ASVQ: What Reparameterization Is Just Enough for Efficient Codebook Learning?

Wenfeng Zou
Independent Researcher
wenfeng1345@outlook.com

Abstract

Addressing codebook collapse in vector quantization models is crucial for discrete representation learning. Recent solutions increasingly rely on expressive codebook reparameterizations, improving adaptability at the cost of additional optimization and computational overhead. This motivates a simple question: in empirical terms, what reparameterization scheme is just enough for efficient codebook learning? In this work, we propose Adaptive-Scale Vector Quantization (ASVQ), a lightweight method for learning the codebook with zero trainable reparameterization parameters in a decoupled way. Experiments across image and audio tokenization tasks demonstrate that ASVQ consistently improves codebook utilization, reconstruction quality, and training stability, while remaining competitive with more complex reparameterization-based methods. These results suggest that ASVQ provides a favorable trade-off between codebook expressiveness and efficiency. Code: https://github.com/elixirmkr/ASVQ

1 Introduction

Recently, vector quantization (VQ) has become pivotal to contemporary unsupervised representation learning and generative modeling [29, 8]. By discretizing continuous distributions in the latent space, VQ models can effectively capture the intrinsic manifold structure of data, thereby enabling discrete probability density estimation over continuous-domain signals. As a result, their applications now span a wide range of scenarios, from high-fidelity image generation [24, 7, 26] to speech synthesis [31, 34].

On the other hand, with the rapid rise of large-scale multimodal foundation models, VQ has evolved beyond a mere compression mechanism into a crucial bridge that connects continuous modalities such as vision and speech with the discrete token space of autoregressive language models [33, 7]. Recent multimodal tokenizers are designed directly for unified understanding and generation frameworks [22, 6], employing single or multiple VQ codebooks to map images, videos, and other modalities into discrete symbols that can be shared with or aligned to textual tokens. This trend has further driven the expansion of codebook sizes to tens of thousands of entries or even more [38, 3].

However, when trained with large-scale codebooks, VQ models often suffer from the notorious issue of *codebook collapse* [27, 28], which exhibits unstable training dynamics and manifests as extremely low codebook utilization, i.e., the learned discrete representation uses only a fraction of the full capacity of the codebook.

Current studies typically resort to *reparameterizing* the codebook to alleviate this problem. For instance, some recent studies transform codebooks by introducing various projection modules, aiming to optimize their spatial coverage [16, 32, 38, 21, 3, 39]. Although these methods have achieved some success, they often come at the cost of increased model complexity, substantial computational and optimization overhead, or the need for intricate multi-stage or bilevel training strategies. This raises a

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

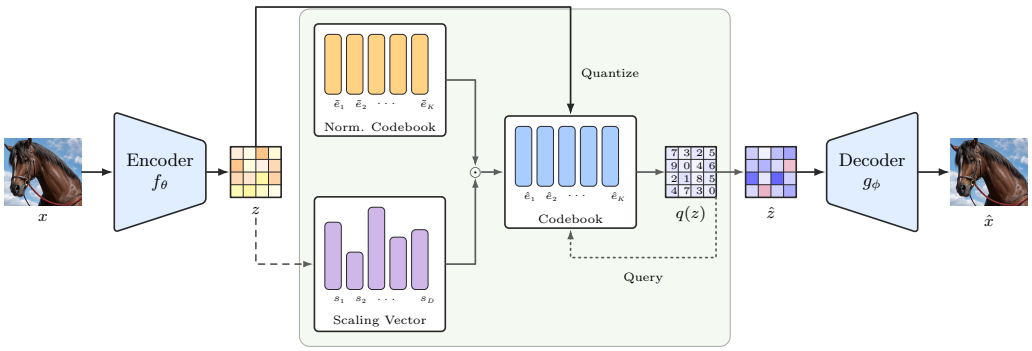


Figure 1: **Overview of Adaptive-Scale Vector Quantization (ASVQ).**

natural question of whether rotations, dense channel mixing, or complex nonlinear transformations are truly necessary in practice for efficient codebook learning.

In this work, we show that a simple channel-wise scaling of normalized codewords is surprisingly efficient. The main contributions of this paper can be summarized as follows:

- We analyze the scale component in nearest-neighbor assignment and codebook updates, and show that scale adaptation is a key factor in effective codebook learning across different methods.
- Motivated by these findings, we propose ASVQ, a diagonal adaptive-scale reparameterization approach in a decoupled way for codebook learning with zero additional trainable parameters.
- We conduct both quantitative and qualitative experiments across multiple model architectures and datasets. The results show that channel-wise scaling captures much of the empirical benefit of more complex reparameterizations across codebook utilization and model performance under diverse settings.

2 Related Work

Vector Quantized Variational Autoencoder (VQ-VAE), first proposed by van den Oord et al. [29], introduces a discrete codebook at the bottleneck of an autoencoder and maps continuous latent variables to discrete indices via hard nearest-neighbor assignment, successfully learning structured discrete representations. The codebook can be updated either through gradient descent [2] or via an exponential moving average (EMA) scheme [29, 24]. In high-resolution visual generation settings, VQGAN [7] builds upon the reconstruction objective of VQ-VAE by incorporating perceptual [36] and adversarial training [9], thereby learning discrete visual tokens that better align with human perception and significantly improving both reconstruction and generation quality. While traditional VQ-based methods have driven notable gains in reconstruction performance, they often suffer from low codebook utilization [27, 12]. Although multi-channel quantization techniques [16] partially enhance fine-grained detail recovery, such approaches are highly prone to representation collapse and the emergence of a large number of "dead codes" when scaling to large codebooks. This scale-induced performance bottleneck remains a central open challenge in the field.

To address this issue, a variety of strategies have been explored. Zheng and Vedaldi [37], Zeghidour et al. [34], and Dhariwal et al. [5] propose periodically detecting and resetting dead codes that remain unused over extended training periods. For instance, these codes can be replaced with random vectors sampled from the active codebook or with encoder output vectors from the current mini-batch. While such methods can directly improve codebook utilization metrics, they primarily treat the symptoms of codebook collapse rather than addressing the underlying causes that lead codes to become inactive.

▷ **Codebook Reparameterization Methods**
To fundamentally mitigate the dead-code problem and improve codebook scalability, a number of reparameterization strategies have been proposed in recent years.

- **Affine Reparameterization (AffineVQ):** Huh et al. [10] identify the distributional mismatch between encoder embeddings and codebook vectors as an important source of training instability, and propose to reduce this mismatch through a learnable affine transformation.

- **VQGAN-LC:** Zhu et al. [38] initializes an extremely large static codebook using a large collection of features extracted from a pretrained CLIP encoder, and trains a linear projector to align the outputs of the VQ-GAN encoder with the codebook space. This approach successfully scales the codebook on the order of hundreds of thousands while maintaining very high utilization.
- **Meta Quantization (MQ):** To enhance the expressive capacity of the codebook, Qin and Zhang [21] introduce hypernetworks and meta-learning frameworks. Hypernetworks dynamically generate codebook parameters conditioned on the input; however, this formulation introduces a complex bi-level optimization problem, substantially increasing training complexity.
- **SimVQ:** Zhu et al. [39] demonstrate that even applying a simple learnable linear transformation, such as a fully connected layer, to a base codebook can effectively alleviate representation collapse.
- **FVQ:** Building upon SimVQ [39], Chang et al. [3] further propose using more expressive Transformer-based nonlinear projection modules to generate or transform the codebook. This design dramatically improves codebook scalability and is reported to achieve 100% codebook utilization.

The aforementioned methods span a wide range of reparameterization choices. While these methods improve codebook utilization, they introduce different degrees of freedom, including translation, channel-wise scaling, rotation, dense channel mixing, and nonlinear transformations. As a result, it remains unclear which components are essential for the empirical gains. ASVQ is motivated by this question. Rather than increasing the expressiveness of the codebook transformation, ASVQ deliberately restricts it to a normalized base codebook followed by a diagonal channel-wise scale. This design can be viewed as a minimal member of the broader reparameterization family and a just-enough mechanism for stable large-codebook learning.

3 Preliminaries

In this section, we first review the basic form of the Vector Quantized Variational Autoencoder (VQ-VAE) [29] and the Straight-Through Estimator (STE), and introduce the notation required for the subsequent analysis.

3.1 VQ Model and Loss Function

Let $X \sim p_{\text{data}}$ denote a random input and $x \in \mathcal{X}$ one of its realizations. For a realized input, the encoder f_θ produces a continuous latent vector $z = f_\theta(x) \in \mathbb{R}^D$. We write $Z = f_\theta(X)$ when treating the encoder output as a random vector. The vector quantization layer maintains a codebook containing K codewords $\mathcal{E} = \{e_k \in \mathbb{R}^D\}_{k=1}^K$. The quantization operation maps z to its nearest-neighbor codeword index:

$$q(z) \coloneqq \operatorname*{argmin}_{1 \le k \le K} \|z - e_k\|_2^2 \tag{1}$$

yielding the quantized latent representation $z_q \coloneqq e_{q(z)}$. The decoder g_ϕ reconstructs the input using this quantized latent vector: $\hat{x} = g_\phi(z_q)$. For the distributional training objective, let $Z_q \coloneqq e_{q(Z)}$. The standard VQ-VAE objective consists of three components: reconstruction loss, codebook loss, and commitment loss:

$$\begin{aligned}\mathcal{L}(\theta, \phi, \mathcal{E}) = &\underbrace{\mathbb{E}_{X \sim p_{\text{data}}}\big[\ell\big(X, g_\phi(Z_q)\big)\big]}_{\mathcal{L}_{\text{rec}}} + \underbrace{\mathbb{E}_{X \sim p_{\text{data}}}\big[\|\operatorname{sg}[Z] - e_{q(Z)}\|_2^2\big]}_{\mathcal{L}_{\text{codebook}}} \\ &+ \beta \underbrace{\mathbb{E}_{X \sim p_{\text{data}}}\big[\|Z - \operatorname{sg}[e_{q(Z)}]\|_2^2\big]}_{\mathcal{L}_{\text{commit}}}\end{aligned} \tag{2}$$

where $\operatorname{sg}[\cdot]$ denotes the stop-gradient operator and $\beta > 0$ is a balancing coefficient. The term $\mathcal{L}_{\text{codebook}}$ updates only the codebook parameters $\mathcal{E}$, while $\mathcal{L}_{\text{commit}}$ updates only the encoder parameters θ; together, they constrain the encoder outputs to align with the codebook distribution. Many implementations also employ exponential moving averages (EMA) [24] or online k-means-style updates [19].

Require: Encoder f_θ, decoder g_ϕ,
codebook $\mathcal{E} = \{e_k\}_{k=1}^K$, embedding dimension D,
commitment β, EMA factor γ
repeat
  Sample a minibatch matrix $\mathbf{X}$ with rows $x_n \sim p_{\text{data}}$, $n = 1, \ldots, N$
  Encode: $\mathbf{Z} \leftarrow f_\theta(\mathbf{X})$, with rows z_n
  Normalize: $\tilde{e}_k \leftarrow \dfrac{e_k}{\max(\sigma(\mathcal{E}),\, 10^{-8})}$
  EMA scale: $s \leftarrow \gamma s + (1-\gamma)\,\widehat{\sigma}(\mathbf{Z})$
  Effective codebook: $\hat{e}_k \leftarrow s \odot \tilde{e}_k$
  Quantize each row: $q(z_n) \leftarrow \arg\min_k \|z_n - \hat{e}_k\|_2^2$
  Straight-through: $z_{q,n} \leftarrow z_n + \mathrm{sg}[\hat{e}_{q(z_n)} - z_n]$
  Decode: $\hat{x}_n \leftarrow g_\phi(z_{q,n})$
  $\mathcal{L} = \dfrac{1}{N}\sum_{n=1}^N \big[\mathrm{MSE}(x_n, \hat{x}_n) + \|\,\mathrm{sg}[z_n] - \hat{e}_{q(z_n)}\|_2^2 + \beta\|z_n - \mathrm{sg}[\hat{e}_{q(z_n)}]\|_2^2\big]$
  Update $\theta, \phi, \mathcal{E}$ via $\nabla\mathcal{L}$
until converged

Algorithm 1: Training procedure of ASVQ.

3.2 Straight-Through Estimator (STE)

Since the nearest-neighbor selection in $q(z)$ is a non-differentiable $\arg\min$ operation, VQ models typically employ the straight-through estimator [2] to approximate gradients through the quantization layer. A common strategy is:

$$z_q = z + \mathrm{sg}[e_{q(z)} - z] \tag{3}$$

Here, $e_{q(z)}$ is used in the forward pass, while in the backward pass:

$$\frac{\partial \mathcal{L}}{\partial z} \approx \frac{\partial \mathcal{L}}{\partial z_q} \tag{4}$$

In other words, gradients bypass the quantization decision itself during backpropagation, copying the gradient of z_q to z without accounting for the effect of changing the nearest-neighbor index $q(z)$ on the loss.

4 Adaptive-Scale Vector Quantization

In this section, we first provide the theoretical intuition about the scale component, next present the empirical evidence highlighting the importance of channel-scale adaptation in codebook learning across different reparameterization schemes, and formalize the definition of ASVQ. Finally, we discuss the mechanism differences between ASVQ and AffineVQ.

4.1 Scale Component in Codeword Assignment and Update

A central question behind reparameterization-based VQ methods is which part of the added expressiveness is essential for improving codebook learning. We first examine how the nearest-neighbor assignment rule responds to feature statistics and codeword geometry. To see this, consider only two distinct codewords e_i and e_j. A sample z is assigned to codeword i over j if and only if the squared-distance difference $\Delta_{ij}(z) < 0$:

$$\begin{aligned} \Delta_{ij}(z) &:= \|z - e_i\|_2^2 - \|z - e_j\|_2^2 \\ &= 2z^\top(e_j - e_i) + \|e_i\|_2^2 - \|e_j\|_2^2 \end{aligned} \tag{5}$$

To study the distribution of this assignment rule, consider the encoder output $Z = f_\theta(X)$. Denote the codeword difference by $\delta_{ij} = e_j - e_i$ and the feature covariance by $\Sigma = \mathrm{Cov}(Z)$. Since the last two terms in Eq. (5) are constants with respect to Z, the variance of the assignment variable is

$$\mathrm{Var}(\Delta_{ij}(Z)) = 4\delta_{ij}^\top \Sigma \delta_{ij} \tag{6}$$

This expression shows that assignment sensitivity depends jointly on the covariance structure of latent features and the alignment of codeword differences with high-variance directions. If $\Sigma = U\Lambda U^\top$, then

$$\mathrm{Var}(\Delta_{ij}(Z)) = 4\sum_{r=1}^{D} \lambda_r (u_r^\top \delta_{ij})^2 \tag{7}$$

where λ_r and u_r are the eigenvalues and eigenvectors of Σ. Hence, a high-variance direction contributes strongly to the assignment when the codeword difference has a non-negligible projection onto that direction. Conversely, if δ_{ij} is orthogonal to a high-variance direction, that direction does not affect the assignment between this pair of codewords.

The same scale-sensitive effect also appears in codebook updates. Under a fixed-assignment view, the per-sample gradient of the standard codebook loss with respect to an assigned codeword is proportional to the residual between the codeword and the assigned feature. As shown in Appendix A.1, for channel d,

$$\mathrm{Var}(g_{k,d}(Z) \mid q(Z) = k) = 4\,\mathrm{Var}(Z_d \mid q(Z) = k) \tag{8}$$

Thus, after assignments are made, the variability of codeword updates is also controlled by the conditional feature scale within each Voronoi region [30]. Together with assignment frequency, this affects the effective update signal received by each codeword. This analysis is local and does not claim to fully characterize the discontinuous dynamics of VQ training, but it identifies a concrete scale-dependent component in both assignment and update behavior.

4.2 Why Scale Adaptation May Be Just-Enough

The previous subsection isolates a common factor behind both assignment and update behavior: channel-wise feature scale changes the effective weighting of distances before assignment and the amplitude of updates after assignment. This suggests that scale is not merely a descriptive statistic of the latent space, but a quantity that the codebook need to implicitly track to cover the data manifold effectively during training. When recent reparameterized VQ methods improve training, a natural follow-up question is how much of the resulting codebook motion is actually spent on adapting feature-scale drift.

To answer this, we analyze the codebook trajectory of different methods across checkpoints. For a learned codebook, we define its global channel-wise scale by the standard deviation across codewords and its normalized geometry by dividing each channel by that scale. We then decompose the change between consecutive checkpoints into a scale term and a normalized-geometry term using a midpoint decomposition, and accumulate the squared Frobenius norm of the two components over training. Figure 2 reports the resulting cumulative update-energy shares; the step-by-step calculation that produces them is given in Appendix A.2.

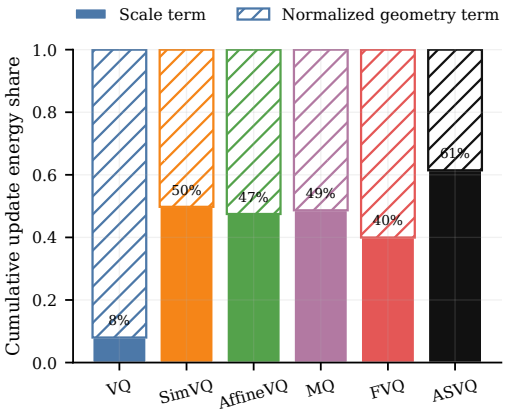


Figure 2: **Scale versus normalized-geometry contributions to codebook motion.** For each method, we decompose the codebook change between consecutive checkpoints into a channel-scale term and a normalized-geometry term, then accumulate their squared norms over training.

Surprisingly, two patterns stand out. When reparameterization is introduced, SimVQ, AffineVQ, MQ, and FVQ all allocate roughly 40%–50% of cumulative codebook motion to scale changes, whereas vanilla VQ shows only 8%. This indicates that existing reparameterized methods already spend a decent fraction of their effective degrees of freedom on tracking feature-scale drift, even when they are parameterized with much more expressive transformations. Therefore, it is possible that explicitly separating scale from geometry captures an important component of the optimization dynamics and absorbs the channel scale-tracking behavior. This directly motivates ASVQ as a just-enough reparameterization method for efficient codebook learning.

4.3 ASVQ Formulation

We hereby formally introduce Adaptive-Scale Vector Quantization (ASVQ). ASVQ preserves nearest-neighbor quantization and the standard VQ training objective, while explicitly decoupling between the internal channel scale of the codebook and the channel scale of the data.

Codebook Normalization We introduce channel-wise scale normalization for the base codewords. Given the codebook parameters $\mathcal{E} = \{e_k\}_{k=1}^K$, we define the mean and standard deviation of each channel across the codebook as:

$$\mu_d = \frac{1}{K}\sum_{j=1}^{K} e_{j,d}, \qquad \sigma(\mathcal{E})_d = \sqrt{\frac{1}{K}\sum_{j=1}^{K}(e_{j,d} - \mu_d)^2} \tag{9}$$

Here, the channel-wise standard deviation is used only as a dispersion estimator for rescaling, so we normalize codewords without translation:

$$\tilde{e}_k = \frac{e_k}{\sigma(\mathcal{E})} \tag{10}$$

Global Channel Scaling Factor Moreover, we introduce a global scaling factor s to track the channel scale of the latent features during training. The factor is estimated directly from encoder feature statistics via exponential moving average (EMA):

$$s^{(t)} \leftarrow \gamma s^{(t-1)} + (1-\gamma)\widehat{\sigma}(\mathbf{Z}^{(t)}) \tag{11}$$

where $\widehat{\sigma}(\mathbf{Z}^{(t)})$ denotes the empirical channel-wise standard deviation vector computed across the mini-batch from the encoder output matrix $\mathbf{Z}^{(t)}$ at optimization step t. Feature-wise operations are performed row-wise on this matrix, and $\gamma \in (0, 1)$ is the smoothing coefficient.

Therefore, the effective codebook of ASVQ, $\hat{\mathcal{E}} = \{\hat{e}_k\}_{k=1}^K$, is defined as:

$$\hat{e}_k = s \odot \tilde{e}_k \tag{12}$$

where $\odot$ denotes element-wise multiplication. The normalized base codebook is rescaled by the assignment-independent adaptive scale, ensuring that the channel scale of the effective codewords match the feature scale. The quantization assignment uses $\hat{e}_k$:

$$q(z) = \arg\min_k \|z - \hat{e}_k\|_2^2, \qquad z_q = \hat{e}_{q(z)} \tag{13}$$

Figure 1 illustrates the overall framework of ASVQ, and the detailed training procedure of ASVQ is provided in Algorithm 1.

4.4 Discussion of the Mechanism of ASVQ to AffineVQ

The difference between the two methods lies in whether the base codebook is constrained to have channel-wise unit normalization, and whether the scale update is independent of the current assignment.

ASVQ's effective codebook has channel-wise scale:

$$\sigma(\tilde{\mathcal{E}}) = \mathbf{1}, \qquad \sigma(\hat{\mathcal{E}}) = s \tag{14}$$

where s is estimated directly from encoder feature statistics. As a result, $\sigma(\hat{\mathcal{E}})$ tracks the batch feature statistics $\widehat{\sigma}(\mathbf{Z}^{(t)})$ through the EMA update, and the base codebook no longer needs to implicitly learn this scale through assignment-dependent updates. This also explains the two separate ablations in Table 4. With normalization alone, $\sigma(\hat{\mathcal{E}}) = \mathbf{1}$, which cannot track the feature scale as it evolves during training. With scaling alone, $\sigma(\hat{\mathcal{E}}) = s \odot \sigma(\mathcal{E})$, and the base codebook still retains a set of freely varying channel scales, so the scale is jointly represented by both s and $\mathcal{E}$, with no guarantee that the effective codebook scale matches the data scale. When the two are combined, normalization removes this redundant degree of freedom, and s becomes the sole, data-statistics-driven global scale. This is the key difference between ASVQ and AffineVQ. AffineVQ has the form

$$\hat{e}_k = e_{\text{mean}} + e_{\text{std}} \odot e_k \tag{15}$$

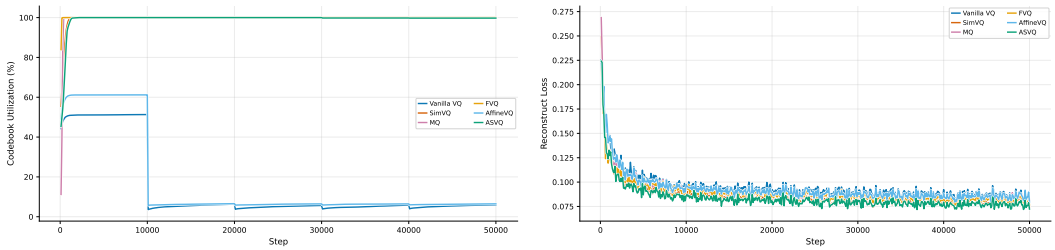


Figure 3: **Training dynamics on ImageNet-1k.** Codebook utilization (left) and reconstruction loss (right) over 50k training steps, under the setting of Table 1.

whose effective codebook channel standard deviation is $\sigma(\hat{\mathcal{E}}) = |e_{\mathrm{std}}| \odot \sigma(\mathcal{E})$. Since e_k is not constrained to unit channel-wise standard deviation, the scale information still lies in e_{std} and the original codebook at the same time. Moreover, these quantities are jointly optimized through the assignment-dependent VQ loss, whereas the estimation of s in ASVQ is decoupled from gradient descent, independent of which codewords are currently active.

5 Experiments

In this section, we evaluate the performance of ASVQ on image tokenization tasks, encompassing fundamental evaluations based on VQ-VAE [29] and high-resolution image synthesis experiments based on VQGAN [7]. Subsequently, we extend ASVQ to the audio modality, verifying its effectiveness and generalization capabilities as a universal tokenizer on the LibriTTS-clean dataset [35]. Finally, we provide more diverse visualization analyses, qualitative evaluations, and ablation studies.

5.1 VQ-VAE-Based Evaluation

Setup On ImageNet-1k [4], we compare VanillaVQ [29], AffineVQ [10], SimVQ [39], MQ [21], FVQ [3], and ASVQ in a controlled VQ-VAE image tokenization setting. All methods share the same training configuration and differ only in the codebook parameterization module. Following the convolutional encoder-decoder design introduced in VQGAN, we use a latent dimension of 32 and a codebook with 16,384 entries, and train on 128×128 images. To isolate the effect of codebook reparameterization, adversarial and perceptual terms [9, 36] are disabled, so the comparison focuses purely on reconstruction quality and codebook behavior. The exact definitions of codebook perplexity (PPL) and utilization (Util.) are presented in Appendix A.4, and full hyperparameters are summarized in Appendix Table 5.

Notably, the original implementation by Huh et al. [10] augments AffineVQ with alternated optimization and a synchronized commitment loss, alongside various training heuristics such as LRU replacement, ℓ_2 normalization, K-means initialization, and a warmup schedule. To ensure our evaluation reflects only differences within the reparameterization family itself, we consistently disabled auxiliary techniques across all methods. Consequently, our implementation evaluates only the core reparameterization modules rather than the fully engineered pipelines.

Table 1: Comparison of image tokenization performance on the ImageNet-1k validation dataset. *Extra params* denote trainable reparameterization parameters. SSIM scores are computed using the PIQ library [14] and scaled by 100.

Method	Transform type	Extra params	PSNR ↑	SSIM ↑	PPL ↑	Util. (%) ↑
VanillaVQ	none	0	24.09	68.81	561	4.3
SimVQ	dense linear	1,056	25.02	72.68	12,421	**100.0**
AffineVQ	affine	64	24.27	69.62	822	6.0
MQ	nonlinear	5,280	24.56	69.53	11,991	**100.0**
FVQ	nonlinear	886,016	24.60	69.55	13,548	**100.0**
ASVQ	diagonal scale	0	**25.17**	**73.85**	**15,144**	99.7

Main Results Table 1 shows remarkable results. AffineVQ collapses unexpectedly in this evaluation, reaching only 6.0% utilization and low perplexity. However, ASVQ delivers the strongest overall trade-off: it achieves the best PSNR and SSIM, the highest perplexity, and raises utilization to 99.7%. We hypothesize the performance discrepancy between AffineVQ and ASVQ stems from scale identifiability and the source of the update signal, as discussed in Section 4.4. Figure 3 illustrates the training dynamics across reparameterization variants: ASVQ rapidly achieves and sustains near-full utilization, matching more complex methods, and maintains the lowest reconstruction loss throughout. Significantly, these gains require no trainable reparameterization parameters, unlike dense or nonlinear alternatives that demand substantially more capacity. This observation suggests that explicit channel-wise scale adaptation is already sufficient to recover nearly full codebook usage and superior reconstruction quality in this large-scale setting.

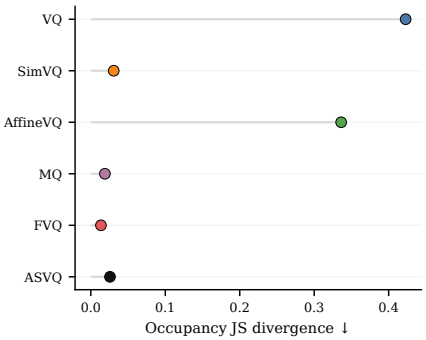


Figure 4: **Occupancy-based manifold matching.**

Manifold Coverage Results A natural concern is whether ASVQ can still cover the latent data manifold as effectively as more flexible reparameterized methods. To examine this point, we diagnose the learned codebook from the perspective of occupancy matching, quantified by occupancy JS divergence (defined in Appendix A.4). Lower occupancy JS divergence indicates better agreement between the learned codebook occupancy and the latent data manifold occupancy. Figure 4 shows that ASVQ falls in the same low-divergence regime as SimVQ, MQ, and FVQ and compares favorably with VanillaVQ and AffineVQ, despite using a much more restricted transformation family. Empirically, this result alleviates the concern that diagonal scale adaptation may be too weak to support effective manifold coverage. We also employ t-SNE [18] to visualize the dimensionality-reduced embeddings. As shown in Figure 5, Vanilla VQ exhibits severe "cluster shrinkage," where a vast number of codewords collapse into narrow regions, rendering them ineffective. In contrast, ASVQ displays a uniform codebook distribution in the latent space, broadly covering the data manifold.

5.2 VQGAN-Based Evaluation

Setup To complement the controlled VQ-VAE comparison above, we further evaluate ASVQ in a VQGAN-based tokenizer [7] on FFHQ [13]. For a fair comparison, we follow the reported FFHQ setting used in the cited VQGAN-LC [38] and MQ [21] results, including 256×256 image inputs and the same $f = 16$ spatial downsampling ratio. The encoder-decoder backbone is inherited from the original VQGAN architecture, with latent dimension 32 and a codebook size of 16,384. Unlike the VQ-VAE evaluation, this setting keeps the standard VQGAN perceptual and adversarial objectives enabled. The complete configuration is summarized in Appendix Table 5.

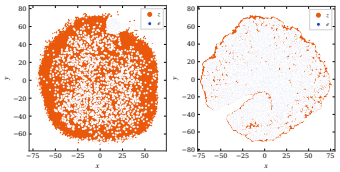


Figure 5: **Codebook distribution visualized with t-SNE.** Left: vanilla VQ. Right: ASVQ.

Results Table 2 reports the evaluation results based on perceptual quality metrics (rFID [28], LPIPS [36]) and codebook utilization. With the same 16,384-code budget as the standard VQGAN baselines, ASVQ raises utilization to 97.8% and achieves the best perceptual quality. Compared with MQGAN, ASVQ trades a small drop in utilization for clearly better perceptual fidelity; compared with VQGAN-LC, it obtains stronger reconstruction performance without relying on a 100k codebook. The overall message remains consistent with the ImageNet-1k study.

Table 2: Comparison of image tokenization performance on the FFHQ validation dataset. † and ‡ denote results reported in VQGAN-LC and MQ, respectively.

Method	Codebook Size	Util. (%) ↑	rFID ↓	LPIPS ↓
VQGAN†	16,384	2.3	5.25	0.12
VQGAN-FC† [32]	16,384	10.9	4.86	0.11
VQGAN-EMA†	16,384	68.2	4.79	0.10
VQGAN-LC†	100,000	99.5	3.81	0.08
MQGAN‡	16,384	**100**	4.25	0.08
ASVQ	16,384	97.8	**2.63**	**0.07**

Table 3: Comparison of audio reconstruction performance on the LibriTTS-clean dataset. Bandwidths of 0.9 kbps and 1.2 kbps correspond to codebook sizes of 4,096 and 65,536, respectively.

Method	Bandwidth	Util. (%) ↑	UTMOSv2 ↑	PESQ ↑	STOI ↑	V/UV F1 ↑
Ground Truth	-	-	3.11	-	-	-
VanillaVQ	0.9 kbps	4.1	2.81	1.61	0.84	0.91
VanillaVQ	1.2 kbps	0.1	2.29	1.41	0.81	0.89
SimVQ	0.9 kbps	**99.8**	2.89	1.90	0.88	0.93
SimVQ	1.2 kbps	91.2	2.91	2.01	0.89	**0.94**
ASVQ	0.9 kbps	94.5	2.85	1.89	0.88	0.93
ASVQ	1.2 kbps	83.9	**2.91**	**2.05**	**0.90**	0.93

5.3 Audio Tokenizer Evaluation

Setup To verify the generalization capability of ASVQ in non-image modalities, we train an audio discrete codec on the LibriTTS-clean dataset [35]. The architecture follows WavTokenizer [11]. We compare against Vanilla VQ [29] as the canonical non-reparameterized baseline and SimVQ [39] as a representative reparameterization baseline, since SimVQ has been previously evaluated in audio settings, whereas most other reparameterization methods are introduced primarily for image tokenization and do not provide standardized audio implementations or training recipes for a faithful comparison. Models are trained for 50 epochs with a batch size of 64 and a learning rate of 1×10^{-4}. The audio window size is set to 1 s (corresponding to 75 tokens). We conduct tests under both low-bitrate (0.9 kbps/4,096 codes) and high-bitrate (1.2 kbps/65,536 codes) settings.

Results Table 3 presents objective and subjective audio reconstruction metrics, including UTMOSv2 [1], PESQ [25], STOI, and V/UV F1. Overall, ASVQ keeps competitive on audio evaluation. At 0.9 kbps, SimVQ has slightly higher utilization and some quality metrics; at 1.2 kbps, ASVQ and SimVQ achieve the same UTMOSv2, with ASVQ marginally outperforming on PESQ/STOI, but slightly trailing on utilization and V/UV F1. Notably, under the large-codebook setting, ASVQ preserves high codebook utilization while delivering superior reconstruction quality, further highlighting its efficient representational capacity across different modalities.

5.4 Visualization

Here we perform a multi-perspective visualization analysis of the trained models. For readability, all figures referenced in this subsection are collected in Appendix A.5.

Qualitative Evaluation We first compare all reparameterization methods on the same ImageNet-1k inputs in Figure 6, where ASVQ recovers thin structures, fine textures and sharp edges, while VanillaVQ and AffineVQ smear them into smooth patches. In Figure 7, we present high-resolution sample reconstructions on the FFHQ test set. Benefiting from ASVQ's precise capture of high-frequency details, the reconstructed images achieve an extremely high degree of fidelity in hair strands, skin textures, and facial features. In Figure 10, we compare the original and reconstructed waveforms of LibriTTS-clean samples across different codebook sizes.

Channel Variance Alignment We plot the comparison between the channel standard deviation of the learned codebook and that of the encoder output features. As illustrated in Figure 8, the codebook and data in the original VQ-VAE [29] suffer from a severe mismatch at the channel scale. Conversely, the empirical $\sigma(\mathcal{E})$ and $\widehat{\sigma}(\mathbf{Z})$ curves for ASVQ almost perfectly overlap, where $\mathbf{Z}$ collects the encoder outputs used for visualization. This high degree of alignment demonstrates that ASVQ, through its decoupled learning strategy, successfully captures the anisotropy [23, 20] of the data.

Training Stability Figure 9 displays the audio reconstruction loss curves of different methods during training. The results indicate that both SimVQ [39] and Vanilla VQ [29] experience drastic loss spikes in the early stages of training, revealing optimization instability. In contrast, ASVQ maintains a smooth descending trend throughout and ultimately converges to the lowest reconstruction loss, showing its robustness and efficiency during the optimization process.

Table 4: Ablation studies on ImageNet-1k.

Factor	Setting	PSNR ↑	SSIM ↑	PPL ↑	Util. (%) ↑
Default ($K = 16{,}384$, ✓ / ✓, $\gamma = 0.99$)		25.17	73.85	15,144	99.7
(a) Codebook size	8,192	24.98	72.75	7,577	100.0
	32,768	25.34	74.93	29,982	99.9
	65,536	**25.55**	**75.66**	**59,624**	**100.0**
(b) Normalization / scaling	× / ×	24.09	68.81	561	4.3
	✓ / ×	24.20	69.36	656	5.4
	× / ✓	24.33	69.49	1,018	9.6
(c) EMA coefficient γ	0.9	25.14	73.78	13,681	99.5
	0.999	25.09	72.93	12,326	97.8
	0.9999	24.13	68.89	663	5.6

5.5 Ablation Studies

Setup Unless otherwise stated, all studies on ImageNet-1k follow the same VQ-VAE setting in Appendix Table 5. We study three factors: (1) codebook size, (2) the effect of normalization and channel-wise scaling, and (3) the EMA coefficient γ used to update s.

Results Table 4(a) shows that ASVQ scales smoothly with larger codebooks: reconstruction quality metrics improve monotonically as the codebook grows, while utilization remains near saturation throughout. Table 4(b) further shows that normalization and scale adaptation are complementary. Using either component alone brings only limited gains over Vanilla VQ, whereas combining both yields a marked improvement. These results strongly support the necessity of our decoupled design. Table 4(c) completes the picture by varying the EMA coefficient γ in Eq. (11). Performance is essentially flat for $\gamma \in [0.9, 0.999]$, and breaks down only at $\gamma = 0.9999$. This is consistent with the role assigned to s: an overly large γ makes s respond too slowly to the rapidly changing batch statistics $\widehat{\sigma}(\mathbf{Z}^{(t)})$ in early training, and can lead to collapse. The default $\gamma = 0.99$ therefore sits comfortably inside a wide stable range.

6 Conclusion and Discussion

We revisit codebook reparameterization from the perspective of identifying the minimal transformation needed for stable and efficient large-codebook learning. Our analysis suggests that channel-wise scale adaptation captures an important part of transformed codebook motion, while geometry still governs relative codeword layout. Motivated by this observation, we introduce ASVQ, which combines codeword normalization with lightweight adaptive channel-wise scaling. Across image and audio tokenization benchmarks, ASVQ consistently improves VQ training while retaining negligible overhead. Overall, the results imply that, in our evaluated settings, a diagonal reparameterization provides a competitive approximation to the practical gains commonly associated with more complex codebook transformations.

Limitations The study is limited in scope to tokenizer-level effectiveness, leaving a comprehensive evaluation on downstream tasks as an important direction for future work. Our explanation of why ASVQ works is mainly heuristic and supported by empirical analysis, rather than a rigorous and comprehensive theory of VQ codebook learning. Establishing a more complete theoretical account of the training dynamics, and clarifying when simple diagonal reparameterization suffices versus when richer transformations are necessary, remain important future work. Our current hypothesis is that when the latent distribution has strong off-diagonal covariance, cross-channel correlation or nonlinear manifold structure, a single global diagonal scaling may be insufficient, necessitating a dense or nonlinear transformation. In terms of audio experiments, the advantage of SimVQ on some metrics also indicates that richer transformations may be valuable in certain settings.

Acknowledgments and Disclosure of Funding

The author is grateful to the anonymous reviewers for their insightful and constructive comments. The author also thanks the AI research community for its commitment to openness and for generously sharing code, data, and models. This research was conducted independently without institutional or funding support, utilizing cloud GPU resources from Modal at the author's own expense. Beyond academic support, the author deeply thanks his family, especially his mother, whose love, strength, and blessing sustained him through the losses and challenges of completing this research. Special appreciation is also extended to Jun Maeda's compositions, particularly *Last Regrets* and *Light Colors*, for conveying profound comfort and inspiration.

References

[1] K. Baba, W. Nakata, Y. Saito, and H. Saruwatari. The t05 system for the VoiceMOS challenge 2024: Transfer learning from deep image classifier to naturalness MOS prediction of high-quality synthetic speech. In *IEEE Spoken Language Technology Workshop (SLT)*, 2024.

[2] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation, 2013. URL https://arxiv.org/abs/1308.3432.

[3] Yifan Chang, Jie Qin, Limeng Qiao, Xiaofeng Wang, Zheng Zhu, Lin Ma, and Xinggang Wang. Scalable training for vector-quantized networks with 100% codebook utilization, 2025. URL https://arxiv.org/abs/2509.10140.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, K. Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[5] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music, 2020. URL https://arxiv.org/abs/2005.00341.

[6] 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 30322–30334, 2026.

[7] 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.

[8] Allen Gersho and Robert M Gray. *Vector quantization and signal compression*, volume 159. Springer Science & Business Media, 2012.

[9] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[10] 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.

[11] Shengpeng Ji, Ziyue Jiang, Wen Wang, Yifu Chen, Minghui Fang, Jialong Zuo, Qian Yang, Xize Cheng, Zehan Wang, Ruiqi Li, Ziang Zhang, Xiaoda Yang, Rongjie Huang, Yidi Jiang, Qian Chen, Siqi Zheng, and Zhou Zhao. Wavtokenizer: an efficient acoustic discrete codec tokenizer for audio language modeling, 2025. URL https://arxiv.org/abs/2408.16532.

[12] Lukasz Kaiser, Samy Bengio, Aurko Roy, Ashish Vaswani, Niki Parmar, Jakob Uszkoreit, and Noam Shazeer. Fast decoding in sequence models using discrete latent variables. In *International Conference on Machine Learning*, pages 2390–2399. PMLR, 2018.

[13] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4401–4410, 2019. doi: 10.1109/CVPR.2019.00453.

[14] Sergey Kastryulin, Jamil Zakirov, Denis Prokopenko, and Dmitry V Dylov. Pytorch image quality: Metrics for image quality assessment. *arXiv preprint arXiv:2208.14818*, 2022. URL https://arxiv.org/abs/2208.14818.

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. URL http://arxiv.org/abs/1412.6980. Poster.

[16] 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 (CVPR)*, pages 11513–11522, 2022.

[17] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

[18] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[19] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

[20] Anna Marbut, Katy McKinney-Bock, and Travis Wheeler. Reliable measures of spread in high dimensional latent spaces. In *International Conference on Machine Learning*, pages 23871–23885. PMLR, 2023.

[21] Peijia Qin and Jianguo Zhang. Learning to quantize for training vector-quantized networks. In *International Conference on Machine Learning*, 2025.

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

[23] Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. Hubs in space: Popular nearest neighbors in high-dimensional data. *Journal of Machine Learning Research*, 11:2487–2531, 2010.

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

[25] A. W. Rix, J. G. Beerends, M. P. Hollier, and A. P. Hekstra. Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 749–752 vol.2, 2001.

[26] 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.

[27] Aurko Roy, Ashish Vaswani, Arvind Neelakantan, and Niki Parmar. Theory and experiments on vector quantized autoencoders. *arXiv preprint arXiv:1805.11063*, 2018.

[28] Yuhta Takida, Takashi Shibuya, WeiHsiang Liao, Chieh-Hsin Lai, Junki Ohmura, Toshimitsu Uesaka, Naoki Murata, Shusuke Takahashi, Toshiyuki Kumakura, and Yuki Mitsufuji. SQ-VAE: Variational bayes on discrete representation with self-annealed stochastic quantization, 2022. URL https://arxiv.org/abs/2205.07547.

[29] Aäron van den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[30] Georges Voronoi. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. deuxième mémoire. recherches sur les parallélloèdres primitifs. *Journal für die reine und angewandte Mathematik (Crelles Journal)*, 1908:198 – 287.

[31] Chengyi Wang, Sanyuan Chen, Yu Wu, Ziqiang Zhang, Long Zhou, Shujie Liu, Zhuo Chen, Yanqing Liu, Huaming Wang, Jinyu Li, et al. Neural codec language models are zero-shot text to speech synthesizers. *arXiv preprint arXiv:2301.02111*, 2023.

[32] 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 *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=pfNyExj7z2.

[33] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2(3):5, 2022.

[34] Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. Soundstream: An end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:495–507, 2021.

[35] Heiga Zen, Viet Dang, Rob Clark, Yu Zhang, Ron J. Weiss, Ye Jia, Zhifeng Chen, and Yonghui Wu. Libritts: A corpus derived from librispeech for text-to-speech, 2019. URL https://arxiv.org/abs/1904.02882.

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

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

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

[39] 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.1 Gradient Variance for Codebook Updates

This section derives Eq. (8) and discusses its relation to codebook update dynamics. The analysis is conducted from a fixed-assignment perspective. It does not model assignment switching, encoder-decoder coupling, or the full discontinuous optimization trajectory of VQ. Its purpose is to isolate how feature scale affects the update signal once a codeword has been selected.

Consider the standard codebook loss for a realized sample z assigned to codeword e_k:

$$\ell(z, e_k) = \| \operatorname{sg}[z] - e_k \|_2^2, \tag{16}$$

where $\operatorname{sg}[\cdot]$ denotes the stop-gradient operator. The gradient with respect to e_k is

$$g_k(z) = \nabla_{e_k} \ell(z, e_k) = 2(e_k - \operatorname{sg}[z]). \tag{17}$$

For channel d,

$$g_{k,d}(z) = 2(e_{k,d} - z_d). \tag{18}$$

Let

$$\mathcal{R}_k = \{ z \in \mathbb{R}^D \mid q(z) = k \} \tag{19}$$

be the Voronoi region of e_k, where $q(z)$ denotes the nearest-neighbor assignment. For $Z = f_\theta(X)$, conditioning on the assignment event $q(Z) = k$ gives

$$\begin{aligned} \operatorname{Var}(g_{k,d}(Z) \mid q(Z) = k) &= \operatorname{Var}(2(e_{k,d} - Z_d) \mid q(Z) = k) \\ &= 4 \operatorname{Var}(Z_d \mid q(Z) = k). \end{aligned} \tag{20}$$

Thus, within a fixed Voronoi region, the channel-wise variance of the codebook gradient is proportional to the conditional channel-wise feature variance.

For a random mini-batch $\mathcal{B} = \{Z_n\}_{n=1}^N$, let

$$\mathcal{B}_k = \{ Z_n \in \mathcal{B} \mid q(Z_n) = k \} \tag{21}$$

denote the subset of samples assigned to e_k. When $|\mathcal{B}_k| > 0$, the average gradient is

$$\bar{g}_k = \frac{1}{|\mathcal{B}_k|} \sum_{Z_n \in \mathcal{B}_k} 2(e_k - Z_n). \tag{22}$$

If the assigned samples are treated as conditionally independent, then

$$\operatorname{Var}(\bar{g}_{k,d} \mid |\mathcal{B}_k|) = \frac{4}{|\mathcal{B}_k|} \operatorname{Var}(Z_d \mid q(Z) = k). \tag{23}$$

This shows that the effective update signal depends on both the conditional feature scale and the number of samples assigned to the codeword. A codeword with low assignment frequency receives sparse updates, while the updates it does receive can still be dominated by high-variance channels inside its assigned region.

The same interpretation applies to EMA [24] or online k-means-style updates [19]. Although these updates are not written as explicit gradient descent steps, they also estimate codeword locations from the samples assigned to each region. Therefore, assignment frequency and conditional feature statistics remain the main factors controlling the effective update signal.

A.2 Step-by-Step Calculation for Figure 2

This section spells out how the cumulative update-energy shares reported in Figure 2 are computed. Throughout, $\mathcal{E}_\tau \in \mathbb{R}^{K \times D}$ denotes the learned codebook at checkpoint τ.

Splitting the codebook into scale and geometry. For each checkpoint we factor the codebook into a channel scale and a normalized geometry:

$$\mathcal{E}_\tau = \mathbf{G}_\tau \operatorname{diag}(\mathbf{s}_\tau), \qquad \mathbf{s}_\tau = \sigma(\mathcal{E}_\tau), \qquad \mathbf{G}_\tau = \mathcal{E}_\tau / \mathbf{s}_\tau. \tag{24}$$

Here $\sigma(\mathcal{E}_\tau) \in \mathbb{R}^D$ is the channel-wise standard deviation over codewords defined in Eq. (9), and the division is element-wise, applied column by column, so that $\sigma(\mathbf{G}_\tau) = \mathbf{1}$. As in Eq. (10), the

normalization used in the implementation does not subtract the mean, so the pair $(\mathbf{s}_\tau, \mathbf{G}_\tau)$ reproduces $\mathcal{E}_\tau$ exactly and no information is lost by the split.

Midpoint decomposition of the codebook change. For two adjacent checkpoints $\tau \in \{t-1, t\}$, define the differences and the midpoints

$$\Delta\mathbf{s} = \mathbf{s}_t - \mathbf{s}_{t-1}, \qquad \Delta\mathbf{G} = \mathbf{G}_t - \mathbf{G}_{t-1}, \qquad \bar{\mathbf{s}} = \frac{\mathbf{s}_t + \mathbf{s}_{t-1}}{2}, \qquad \bar{\mathbf{G}} = \frac{\mathbf{G}_t + \mathbf{G}_{t-1}}{2}. \tag{25}$$

Writing each of $\mathcal{E}_t$ and $\mathcal{E}_{t-1}$ in the form of Eq. (24) at this midpoint and cancelling the two cross terms gives the exact identity

$$\begin{aligned} \Delta\mathcal{E} &= \mathcal{E}_t - \mathcal{E}_{t-1} \\ &= \bar{\mathbf{G}}\,\mathrm{diag}(\Delta\mathbf{s}) + \Delta\mathbf{G}\,\mathrm{diag}(\bar{\mathbf{s}}). \end{aligned} \tag{26}$$

The first term changes only the channel scale of the codebook while keeping its normalized geometry fixed, and the second term does the converse.

Energy terms and accumulated shares. At checkpoint t we record the squared Frobenius norm of each term separately:

$$E^{(t)}_{\mathrm{scale}} = \left\|\bar{\mathbf{G}}\,\mathrm{diag}(\Delta\mathbf{s})\right\|_F^2, \qquad E^{(t)}_{\mathrm{geom}} = \left\|\Delta\mathbf{G}\,\mathrm{diag}(\bar{\mathbf{s}})\right\|_F^2. \tag{27}$$

Summing these two quantities over the checkpoints of a run gives the two cumulative energies drawn in Figure 2, and the reported shares are their normalized magnitudes,

$$\rho_{\mathrm{scale}} = \frac{\sum_t E^{(t)}_{\mathrm{scale}}}{\sum_t E^{(t)}_{\mathrm{scale}} + \sum_t E^{(t)}_{\mathrm{geom}}}, \qquad \rho_{\mathrm{geom}} = 1 - \rho_{\mathrm{scale}}. \tag{28}$$

Two remarks on how to read Eq. (28). First, the two terms of Eq. (26) need not be orthogonal, so ρ_{scale} and ρ_{geom} compare magnitudes rather than partition a variance, and they should be read as the share of codebook motion carried by each component. Second, since the sums run over a whole training run, ρ is a run-level statistic: it describes where the codebook spent its motion across training, and it does not attribute the motion of any single update step.

A.3 Experimental Details

Table 5 summarizes the main hyperparameter settings for the VQ-VAE-based ImageNet-1k evaluation and the VQGAN-based FFHQ evaluation. Within each benchmark, all compared methods share the same setup, and only the quantization module is replaced.

A.4 Metric Definitions

This section defines the codebook-usage and occupancy metrics reported in the main paper. Consider an evaluation set containing N quantized tokens with assignment indices $\{q_n\}_{n=1}^N$, where $q_n \in \{1, \ldots, K\}$ indexes one of the K codewords. Let

$$c_k = \sum_{n=1}^{N} \mathbb{1}[q_n = k], \qquad p_k = \frac{c_k}{N}. \tag{29}$$

Here c_k is the number of times codeword k is selected, and $p = (p_1, \ldots, p_K)$ is the empirical code-assignment distribution.

Utilization. We define codebook utilization (Util.) as the fraction of codewords that are used at least once:

$$\mathrm{Util.} = \frac{100}{K} \sum_{k=1}^{K} \mathbb{1}[c_k > 0]. \tag{30}$$

Thus, utilization measures codebook coverage, irrespective of how evenly the used codes are distributed.

Table 5: Main hyperparameters for the ImageNet-1k VQ-VAE and FFHQ VQGAN evaluations.

Item	ImageNet-1k (VQ-VAE)	FFHQ (VQGAN)
Input resolution	128×128	256×256
Batch size	128	64
Training epochs	5	80
GPU	L40S 48GB	2 A100 80GB
Learning rate	2×10^{-4}	5×10^{-4}
Optimizer	Adam [15]	AdamW [17]
Scheduler	Constant	Linear warmup and cosine decay
Backbone	VQGAN autoencoder [7]	Original VQGAN autoencoder [7]
Base channels	64	128
Channel multipliers	$[1, 1, 2, 4]$	$[1, 1, 2, 2, 4]$
Residual blocks	2	2
Attention	None	at 16×16 resolution
Commitment weight β	0.33	0.33
GAN loss weight	0.0	0.2
Perceptual loss weight	0.0	1.0
VQ loss weight	1.0	1.0
EMA factor γ	0.99	0.99
Codebook init. strategy	Gaussian	Gaussian

Perplexity. We define perplexity (PPL) from the empirical assignment distribution as

$$\mathrm{PPL} = \exp\left(-\sum_{k=1}^{K} p_k \log p_k\right), \tag{31}$$

with the convention $0 \log 0 = 0$. This is the exponential of the Shannon entropy of p, and can be interpreted as the effective number of codes used by the tokenizer. It reaches K when the assignments are perfectly uniform.

Occupancy JS divergence. Let $\{z_n\}_{n=1}^{N}$ be encoder latents and $\{\hat{z}_n\}_{n=1}^{N}$ be the corresponding quantized outputs collected from the same token positions. We fit a 2D PCA basis on $\{z_n\}$, project both sets into that basis, and denote the projected points by $\{u_n\}$ and $\{v_n\}$. On each axis, the histogram range is set by the 1st and 99th percentiles of $\{u_n\}$ and then expanded by 5% of that percentile span. Using the same $B \times B$ grid for both sets (with $B = 48$ in Figure 4), we form two 2D histograms and flatten them into nonnegative vectors $h^{(z)}, h^{(\hat{z})} \in \mathbb{R}^{B^2}$.

After normalization,

$$\tilde{p}_i = \frac{h_i^{(z)}}{\sum_j h_j^{(z)} + \varepsilon}, \qquad \tilde{q}_i = \frac{h_i^{(\hat{z})}}{\sum_j h_j^{(\hat{z})} + \varepsilon}, \tag{32}$$

where $\varepsilon = 10^{-8}$. The occupancy JS divergence is then defined as

$$\begin{aligned} \mathrm{JS}_{\mathrm{occ}}(\tilde{p}, \tilde{q}) &= \frac{1}{2}\,\mathrm{KL}(\tilde{p} \,\|\, m) + \frac{1}{2}\,\mathrm{KL}(\tilde{q} \,\|\, m), \\ m &= \frac{1}{2}(\tilde{p} + \tilde{q}), \end{aligned} \tag{33}$$

where the logarithm in KL is taken in base 2. Lower values indicate better agreement between the occupancy pattern of encoder latents and that of their quantized outputs.

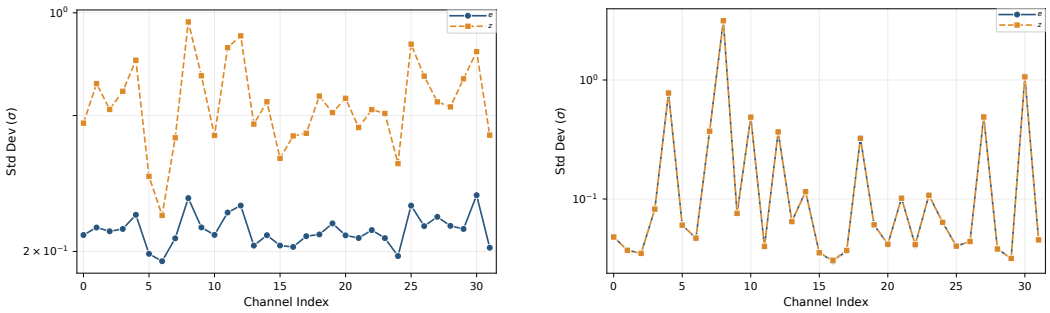
A.5 Visualization and Qualitative Results



Figure 6: **Reconstruction comparison on ImageNet-1k validation images.**



Figure 7: **Qualitative results on FFHQ.** Top: original images; bottom: reconstructed images.



(a) Vanilla VQ (b) ASVQ

Figure 8: **Channel variance alignment.**

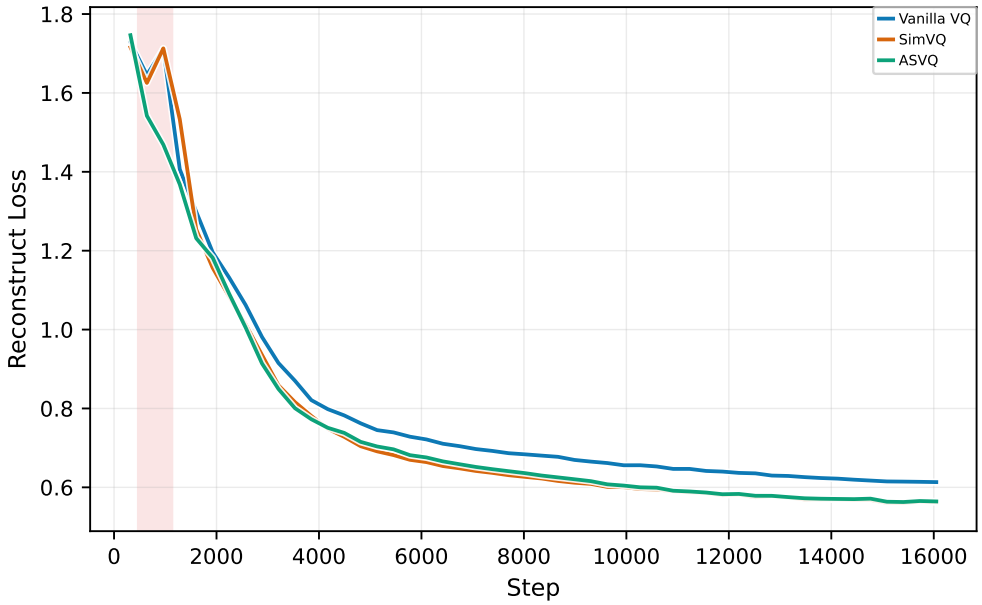


Figure 9: **Training stability analysis.**

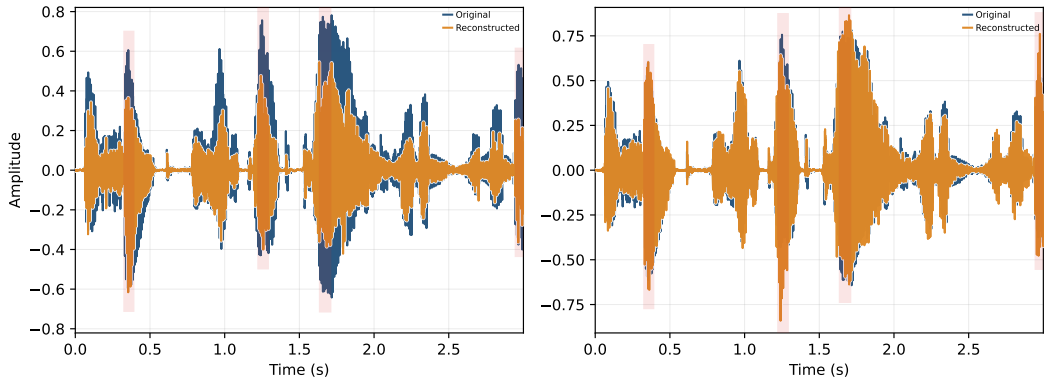


Figure 10: **Waveform comparison on LibriTTS-clean.** It is observed that when the codebook size increases from 4,096 to 65,536, the representational capability of ASVQ is significantly enhanced: the reconstructed waveform in the right panel exhibits more precise amplitude matching in high-energy transient regions, avoiding waveform collapse. This demonstrates that ASVQ allows for fine-grained modeling of speech details through a larger discrete space, realizing superior signal restoration.

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 core contribution of the paper clearly: ASVQ is a lightweight diagonal reparameterization for codebook learning, and the paper studies its effect on utilization, reconstruction quality, and training stability. These claims are then supported by the method section and by the image and audio experiments in Section 5, including ablations and qualitative 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: The paper discusses its main limitations explicitly in the "Limitations" paragraph of Section 6.

   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 whose contribution depends on a complete proof. Instead, Section 4 and Appendix A provide heuristic analysis and derivations intended to build intuition about ASVQ rather than a full formal theory.

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: Section 5 specifies the datasets, baselines, architectures, codebook sizes, and evaluation metrics for the main experiments, and Algorithm 1 "ASVQ Training Procedure" gives the training algorithm. Appendix "Experimental Details" further list the main hyperparameters used for the image benchmarks, so the main experimental pipeline is described in sufficient detail to reproduce the reported conclusions.

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: [Yes]

   Justification: The code is available on Github with sufficient instructions to guarantee reproducibility while the experiments rely on existing public data such as ImageNet-1k, FFHQ, and LibriTTS.

   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: The paper reports the training and evaluation setup for each benchmark in Section 5, including datasets, baselines, resolutions, latent dimension, codebook sizes, losses, batch sizes, learning rates, and training epochs. Appendix "Experimental Details" adds optimizer, scheduler, and shared hyperparameters for the image benchmarks, while Appendix "Metric Definitions" clarifies how the reported metrics are computed.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

Justification: While we report metrics on many experiments, the error bars are generally not report on large-scale datasets and complex models due to the computational constraints. Therefore, the submission does not provide a formal analysis of statistical significance beyond the observed differences in the reported metrics.

Guidelines:

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Appendix Table 5 reports the GPU type and memory for the ImageNet-1k and FFHQ experiments.

   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: This work is a methodological study on vector quantization using established public image and speech benchmarks, and it does not involve human subjects, crowdsourcing, or interventions on participants. The submission appears to stay within standard academic benchmark usage and does not describe practices that would conflict with 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: [No]

    Justification: The paper is focused on the technical design and empirical evaluation of codebook reparameterization methods, and it does not include a dedicated discussion of potential positive or negative societal impacts. Any downstream societal effects of improved discrete tokenization are therefore left outside the scope of the current submission.

    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 poses no such risks

    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 credits the original sources of the datasets, model backbones, and comparison methods used in the experiments through citations in Sections 5, including ImageNet-1k, FFHQ, LibriTTS, VQGAN, and WavTokenizer. All code, data, and models are public available and suitable for research.

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 work does not introduce a new released dataset, codebase, or pretrained model asset that requires separate documentation alongside the paper.

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 work does not involve crowdsourcing, participant recruitment, or any human-subject experiment.

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 paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: LLMs are not an important or non-standard component of the proposed method in this research.

    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.