meaningful utilization. **(b) Code scale $\gg$ Token scale**: when the code distribution spans a much larger region than the token distribution, utilization rises rapidly as codes within the token scale region are quickly activated. However, as more in-range codes are claimed, the commitment loss and VQ loss signals become increasingly saturated, rapidly diminishing the training signal for out-of-range codes. This leaves the majority of the codebook virtually unreachable by nearest-neighbor assignment, resulting in *permanent dead codes*. **(c) Scale divergence**: even in a well-utilized codebook, the two distributions are in continuous dynamic alignment as the reconstruction loss drives the token distribution to evolve. Should the

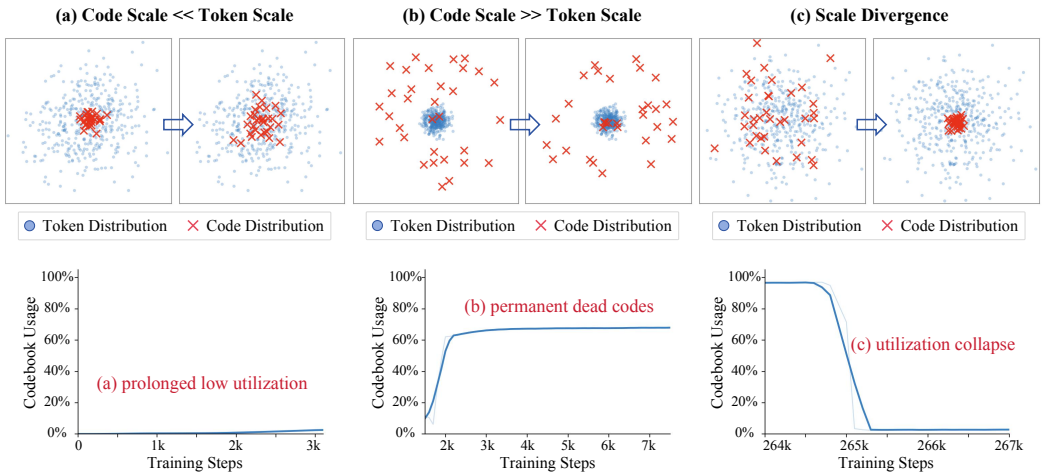


Figure 1: Three characteristic failure modes of VQ training, illustrated through the relationship between the token distribution (blue) and code distribution (red) at different training stages.

Codebook momentarily fail to track a sudden distributional shift, the erroneous STE gradients passed to the Encoder tend to amplify the divergence rather than correct it, triggering a positive-feedback loop that rapidly escalates the scale mismatch and causes *utilization collapse* instantaneously.

The failure modes described above share a common root cause: the Encoder and Codebook are not given the conditions to independently fulfill their own responsibilities. Guided by the principle of separation of concerns, we analyze the proper learning objective of each module and identify where the current training pipeline prevents each from fulfilling its own role.

4.2 Dynamic Straight-Through Estimator

Gradient Estimation Gap. The Encoder's proper learning objective is to optimize the reconstruction space under the discrete code constraint imposed by the commitment loss. Ideally, the commitment loss and the STE-passed reconstruction gradient should cooperate toward this objective. However, when tokens are assigned to distant codes, the STE gradient becomes an unreliable estimate of the true reconstruction direction. Such unreliable gradients can conflict with the commitment loss, push the token and code distributions further apart, and trigger commitment-loss spikes that may escalate to NaN values. By amplifying distributional errors rather than correcting them, they also become a primary driver of sudden utilization collapse, undermining overall training robustness.

Gradient Quality Weighting. The key observation is that when multiple tokens hit the same code, the *relatively farther* tokens are those whose STE gradients are most unreliable. We therefore assign each token a gradient weight based on its distance to the assigned code relative to the best-matched token for that code:

$$w_{ij} = \mathrm{sg}\left[\frac{d^*_{k^*_{ij}}}{\|\mathbf{z}_{ij} - f_\theta(\mathbf{e}_{k^*_{ij}})\|_2^2}\right] \in (0,1], \quad d^*_k = \min_{(i',j')} \|\mathbf{z}_{i'j'} - f_\theta(\mathbf{e}_k)\|_2^2, \tag{4}$$

where d^*_k is the minimum squared distance from code k to any token in the current batch. When a token is the best-matched token for its assigned code, it receives the full gradient with $w_{ij} = 1$. As its relative quantization error grows, the weight decreases accordingly.

We modulate the STE by this per-token weight:

$$\hat{\mathbf{z}}_{ij} = w_{ij} \cdot \mathbf{z}_{ij} + \mathrm{sg}[\hat{\mathbf{z}}_{ij} - w_{ij} \cdot \mathbf{z}_{ij}]. \tag{5}$$

This formulation attenuates the STE gradient for relatively distant tokens while preserving the full gradient for the token with the best assignment. Crucially, when all tokens in a batch are assigned to well-matched codes (high utilization, stable training), $w_{ij} \approx 1$ for all tokens and Dynamic STE reduces to the standard STE. The intervention is thus self-deactivating under healthy training conditions, and no threshold hyperparameter is required.

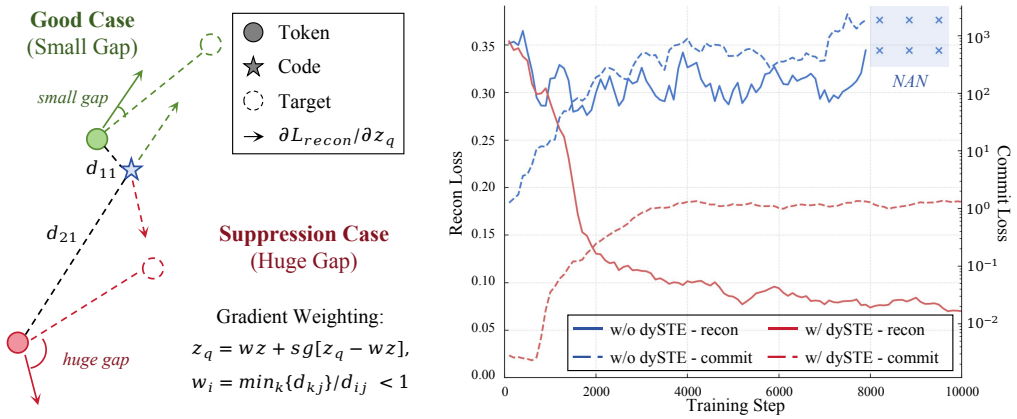


Figure 2: **Left: Dynamic STE design.** Relatively farther tokens have more unreliable gradients and receive proportionally suppressed contributions. **Right: Pilot Study.** Standard STE causes commitment loss spikes and training collapse, whereas Dynamic STE maintains stable losses.

Pilot Study. To verify the effect of Dynamic STE on the Encoder's learning objective, we freeze the Codebook and train only the Encoder–Decoder. Figure 2 shows that standard STE leads to severe commitment-loss spikes and reconstruction-loss oscillations, ultimately causing losses to diverge to NaN, while Dynamic STE suppresses these instabilities and keeps losses stable throughout training. This confirms that the gradient estimation gap introduces latent instability into Encoder training.

4.3 Region VQ Loss

Unguaranteed Distribution Alignment. The Codebook's proper learning objective is to track the encoder output distribution through a clean and independent optimization process. In the ideal setting, this should endow the Codebook with the ability to guarantee codebook utilization on its own, without relying on assistance from the Encoder. However, the standard VQ loss only provides direct learning targets to the codes selected in the current step, leaving the majority of codes without explicit supervision. Although shared-projection methods allow gradients to reach all codes through f_θ, the resulting signal remains indirect and undirected, and is inherently subject to attenuation during training. Consequently, the Codebook cannot independently guarantee full utilization; instead, activation of the remaining codes often depends on unstable fluctuations in the encoder output, meaning that codebook utilization is not reliably guaranteed in practice.

Asymmetry in VQ Loss. The standard VQ loss is inherently asymmetric: every token receives an explicit target through the commitment loss, whereas only selected codes are assigned meaningful objectives. Letting all codes take the nearest token as their target seems a natural remedy, yet this often fails to provide correct learning directions. The solution lies in shifting from point-wise to distribution-wise alignment. Since asymmetry arises from many tokens selecting few codes, we propagate the targets received by active codes to nearby inactive ones, which we term Region VQ.

Code Target Assignment. Let $[K] = \{1, \ldots, K\}$ denote all code indices and $\mathcal{S}_t \subseteq [K]$ denote the set of codes selected at step t. To distinguish codes by activity, we maintain a FIFO queue of length W, which defines the window-active set $\mathcal{A}_t = \bigcup_{s=t-W+1}^{t} \mathcal{S}_s$ and the persistently inactive set $\mathcal{N}_t = [K] \setminus \mathcal{A}_t$. Let $\mathcal{U}_k$ contain the token features $\mathbf{z}_u$ assigned to code k, with $n_k = |\mathcal{U}_k|$. Each source code $k \in \mathcal{S}_t$ receives a quota $q_k \propto n_k$ and propagates its target to the q_k nearest codes in $\mathcal{N}_t$, denoted $\mathcal{R}_k$ (Algorithm 1). Let $\mathcal{K}_j = \{k \in \mathcal{S}_t : j \in \mathcal{R}_k\}$ denote the sources propagating to code j. Recently active codes $\mathcal{A}_t \setminus \mathcal{S}_t$ and unclaimed inactive codes $\{j \in \mathcal{N}_t : |\mathcal{K}_j| = 0\}$ retain self-targets and yield zero loss, while other effective code targets are defined as follows:

$$\mathbf{t}_j = \begin{cases} \dfrac{1}{|\mathcal{U}_j|} \displaystyle\sum_{u \in \mathcal{U}_j} \mathbf{z}_u, & j \in \mathcal{S}_t, \\ \dfrac{1}{|\mathcal{K}_j|} \displaystyle\sum_{k \in \mathcal{K}_j} \mathbf{t}_k, & j \in \mathcal{N}_t,\ |\mathcal{K}_j| > 0. \end{cases} \tag{6}$$

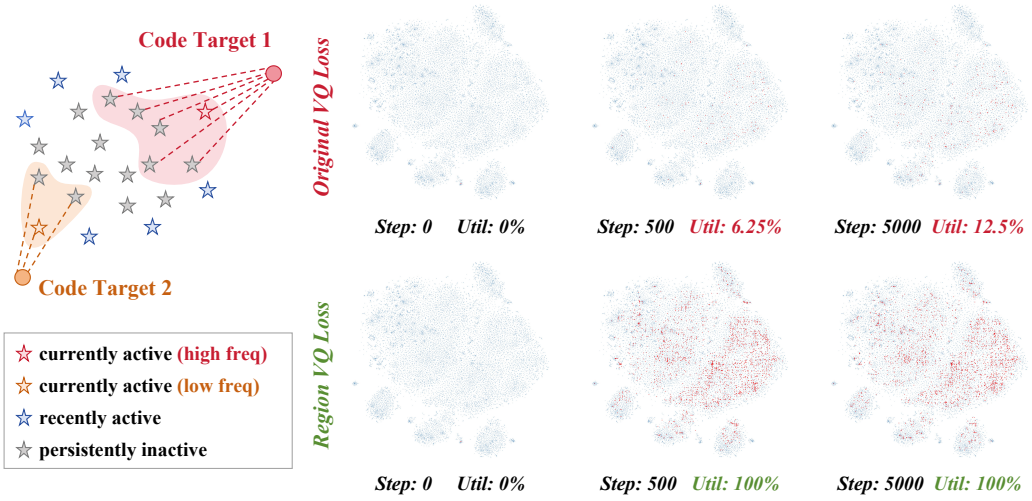


Figure 3: **Left: Region VQ Loss.** Active codes propagate targets to nearby inactive codes proportionally to selection count. **Right: Pilot Study.** T-SNE visualizations at different training steps (blue: encoder outputs; red: codebook entries) with codebook utilization.

Let $\mathcal{M}_t = \mathcal{S}_t \cup \{j \in \mathcal{N}_t : |\mathcal{K}_j| > 0\}$ denote codes with effective targets. The unified codebook update loss is

$$\mathcal{L}_{\text{code}} = \frac{1}{|\mathcal{M}_t|} \sum_{k \in \mathcal{M}_t} \|f_\theta(\mathbf{e}_k) - \text{sg}[\mathbf{t}_k]\|_2^2. \tag{7}$$

Pilot Study. To isolate the effect of Region VQ Loss on the Codebook's learning objective, we freeze the Encoder and optimize only the Codebook, which uses a two-layer ViT Block as the shared projector to ensure sufficient learning capacity. Figure 3 shows T-SNE visualizations of the encoder output distribution and codebook entries at Steps 0, 500, and 5000. The standard VQ loss stagnates at around 12.5% utilization even after 5000 steps, whereas Region VQ Loss reaches full utilization by Step 500 and maintains it throughout training. This confirms that the unguaranteed distribution alignment problem is intrinsic to the VQ loss objective, and that Region VQ Loss directly resolves it by providing every code with a principled learning target.

4.4 Decoupled Schedule

Coupled Optimization. The Encoder–Decoder and the Codebook have fundamentally different optimization characteristics and should be governed by independent learning rate schedules. The Encoder benefits from warmup-plus-annealing to stabilize its complex multi-objective landscape. The Codebook, whose task is to continuously track the evolving encoder distribution, requires sustained high learning rates especially during the early phase when the encoder output distribution changes most rapidly. Coupling both under either a constant learning rate or a warmup-plus-annealing schedule leads to suboptimal performance or reduced codebook utilization.

Objective-Driven Schedule Decoupling. Prior works [11, 2] have shown that a warmup-plus-annealing learning rate schedule benefits VQ training quality, yet it often leads to degraded codebook utilization. To compensate, FVQ introduces a more expressive shared projector to maintain utilization under this schedule. Our preceding analysis reveals that this tension stems from a more fundamental issue: the Encoder–Decoder and the Codebook have inherently different learning objectives, and therefore require distinct optimization schedules. We treat them as two independent optimization systems, each scheduled according to its own objective:

- **Encoder–Decoder** is responsible for reconstruction under discrete regularization, a complex multi-objective task that benefits from warmup-plus-annealing to stabilize early optimization and ensure smooth convergence.

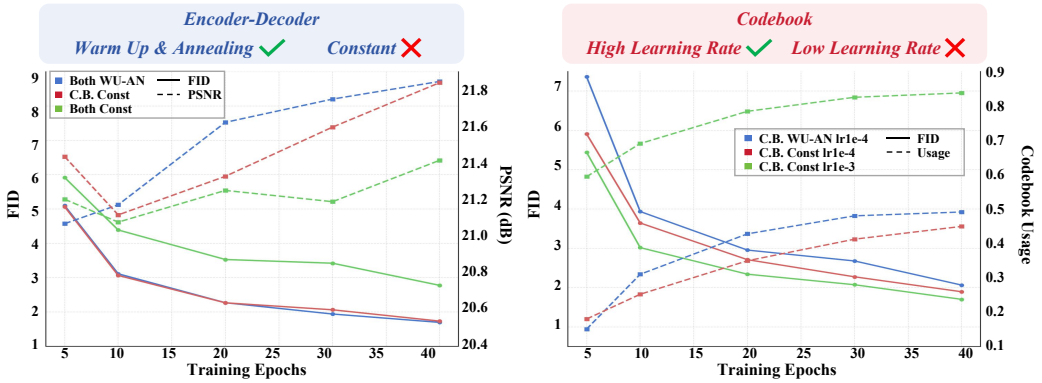


Figure 4: Pilot study on learning rate schedules for different modules. WU-AN denotes Warmup-plus-Annealing; C.B. denotes Codebook.

- **Codebook** is responsible for continuously tracking the encoder output distribution, a clean and dedicated objective for which a constant high learning rate with no warmup may be most beneficial, ensuring adequate gradient magnitude from the very first step.

Pilot Study. We conduct two controlled experiments to verify this design (Figure 4). Using the FVQ architecture (left), we ablate which module benefits from warmup-plus-annealing: setting the Codebook to a constant learning rate (red) incurs no performance degradation, whereas applying a constant rate to the Encoder–Decoder leads to a clear quality drop. Using the SimVQ architecture (right), we examine what schedule the Codebook requires: utilization is not improved by complex schedules, but benefits from a stable and sufficiently high constant learning rate.

5 Experiments

5.1 Main Results

Setup. We evaluate StableVQ on ImageNet [4] at 256×256 resolution using a VQGAN-style [6] encoder–decoder with downsampling factor $f = 16$, producing $16 \times 16 = 256$ tokens per image. We evaluate reconstruction quality by rFID and LPIPS, and codebook utilization as the fraction of activated codes, on the ImageNet validation set. We compare against a range of baselines, with particular focus on shared-projection methods SimVQ [36] and FVQ [2]. For these baselines, we re-implement their results following their respective original configurations, and evaluate using on-the-fly reconstruction rather than a save-then-reload pipeline to ensure fair and accurate metric computation. StableVQ uses the simplest single-layer linear shared projector by default.

Reconstruction results. Table 1 presents reconstruction results. While SimVQ and FVQ both achieve 100% utilization under their respective standard configurations, each comes with notable limitations. SimVQ adopts a constant learning rate to sustain full utilization, but the lack of annealing results in substantially degraded reconstruction quality. FVQ relies on a carefully engineered projector architecture—including ViT block depth and patch size—to maintain utilization under warmup-plus-annealing, and the patch embedding operation constrains the codebook size to perfect squares.

StableVQ achieves superior reconstruction quality with a single linear projection layer, matching or surpassing methods that rely on more complex projectors. It requires no projector-specific design and imposes no structural constraints, while maintaining full utilization across a broader range of challenging scenarios as the ablation studies demonstrate. When VQ training stability is no longer the bottleneck, the optimization strategy becomes the dominant factor in reconstruction quality; StableVQ uses a discriminator following prior works [24, 33] as its adversarial supervision.

Robustness test. To further test the stability of different methods, we introduce UR-AUC in Table 1, which denotes Usage Recovery AUC. It measures codebook utilization recovery under codebook-token distribution mismatch and is computed as the average AUC of codebook usage curves over the

Table 1: Reconstruction results on ImageNet 256×256 with 16×16 tokens. † denotes a larger encoder–decoder. UR-AUC denotes Usage Recovery AUC under the robustness test.

Method	Projector	Epochs	Codebook Size ($n \times d$)	rFID↓	LPIPS↓	Usage↑	UR-AUC↑
LlamaGen [27]	—	40	$16{,}384 \times 8$	2.19	0.2281	97%	—
LlamaGen [27]	—	40	$16{,}384 \times 256$	9.21	—	0.29%	
IBQ† [25]	—	330	$16{,}384 \times 256$	1.37	0.2235	96%	—
IBQ† [25]	—	330	$262{,}144 \times 256$	1.00	0.2030	84%	
Shared-Projection-Based Methods							
VQGAN-LC [35]	Linear-1	20	$16{,}384 \times 8$	3.01	0.2358	99%	—
VQGAN-LC [35]	Linear-1	20	$100{,}000 \times 8$	2.62	0.2212	99%	
SimVQ [36]	Linear-1	40	$16{,}384 \times 256$	2.89	0.2492	100%	$2.17_{\pm 0.32}$
SimVQ [36]	Linear-1	40	$262{,}144 \times 256$	3.16	0.2516	100%	
FVQ [2]	ViTBlock-2	40	$16{,}384 \times 256$	1.70	0.2176	100%	$8.08_{\pm 0.24}$
FVQ [2]	ViTBlock-2	40	$262{,}144 \times 256$	1.29	0.2003	100%	
StableVQ	Linear-1	40	$16{,}384 \times 256$	1.22	0.2235	100%	$\mathbf{60.59}_{\pm 1.52}$
StableVQ	Linear-1	40	$262{,}144 \times 256$	1.05	0.1947	100%	
StableVQ	Linear-1	120	$16{,}384 \times 256$	**1.13**	**0.2134**	100%	
StableVQ	Linear-1	120	$262{,}144 \times 256$	**0.92**	**0.1893**	100%	

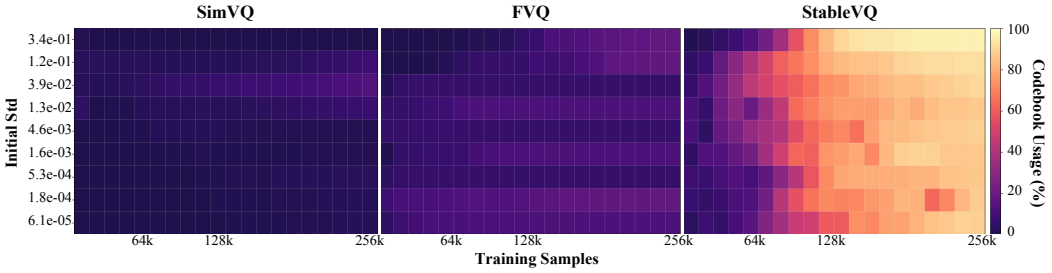


Figure 5: Codebook usage recovery under different codebook-token distribution relationships.

tested mismatch settings (Appendix E). As visualized in Figure 5, SimVQ and FVQ recover usage slowly and only under limited mismatch conditions, whereas StableVQ rapidly restores full codebook usage across diverse codebook-token distribution relationships. This indicates that StableVQ avoids prolonged low utilization and dead-code issues under different training conditions, providing strong robustness guarantees for scaling VQ training and applying it to broader scenarios.

Generation results. Following the IBQ [25] generation setup, we train class-conditional autoregressive transformers on StableVQ tokens. Table 2 shows competitive ImageNet 256×256 generation results, with complete baseline comparisons provided in Appendix C.

Table 2: Class-conditional image generation on ImageNet 256×256.

Type	Tokenizer	Generator	Param.	FID↓	IS↑	Pre.↑	Rec.↑
Vanilla AR	IBQ [25]	IBQ-B [25]	342M	2.88	254.7	**0.84**	0.51
Vanilla AR	IBQ [25]	IBQ-L [25]	649M	2.45	**267.5**	**0.83**	0.52
Vanilla AR	StableVQ	IBQ-B [25]	342M	**2.35**	**256.0**	0.82	**0.58**
Vanilla AR	StableVQ	IBQ-L [25]	649M	**2.18**	250.4	0.82	**0.59**

5.2 Ablation Studies

The main results above are obtained under each method's standard setting, where methods such as SimVQ and FVQ can also reach full utilization. However, these methods still do not resolve the fundamental deficiencies analyzed in Section 4. We therefore turn to more challenging yet common settings to evaluate training robustness.

Table 3: Ablation on proposed strategies under the codebook expansion setting.

Region VQ	Dyn. STE	De. Sch.	Peak Commit	Util.↑	rFID↓	LPIPS↓	PSNR↑	SSIM↑
			53.11	49.13%	2.06	0.2297	21.65	0.5730
✓			>200 (NaN)	—	—	—	—	—
	✓		<0.1	1.27%	6.68	0.3100	19.89	0.5026
		✓	<0.1	83.77%	1.71	0.2209	21.79	0.5864
	✓	✓	<0.1	21.66%	2.07	0.2354	21.36	0.5665
✓	✓		<0.1	**100%**	1.72	**0.2200**	21.84	0.5854
✓		✓	<0.1	**100%**	1.75	0.2206	**21.85**	0.5860
✓	✓	✓	<0.1	**100%**	**1.70**	0.2208	21.81	**0.5879**

Table 4: Ablation on methods under the codebook shrinkage setting.

Projector	Region VQ	Init	Util.↑	rFID↓	PSNR↑	SSIM↑	LPIPS↓
ViTBlock-2 [2]		uniform	18.75%	2.4355	20.9789	0.5677	0.2414
ViTBlock-2 [2]		gaussian	62.5%	1.9530	21.3774	0.5836	0.2273
ViTBlock-2 [2]	✓	uniform	**100%**	**1.8176**	**21.5497**	0.5896	**0.2210**
ViTBlock-2 [2]	✓	gaussian	**100%**	1.8966	21.4603	**0.5931**	0.2255

Ablation 1: Codebook expansion. We first set the experiment to the case of Figure 1(a), where the Codebook is initialized within a very small numerical range—the most common initialization trick in VQ training. In this setting, we use a single linear layer as the shared projector and adopt a warmup-plus-annealing schedule with peak learning rate $1e{-4}$. We report peak commitment loss, codebook utilization, and reconstruction metrics, and evaluate different combinations of the strategies proposed in Section 4 on top of the baseline for a more complete analysis. When Decoupled Schedule is used, the Codebook learning rate is set to a constant $1e{-3}$.

The results are consistent with the analysis in Section 4. Region VQ Loss alone causes NaN collapse because the Codebook still updates too slowly under the shared warmup schedule. Dynamic STE suppresses commitment-loss spikes and stabilizes the Encoder, but utilization remains very low, revealing that conventional training relies on unstable encoder oscillations to activate codes. Decoupled Schedule improves distribution tracking from the start, yet still falls short of full utilization under the standard VQ loss. Once Region VQ Loss is combined with either Dynamic STE or Decoupled Schedule, full utilization is recovered with strong reconstruction quality. Using all three components gives the most complete solution, jointly ensuring Encoder stability, dense Codebook supervision, and fast distribution tracking.

Ablation 2: Codebook shrinkage. We further set the experiment to the case of Figure 1(b), where the Codebook occupies a large range. In VQ training, this commonly arises when the space is constrained by ℓ_2 normalization: after normalization, either uniform or gaussian initialization distributes codes broadly across the space, whereas encoder outputs, reflecting the statistics of natural images, concentrate in a much smaller region. This naturally creates a codebook shrinkage scenario. In this experiment, we set the codebook dimension to 1024 and use a stronger ViTBlock shared projector, following the FVQ configuration. The results show that FVQ fails to achieve full utilization under either codebook initialization, and its utilization is strongly affected by initialization. Adding Region VQ consistently reaches full utilization and improves reconstruction quality regardless of initialization. This further supports that StableVQ provides a more principled solution, enabling ideal codebook usage across different VQ training conditions.

5.3 Comparison with Prior STE Corrections

VQ-STE++ [11] employs Alternating Optimization, which is similar in motivation to Dynamic STE: both aim to suppress unreliable task gradients when quantization error is large. However, VQ-STE++ relies on alternating inner and outer updates, introducing several sensitive hyperparameters and additional training overhead. It removes the encoder commitment loss and assumes an initially aligned code–token distribution established through k-means initialization, making optimization sensitive when this alignment is absent.

The Rotation Trick [9] transforms the encoder gradient through a rotation matrix R and a rescaling factor $\|q\|/\|z\|$. For rotation, R redirects the backward gradient according to the angle between token z and code q. Although elegant, it provides no rigorous guarantee of a more accurate gradient and may instead redirect the gradient away from the desired direction. For rescaling, the factor suppresses the encoder gradient when $\|q\| < \|z\|$, but amplifies it when $\|q\| > \|z\|$, even for distant token–code pairs. Dynamic STE instead uses relative within-batch quantization distances to attenuate unreliable gradients and never amplify them. Moreover, under ℓ_2 normalization, the rescaling factor becomes one and loses its attenuation effect, whereas Dynamic STE remains active.

We conduct a controlled 15-epoch experiment with Region VQ and all other experimental settings fixed. As shown in Table 5, VQ-STE++ consistently exhibits a pronounced collapse in codebook utilization without the joint use of k-means pre-alignment and a norm bottleneck. In contrast, both the Rotation Trick and Dynamic STE achieve full codebook utilization with the support of Region VQ, while Dynamic STE yields substantially better reconstruction performance across metrics.

Table 5: Comparison of STE corrections in training stability and reconstruction quality.

Method	Training Status	Usage↑	rFID↓	PSNR↑	LPIPS↓
VQ-STE++ [11]	Collapsed	–	–	–	–
Rotation Trick [9]	Completed	100.00%	3.6437	20.4024	0.2568
Dynamic STE (ours)	Completed	100.00%	**2.7942**	**21.1882**	**0.2359**

5.4 Comparison with Explicit Distribution-Alignment Methods

Wasserstein VQ [7] and MMD VQ [8] formulate global distribution matching as a training loss, using Gaussian moments and kernel statistics, respectively. Region VQ Loss instead addresses the asymmetric supervision of standard VQ by assigning persistently inactive codes explicit, local targets propagated from statistically supported active regions. It therefore requires no parametric assumption about the token distribution and avoids the costly pairwise kernel computation of MMD VQ.

We evaluate these methods on the non-Gaussian mixture benchmark introduced by MMD VQ [8]. At $\zeta = 0$, the target distribution reduces to a single Gaussian; increasing ζ separates the two mixture modes and strengthens its non-Gaussian structure. Following its setting, we use 16,384 codes of dimension 8, sample 20K tokens per step, and report codebook utilization at 10K steps. We additionally measure the per-step training time of all methods under the same environment. To mitigate under-coverage caused by heavily overlapping recipient sets in this synthetic benchmark, we automatically enlarge the propagation quotas when recipient collisions are severe.

Table 6 reports the resulting codebook utilization across different ζ values and the corresponding per-step training time. As the target distribution becomes strongly non-Gaussian, Wasserstein VQ and MMD VQ fall to 34.8% and 75.6% utilization at $\zeta = 4$, respectively, whereas Region VQ maintains 99.4%. Region VQ also remains close to Wasserstein VQ in training time and is $34.2\times$ faster than MMD VQ. These results demonstrate that Region VQ combines robust codebook utilization across different distribution relationships with low training overhead.

Table 6: Codebook utilization and training efficiency on the non-Gaussian benchmark.

Method	$\zeta = 0$	$\zeta = 1$	$\zeta = 2$	$\zeta = 3$	$\zeta = 4$	Time (ms)	Relative Time
Wasserstein VQ [7]	99.9%	97.0%	62.7%	44.8%	34.8%	6.98	$1.00\times$
MMD VQ [8]	99.9%	99.8%	92.5%	85.7%	75.6%	295.57	$42.36\times$
Region VQ (ours)	99.8%	99.3%	93.2%	99.4%	99.4%	8.64	$1.24\times$

6 Conclusion

StableVQ shows that the long-standing instability of VQ training is not a limitation of vector quantization itself, but a consequence of entangled optimization objectives. By restoring *separation of concerns* between the Encoder–Decoder and the Codebook, StableVQ turns full codebook utilization from a fragile heuristic outcome into a principled property that can be directly guaranteed. We believe this provides an important foundation for extending VQ to broader scenarios and more challenging applications, where robust training is essential.

Acknowledgments and Disclosure of Funding

This work was supported by the National Natural Science Foundation of China (Grant No. U25B2067) and KlingAI Research.

References

[1] 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.

[2] Yifan Chang, Jie Qin, Limeng Qiao, Xiaofeng Wang, Zheng Zhu, Lin Ma, and Xingang Wang. Scalable training for vector-quantized networks with 100% codebook utilization. *arXiv preprint arXiv:2509.10140*, 2025.

[3] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *International conference on machine learning*, pages 1691–1703. PMLR, 2020.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[5] Sinan Du, Jiahao Guo, Bo Li, Shuhao Cui, Zhengzhuo Xu, Yifu Luo, Yongxian Wei, Kun Gai, Xinggang Wang, Kai Wu, et al. Vqrae: Representation quantization autoencoders for multimodal understanding, generation and reconstruction. *arXiv preprint arXiv:2511.23386*, 2025.

[6] 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.

[7] Xianghong Fang, Litao Guo, Hengchao Chen, Yuxuan Zhang, Xiaofan Xia, Dingjie Song, Yexin Liu, Hao Wang, Harry Yang, Yuan Yuan, and Qiang Sun. Enhancing vector quantization with distributional matching: A theoretical and empirical study. *arXiv preprint arXiv:2506.15078*, 2025.

[8] Xianghong Fang, Yuan Yuan, Dehan Kong, and Tim G. J. Rudner. Vq-transplant: Efficient vq-module integration for pre-trained visual tokenizers. In *International Conference on Learning Representations*, 2026.

[9] Christopher Fifty, Ronald G. Junkins, Dennis Duan, Aniketh Iyengar, Jerry W. Liu, Ehsan Amid, Sebastian Thrun, and Christopher Ré. Restructuring vector quantization with the rotation trick. In *International Conference on Learning Representations*, 2025.

[10] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[11] Minyoung Huh, Brian Cheung, Pulkit Agrawal, and Phillip Isola. Straightening out the straight-through estimator: Overcoming optimization challenges in vector quantized networks. In *International Conference on Machine Learning*, pages 14096–14113. PMLR, 2023.

[12] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[13] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11523–11532, 2022.

[14] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2142–2152, 2023.

[15] 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.

[16] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[17] 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.

[18] Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: Vq-vae made simple. *arXiv preprint arXiv:2309.15505*, 2023.

[19] Ziqi Pang, Tianyuan Zhang, Fujun Luan, Yunze Man, Hao Tan, Kai Zhang, William T Freeman, and Yu-Xiong Wang. Randar: Decoder-only autoregressive visual generation in random orders. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 45–55, 2025.

[20] 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.

[21] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International conference on machine learning*, pages 8821–8831. Pmlr, 2021.

[22] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019.

[23] 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.

[24] Axel Sauer, Tero Karras, Samuli Laine, Andreas Geiger, and Timo Aila. Stylegan-t: Unlocking the power of gans for fast large-scale text-to-image synthesis. In *International conference on machine learning*, pages 30105–30118. PMLR, 2023.

[25] Fengyuan Shi, Zhuoyan Luo, Yixiao Ge, Yujiu Yang, Ying Shan, and Limin Wang. Scalable image tokenization with index backpropagation quantization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16037–16046, 2025.

[26] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[27] 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.

[28] 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.

[29] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. *Advances in neural information processing systems*, 29, 2016.

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

[31] 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. *arXiv preprint arXiv:2110.04627*, 2021.

[32] Lijun Yu, José Lezama, Nitesh B Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Vighnesh Birodkar, Agrim Gupta, Xiuye Gu, et al. Language model beats diffusion–tokenizer is key to visual generation. *arXiv preprint arXiv:2310.05737*, 2023.

[33] Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. *arXiv preprint arXiv:2510.11690*, 2025.

[34] Chuanxia Zheng and Andrea Vedaldi. Online clustered codebook. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22798–22807, 2023.

[35] 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%. *Advances in Neural Information Processing Systems*, 37: 12612–12635, 2024.

[36] Yongxin Zhu, Bocheng Li, Yifei Xin, Zhihua Xia, and Linli Xu. Addressing representation collapse in vector quantized models with one linear layer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22968–22977, 2025.

A Related Work

Vector Quantization. Vector-quantized representation learning is introduced by VQ-VAE [30], which maps continuous encoder features to discrete code indices through nearest-neighbor lookup in a learned codebook. VQ-VAE-2 [22] improves this framework with hierarchical latent maps, and VQGAN [6] combines vector quantization with perceptual and adversarial objectives, making discrete visual tokenizers a standard interface for high-fidelity image synthesis. A large body of work improves reconstruction quality and token capacity by modifying the quantization structure, including residual or multi-stage quantization methods such as RQ-VAE [13] and ViT-based tokenizer architectures such as ViT-VQGAN [31]. Another line addresses codebook collapse and low utilization, which become increasingly severe when scaling codebook size or embedding dimension; representative solutions include codebook reset and replacement strategies [34], low-dimensional code embeddings and normalization [31], and soft-assignment-based training strategies such as stochastic or Gumbel-softmax quantization [21, 12] and IBQ [25], which use relaxed or soft-to-hard categorical paths to improve codebook gradients beyond the selected hard code. Scalar-quantization-based methods, including FSQ [18] and LFQ [32], replace learned vector-codebook lookup with quantization over scalar or binary values, simplifying optimization and improving usage at scale while introducing different capacity trade-offs. Shared-projection methods such as VQ-STE++ [11], SimVQ [36], and FVQ [2] reparameterize code vectors through a shared function, providing an elegant and lightweight way to address the long-standing low-utilization problem in VQ training. Recently, pretrained vision foundation models have also been used to improve visual tokenizers, for example by initializing, regularizing, or aligning tokenizer representations with vision-foundation-model features [35, 5], which can strengthen semantic representation quality but is complementary to the training-stability problem studied in this work. StableVQ builds on the simple shared-projection foundation, but revisits the optimization responsibilities of the Encoder–Decoder and Codebook and targets the latent instability that remains even when codebook utilization has been substantially improved.

Image Generation. Image generation has been developed along both continuous and tokenized modeling paradigms. Early autoregressive models such as PixelCNN [29] and iGPT [3] model images directly in pixel space, but their sequential generation cost and weak compression make high-resolution synthesis difficult. Discrete tokenizers alleviate this bottleneck by converting images into compact latent token sequences: VQGAN [6] applies transformer-based autoregressive modeling in the VQ latent space, while VQ-VAE2 [22], RQ-Transformer [13], and related residual or hierarchical token models further explore multi-level discrete representations. Non-autoregressive and masked-token generators such as MaskGIT [1], MAGE [14], and MAGVIT-v2 [32] predict missing visual tokens and refine them iteratively, improving sampling efficiency and demonstrating the importance of tokenizer quality for generation. More recently, large-scale autoregressive image generators such as LlamaGen [27], VAR [28], RandAR [19], and Open-MAGVIT2 [17] show that language-model-style next-token, next-scale, or randomized-order prediction can achieve strong visual synthesis when paired with expressive visual tokens. In parallel, diffusion, score-based, and flow-matching models [10, 26, 16, 23, 20] generate images through continuous denoising or transport processes, and hybrid alternatives such as MAR [15] reduce or remove the dependence on hard vector quantization. These advances make the tokenizer a critical upstream component: regardless of whether the downstream generator is autoregressive, masked, multi-scale, or hybrid, unstable VQ training can limit reconstruction fidelity, code utilization, and ultimately generation quality. StableVQ is therefore orthogonal to generator design and aims to provide a more reliable discrete representation substrate for token-based image generation.

B Limitations

StableVQ focuses on making VQ tokenizer training stable and reliable, providing a foundation on which stronger training objectives and downstream modeling choices can be explored. A natural direction is to study training recipes that better balance reconstruction quality, semantic structure, and generation-friendliness once codebook utilization and optimization stability are no longer the main bottlenecks. Another promising direction is to connect stable visual tokenization with unified generation and understanding objectives, where discrete tokens may need to preserve both low-level fidelity and high-level semantic information. Finally, while our experiments focus on image

tokenizers, the same separation-of-concerns perspective may be useful in other domains that rely on vector quantization, such as video, audio, multimodal representation learning, or compression.

C Additional Experimental Results

We provide additional baselines and experimental results for reference. For reconstruction, Table 7 supplements the main results with metrics from methods not covered in the main text, such as VQGAN and MaskGIT. We also provide more complete metrics for selected baselines under the same evaluation script in Table 8, including PSNR, SSIM, and other reference metrics.

Table 7: Complete reconstruction results on ImageNet 256×256. Codebook Size is reported as $n \times d$ (number of codes $\times$ channel dimension). † denotes a larger encoder–decoder trained for up to 330 epochs. ‡ denotes training for 120 epochs.

Method	Projector	Tokens	Codebook Size ($n \times d$)	rFID↓	LPIPS↓	Usage↑
VQGAN [6]	—	16×16	$1{,}024 \times 256$	7.94	—	44%
VQGAN [6]	—	16×16	$16{,}384 \times 256$	4.98	0.2843	5.9%
SD-VQGAN [23]	—	16×16	$16{,}384 \times 8$	5.15	—	—
MaskGIT [1]	—	16×16	$1{,}024 \times 256$	2.28	—	—
LlamaGen [27]	—	16×16	$16{,}384 \times 8$	2.19	0.2281	97%
LlamaGen [27]	—	16×16	$16{,}384 \times 256$	9.21	—	0.29%
IBQ† [25]	—	16×16	$16{,}384 \times 256$	1.37	0.2235	96%
IBQ† [25]	—	16×16	$262{,}144 \times 256$	1.00	0.2030	84%
Shared-Projection-Based Methods						
VQGAN-LC [35]	Linear-1	16×16	$16{,}384 \times 8$	3.01	0.2358	99%
VQGAN-LC [35]	Linear-1	16×16	$100{,}000 \times 8$	2.62	0.2212	99%
SimVQ [36]	Linear-1	16×16	$16{,}384 \times 256$	2.89	0.2492	100%
SimVQ [36]	Linear-1	16×16	$262{,}144 \times 256$	3.16	0.2516	100%
FVQ [2]	ViTBlock-2	16×16	$16{,}384 \times 256$	1.70	0.2176	100%
FVQ [2]	ViTBlock-2	16×16	$262{,}144 \times 256$	1.29	0.2003	100%
StableVQ	Linear-1	16×16	$16{,}384 \times 256$	1.22	0.2235	100%
StableVQ	Linear-1	16×16	$262{,}144 \times 256$	1.05	0.1947	100%
StableVQ‡	Linear-1	16×16	$16{,}384 \times 256$	**1.13**	**0.2134**	100%
StableVQ‡	Linear-1	16×16	$262{,}144 \times 256$	**0.92**	**0.1893**	100%

Table 8: Additional reconstruction metrics under the same evaluation script on ImageNet 256×256.

Method	Codebook Size ($n \times d$)	Epochs	rFID↓	PSNR↑	SSIM↑	LPIPS↓	Usage↑
SimVQ [36]	$16{,}384 \times 256$	40	2.89	21.36	0.5551	0.2492	100%
SimVQ [36]	$262{,}144 \times 256$	40	3.16	21.31	0.5527	0.2516	100%
FVQ [2]	$16{,}384 \times 256$	40	1.70	21.85	0.5894	0.2176	100%
FVQ [2]	$262{,}144 \times 256$	40	1.29	22.50	0.6146	0.2003	100%
FVQ [2]	$16{,}384 \times 256$	120	1.46	21.91	0.5951	0.2156	100%
FVQ [2]	$262{,}144 \times 256$	120	1.07	22.55	0.6244	0.1965	100%
StableVQ	$16{,}384 \times 256$	40	1.22	21.84	0.5816	0.2235	100%
StableVQ	$262{,}144 \times 256$	40	1.05	22.77	0.6277	0.1947	100%
StableVQ	$16{,}384 \times 256$	120	**1.13**	**22.01**	**0.5965**	**0.2134**	100%
StableVQ	$262{,}144 \times 256$	120	**0.92**	**22.81**	**0.6345**	**0.1893**	100%

For downstream generation, we train class-conditional autoregressive transformers following IBQ [25] on top of StableVQ tokens and evaluate on ImageNet 256×256 using FID, IS, precision, and recall. As shown in Table 9, StableVQ achieves competitive generation quality, validating that improved reconstruction quality translates to gains in downstream generation.

Table 9: Class-conditional image generation results on ImageNet 256×256.

Type	Tokenizer	Generator	Param.	FID↓	IS↑	Pre.↑	Rec.↑
Diff.	SD-VAE [23]	DiT-L/2 [20]	458M	5.02	167.2	0.75	0.57
Diff.	SD-VAE [23]	DiT-XL/2 [20]	675M	2.27	278.2	0.83	0.57
Mask.	VQGAN [6]	MaskGIT [1]	227M	6.18	182.1	0.80	0.51
VAR	VAR [28]	VAR-d16 [28]	310M	3.30	274.4	0.84	0.51
VAR	VAR [28]	VAR-d20 [28]	600M	2.57	302.6	0.83	0.56
AR	LlamaGen [27]	LlamaGen-L [27]	343M	3.80	248.3	0.83	0.51
AR	LlamaGen [27]	LlamaGen-XL [27]	775M	3.39	227.1	0.81	0.54
AR	FVQ [2]	LlamaGen-L [27]	343M	2.39	276.6	0.84	0.56
AR	FVQ [2]	LlamaGen-XL [27]	775M	2.07	287.0	0.83	0.58
AR	IBQ [25]	IBQ-B [25]	342M	2.88	254.7	0.84	0.51
AR	IBQ [25]	IBQ-L [25]	649M	2.45	267.5	0.83	0.52
AR	StableVQ	IBQ-B [25]	342M	2.35	256.0	0.82	0.58
AR	StableVQ	IBQ-L [25]	649M	2.18	250.4	0.82	0.59

D Additional Analysis Experiments

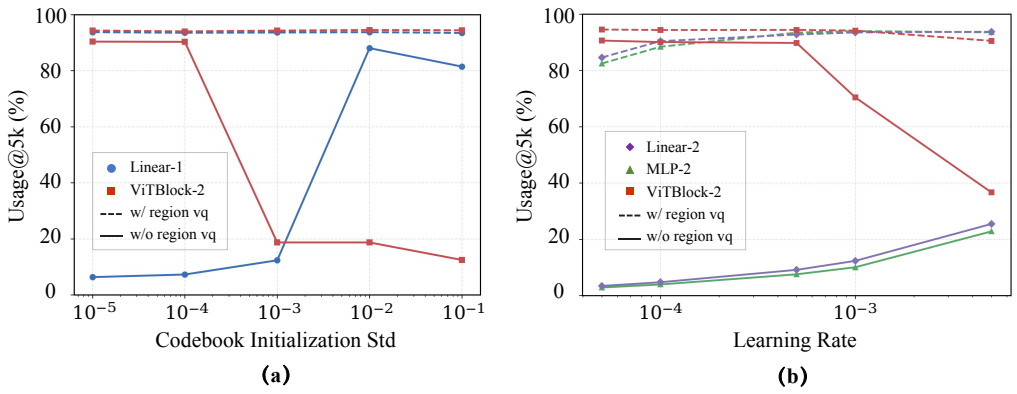


Figure 6: Additional analysis experiments. (a) Usage@5k under different Gaussian codebook initialization scales. (b) Usage@5k under different learning rates.

D.1 Effect of Codebook Initialization

Codebook initialization has long been an important practical trick in conventional VQ training, because an inappropriate initialization scale can lead to poor codebook utilization. A common strategy is to initialize the Codebook within a small numerical range, which often helps utilization increase more stably. However, this trick is not always applicable: for example, when an ℓ_2 normalization constraint is imposed, the effective code distribution can no longer be controlled simply by shrinking the raw initialization range. Moreover, an overly small initialization scale may also prolong the usage-recovery phase, reducing training efficiency.

To study how initialization affects codebook utilization across configurations, we follow the pilot setting in Section 4.3: the Encoder is frozen, and only the Codebook is optimized to fit a fixed target distribution. We vary the Gaussian initialization scale of the Codebook and report codebook utilization at 5k steps, computed within a window of 65,536 tokens. As shown in Figure 6(a), when the initialization scale is too small, the linear shared projector used in SimVQ-style settings recovers utilization slowly; when the initialization scale becomes larger, the ViT-based shared projector used in FVQ-style settings faces a clear dead-code risk. Replacing the standard VQ loss with Region VQ Loss substantially improves both cases, allowing utilization to approach full usage at 5k steps across initialization scales. This verifies that StableVQ reduces the dependence of VQ training on codebook initialization design, and also suggests robustness to scale divergence during training.

D.2 Effect of Shared Projector

For methods that rely on a shared projector to improve codebook utilization, projector design has recently become an important practical consideration. When the projector is the simplest linear layer, its limited learning capacity can make it difficult for the Codebook to quickly track the encoder output distribution, often leading to slow utilization growth and more frequent scale-divergence events during training. In contrast, using a more expressive ViT block can improve tracking capacity, but introduces additional architecture-specific design cost and constraints on codebook size.

To examine how shared projector design affects utilization, we again follow the frozen-Encoder pilot setting and measure codebook utilization at 5k steps under different learning rates, using the same 65,536-token window. The Codebook is initialized with a small Gaussian standard deviation of 10^{-4}, so the results more directly reflect projector learning capacity while reducing the influence of dead-code effects. As shown in Figure 6(b), neither a two-layer linear projector nor a two-layer MLP can raise utilization to a high level within 5k steps, even when the learning rate is increased to 5×10^{-3}. The two-layer ViT block performs better at moderate learning rates, but its utilization drops sharply when the learning rate becomes too large. With Region VQ Loss, however, different projector structures all achieve strong utilization within 5k steps, and their usable learning-rate range is substantially widened. This suggests that the Codebook benefits from an independent learning-rate schedule for tracking the encoder distribution, but also shows that changing projector capacity alone is insufficient for stable and rapid utilization growth. The more critical factor is to provide the Codebook with a principled learning objective, which allows StableVQ to improve utilization across shared-projector designs and reduces the dependence of robust VQ training on carefully engineered projector structures.

D.3 Computational and Memory Overhead

Table 10 reports training-time and memory profiles for the baseline and StableVQ with 16K and 262K codebooks, measured on the same machine for a direct comparison. The Decoupled Schedule only changes the learning-rate assignment and adds no forward or backward operations, while the measured timing difference for Dynamic STE is within noise. Region VQ Loss incurs additional work when propagating targets to persistently inactive codes; as codebook utilization increases, fewer codes require this supervision and the associated cost decreases. We therefore report its training cost at multiple utilization levels to capture this dependence, with consistently low additional overhead. All three components affect training only: the learned tokenizer retains the same quantization and decoding procedure at inference.

Table 10: Training-time and memory overhead of StableVQ components.

	Training Cost					Memory Usage	
Codebook	Baseline (ms)	Dynamic STE (ms)	Region VQ @30% (ms, Δ)	Region VQ @50% (ms, Δ)	Region VQ @90% (ms, Δ)	Baseline (GiB)	Region VQ (GiB, Δ)
16K	438.1	437.2	441.2 ($+0.71\%$)	440.4 ($+0.51\%$)	440.2 ($+0.46\%$)	34.06	34.07 ($+0.04\%$)
262K	462.2	462.7	472.2 ($+2.16\%$)	469.2 ($+1.50\%$)	466.2 ($+0.85\%$)	35.12	35.55 ($+1.24\%$)

E Robustness Test Illustration

The robustness test is designed to evaluate whether a VQ training method can recover high codebook utilization when the relationship between the Codebook distribution and the token distribution changes. This setting complements the final validation usage reported in the main reconstruction table: a method may eventually report high utilization under its standard configuration, yet still recover slowly or fail when the codebook-token relationship becomes less favorable. We control this distributional relationship through different codebook initializations.

We construct different codebook-token distribution relationships by varying the Gaussian initialization scale of the codebook vector base while keeping the data, architecture, optimizer, learning-rate schedule, and training budget fixed. Changing this scale alters the initial geometry between the projected code vectors and the encoder token distribution: small scales place code vectors in a compact region, whereas larger scales spread the code distribution over a broader region relative to

the token distribution. This provides a controlled way to test codebook usage recovery under multiple mismatch settings without changing the input data or reconstruction objective.

For each method and each mismatch setting, we train the tokenizer for the same early-stage budget and record codebook usage throughout training. Usage is computed within a window of 65,536 tokens, so the reported values are slightly lower than utilization measured over the full validation set. Figure 5 visualizes these trajectories as heatmaps: each row corresponds to one mismatch setting, and brighter colors indicate higher codebook usage. A stable method should recover high usage quickly across most rows rather than depending on a narrow range of favorable initial relationships.

We summarize the heatmaps with UR-AUC, which denotes Usage Recovery AUC. Let $U_m(t)$ be the codebook usage percentage at training step t under mismatch setting m. We compute the normalized area under each usage curve and average over the tested mismatch settings:

$$\text{UR-AUC} = \frac{1}{|\mathcal{M}|} \sum_{m \in \mathcal{M}} \frac{1}{T - t_0} \int_{t_0}^{T} U_m(t)\, dt. \tag{8}$$

In practice, we use the logged usage values and compute the integral with the trapezoidal rule. Higher UR-AUC indicates faster and more consistent recovery of codebook utilization under codebook-token distribution mismatch. The error bars for UR-AUC in Table 1 are computed as the sample mean and sample standard deviation over three runs with different random seeds, where each run includes the same set of initialization-induced mismatch settings described above.

F Region VQ Algorithm

Algorithm 1 summarizes the Region VQ procedure in one training step. It follows the design in Section 4.3: current assignments determine active codes and their targets, a FIFO queue identifies persistently inactive codes, and targets from sufficiently reliable active codes are propagated to nearby inactive ones before computing the codebook loss.

Algorithm 1 Region VQ in one training step

Require: Token features $\mathbf{Z} = \{\mathbf{z}_u\}_{u=1}^{N}$, projected codebook $\mathbf{E} = \{\mathbf{e}_k\}_{k=1}^{K}$, current assignments $a_u \in [K]$, FIFO queue Q of length W
Ensure: Code targets $\{\mathbf{t}_k\}_{k=1}^{K}$ and effective code set $\mathcal{M}_t$ for codebook loss

Append current assignment indices $\{a_u\}_{u=1}^{N}$ to Q
Let $\mathcal{S}_t \leftarrow \{a_u\}_{u=1}^{N}$ be the currently active codes
Let $\mathcal{A}_t \leftarrow \bigcup Q$ be the codes active within the FIFO window
Let $\mathcal{N}_t \leftarrow [K] \setminus \mathcal{A}_t$ be the persistently inactive codes
Initialize $\mathbf{t}_k \leftarrow \mathbf{e}_k$ for all $k \in [K]$ ▷ Self-target by default
Initialize effective code set $\mathcal{M}_t \leftarrow \emptyset$
for each $k \in \mathcal{S}_t$ **do**
    $\mathcal{U}_k \leftarrow \{u : a_u = k\}, \quad n_k \leftarrow |\mathcal{U}_k|$
    $\mathbf{t}_k \leftarrow \frac{1}{n_k} \sum_{u \in \mathcal{U}_k} \mathbf{z}_u$
    Add k to $\mathcal{M}_t$
end for
Let $\mathcal{S}_t^{+} \leftarrow \{k \in \mathcal{S}_t : n_k > 1\}$
for each $k \in \mathcal{S}_t^{+}$ **do**
    $q_k \leftarrow \left\lceil n_k |\mathcal{N}_t| \,/\, \sum_{\ell \in \mathcal{S}_t^{+}} n_\ell \right\rceil$
    Select the q_k nearest codes to k from $\mathcal{N}_t$ in projected space and denote them by $\mathcal{R}_k$
end for
for each $j \in \mathcal{N}_t$ that is selected by at least one active code **do**
    Let $\mathcal{K}_j \leftarrow \{k \in \mathcal{S}_t^{+} : j \in \mathcal{R}_k\}$
    $\mathbf{t}_j \leftarrow \frac{1}{|\mathcal{K}_j|} \sum_{k \in \mathcal{K}_j} \mathbf{t}_k$
    Add j to $\mathcal{M}_t$
end for
Compute codebook loss only on effective codes: $\mathcal{L}_{\text{code}} = \frac{1}{|\mathcal{M}_t|} \sum_{k \in \mathcal{M}_t} \|\mathbf{e}_k - \text{sg}[\mathbf{t}_k]\|_2^2$

G Experimental Details

We provide the detailed configurations used for reconstruction training in Table 11. For the generation experiments in Appendix C, we follow the corresponding IBQ training setting and train for approximately 350 epochs. Other experiments in this paper describe their key differences from the standard settings in their respective contexts.

Table 11: Main experiment configurations.

Config	SimVQ	FVQ	StableVQ
Base Batch Size	128	128	128
Training Epochs	40	40 / 120	40 / 120
Optimizer	Adam	Adam	Adam
Optimizer Parameters	$\beta_1 = 0.9, \beta_2 = 0.95$	$\beta_1 = 0.9, \beta_2 = 0.95$	$\beta_1 = 0.9, \beta_2 = 0.95$
Base Learning Rate	1e−4	1e−4	1e−4
Codebook Init Type	Gaussian	Uniform	Uniform
LR Warmup	0%	10%, 0.005× to 1×	10%, 0.005× to 1×
LR Plateau	100%, 1×	27%, 1×	27%, 1×
LR Annealing	0%	63%, 1× to 0.01×	63%, 1× to 0.01×
Codebook LR	Same as LR	Same as LR	Constant 1e−3

H Visualization Results

We provide qualitative visualization results for reconstruction and downstream generation. Figure 7 compares reconstruction results with a 16k-code codebook, Figure 8 compares reconstruction results with a 262k-code codebook, and Figure 9 shows samples from the downstream generation task.

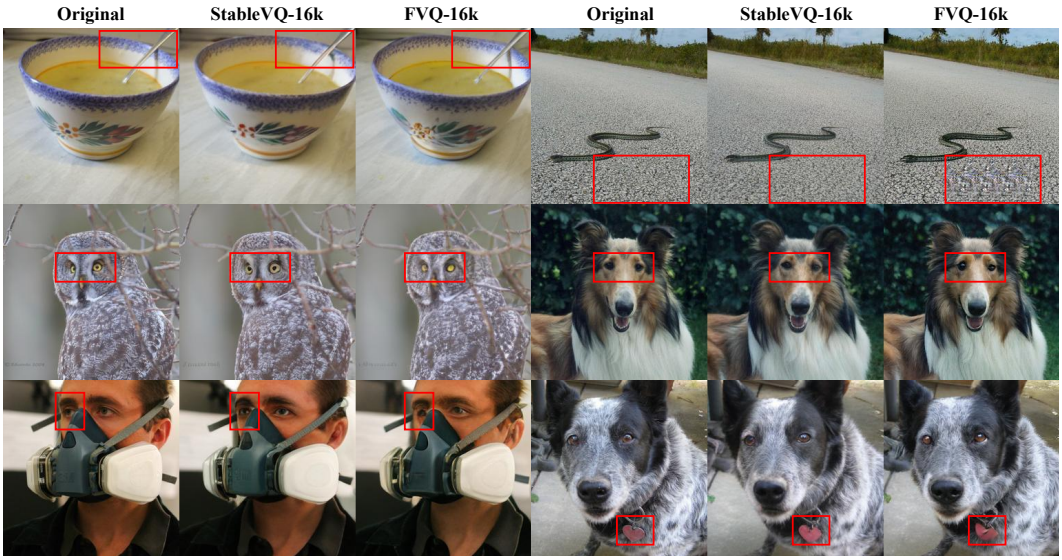


Figure 7: Qualitative reconstruction comparison with a 16k codebook.

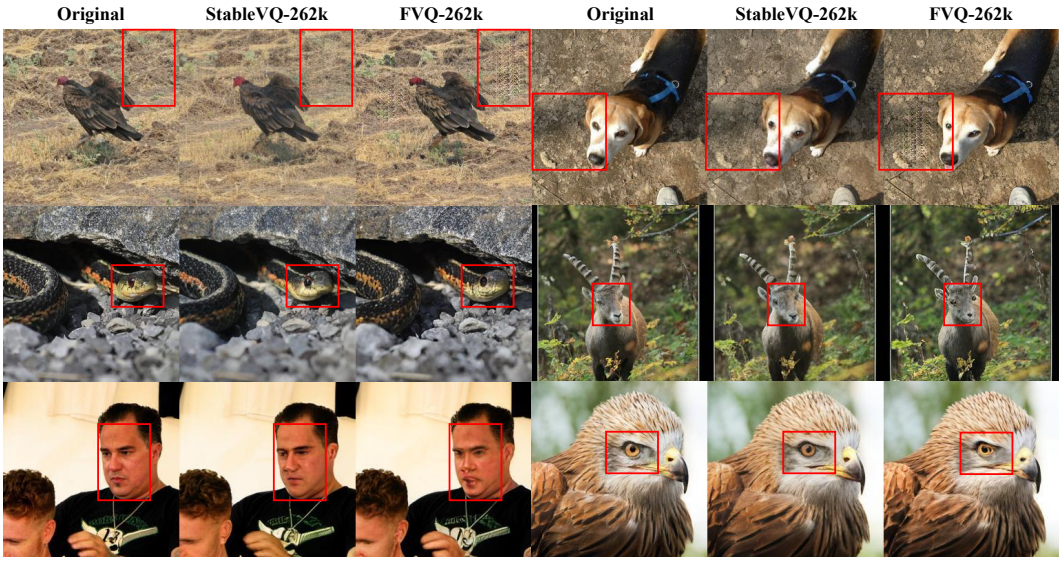


Figure 8: Qualitative reconstruction comparison with a 262k codebook.



Figure 9: Qualitative samples from the downstream class-conditional image generation task.

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 state the separation-of-concerns analysis, the three StableVQ components, and the experimental scope on ImageNet visual tokenizers. The experimental claims are supported by the main results, ablations, robustness test, and appendix analyses.

   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: Appendix B discusses the current scope and limitations, including the focus on image tokenizers and future work on reconstruction-generation trade-offs and other modalities.

   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: The paper does not present formal theoretical results, theorems, or proofs. The equations and algorithm specify the proposed training objectives and procedures.

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: The method is specified in Section 4, with Region VQ summarized in Algorithm 1. The experimental setup, robustness test, and main configurations are described in Section 5 and Appendix G.

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: A public code release is not included with this version. The experiments use ImageNet, a widely used public benchmark dataset, and the paper provides methodological and configuration details to support reproduction.

   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: Section 5 specifies the dataset, resolution, tokenizer setting, metrics, and key ablation settings. Appendix G gives the main reconstruction configurations, including batch size, training epochs, learning-rate schedules, codebook learning rate, initialization, and optimizer setting.

   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: [Yes].

   Justification: For the robustness-related metric that supports one of the paper's main claims, Table 1 reports the mean and standard deviation of UR-AUC over tests run with different random seeds. This robustness test itself further evaluates codebook usage recovery across

different codebook-token distribution mismatch conditions induced by codebook initialization. For full tokenizer training and downstream generation, we do not report run-to-run error bars due to the high computational cost.

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: Appendix G provides the training configurations, and Appendix D.3 reports the additional training-time and memory overhead of StableVQ. The overall computational budget is comparable to that of prior work; however, a complete account of the computational resources used is not provided.

   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: The work uses standard public benchmark data and proposes training algorithms for visual tokenizers. We are not aware of any aspect of the research that deviates from the NeurIPS Code of Ethics.

   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: [N/A].

    Justification: The work is a foundational algorithmic improvement for VQ tokenizer training and uses public benchmark data. It does not introduce a deployed system, new dataset, or application-specific capability with direct societal impact.

    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: The paper does not release a high-risk pretrained image generator, scraped dataset, or user-facing system. It studies training methods for visual tokenizers on ImageNet.

    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: The paper uses standard benchmark data and existing methods for comparison, and all related datasets, baselines, and prior methods used in the paper are cited in the main text or appendix.

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: The paper does not introduce or release a new dataset, benchmark, or model asset as part of the submission. It proposes training procedures and reports experimental results.

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: The research does not involve crowdsourcing, user studies, or experiments 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: The research does not involve human subjects, so IRB approval or equivalent review is not applicable.

    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 does not use LLMs as an important, original, or non-standard component.

    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.