Distributional Matching for Vector Quantization: A Unified Theoretical and Empirical Framework

Xianghong Fang[1*] **Litao Guo**[2] **Hengchao Chen**[1] **Yuxuan Zhang**[1] **XiaofanXia**[1]
Dingjie Song[4] **Yexin Liu**[2] **Hao Wang**[5] **Harry Yang**[2] **Qiang Sun**[1] **Yuan Yuan**[3*]
[1]University of Toronto [2]The Hong Kong University of Science and Technology [3]Boston College
[4]Lehigh University [5]Southern University of Science and Technology

Website Code & Models

Abstract

The effectiveness of modern visual representation learning and autoregressive models critically depends on vector quantization (VQ), which discretizes continuous feature representations using a learnable codebook. Despite its widespread use, existing VQ methods often suffer from training instability and codebook collapse, arising from gradient mismatch induced by the straight-through estimator and the under-utilization of code vectors. In this work, we show that both issues can be traced to a fundamental mismatch between the distributions of feature vectors and code vectors, leading to inefficient representation and information loss. Building on this observation, we propose a distributional matching framework for vector quantization. We introduce principled criteria for desirable VQ behavior and demonstrate through theoretical analysis and empirical evaluation that aligning feature and code vector distributions provides a unifying mechanism for mitigating training instability and codebook collapse. We instantiate this framework using a Wasserstein-based objective with an efficient closed-form under a mild Gaussian approximation, and further show that a nonparametric alternative based on maximum mean discrepancy yields comparable performance. Extensive experiments on visual tokenization benchmarks support the effectiveness and robustness of the proposed approach.

1 Introduction

Vector quantization (VQ) [49] is a fundamental building block in a wide range of modern representation learning and visual tokenization frameworks, including autoregressive and hybrid generative models [39, 10, 7, 28, 15, 47, 29]. By discretizing continuous feature representations using a learnable codebook, VQ enables compact and structured latent representations that are well suited for downstream modeling. Despite its broad adoption, VQ remains notoriously difficult to optimize in practice, often exhibiting unstable training dynamics and severe codebook collapse.

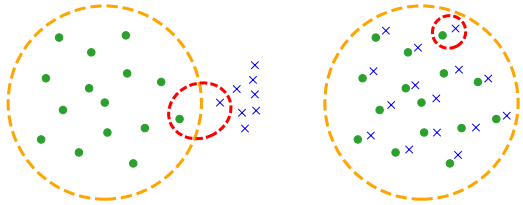
The first challenge stems from the non-differentiability of the quantization operation, which prevents gradients from being directly propagated from quantized features to their continuous counterparts. To address this issue, prior work introduces the straight-through estimator (STE) [2, 49], which approximates gradients by copying them from the quantized features to encoder outputs. However, the effectiveness of this approximation critically depends on the magnitude of the quantization error. When the discrepancy between continuous features and their assigned code vectors becomes large, the resulting gradient mismatch can lead to unstable optimization and degraded training behavior [28].

* Corresponding authors. Email: xianghong.fang@mail.utoronto.ca, yuanyua@bc.edu.

Preprint.

A second, closely related challenge is codebook collapse, where only a small subset of code vectors receives assignments while the majority remain unused. From a geometric perspective, this corresponds to a degenerate Voronoi partition[1] in which most cells are never activated [59]. Although extensive research has sought to alleviate this problem, low code vector utilization often remains in practice, particularly with large codebooks [9, 46, 53, 28, 59]. This limitation is further exacerbated as increasing the codebook size expands the number of Voronoi cells, making it substantially harder to ensure that all cells are sufficiently populated.

In this work, we examine these challenges from a distributional perspective. Rather than treating training instability and codebook collapse as separate phenomena, we observe that both issues are closely tied to a fundamental mismatch between the distributions of feature vectors produced by the encoder and the distributions of the learnable code vectors. Figure 1 illustrates this intuition using two representative scenarios. When the two distributions are poorly aligned, feature vectors concentrate around a small subset of code vectors, resulting in large quantization errors and low codebook utilization. In contrast, when the distributions are well matched, quantization errors are reduced and codebook utilization approaches its maximum.



Distributional Mismatch Distributional Match

Figure 1: The symbols $\cdot$ and $\times$ represent the feature and code vectors, respectively. The left figure illustrates the distributional mismatch between the feature and code vectors, while the right figure visualizes their distributional match.

Building on this observation, we introduce three principled criteria that characterize the desirable behavior in vector quantization. Guided by this criterion triple, we show through both theoretical analysis and empirical evaluation that distributional alignment between feature vectors and code vectors provides a unifying mechanism for mitigating training instability and codebook collapse. To operationalize this idea, we adopt a distribution matching objective based on the quadratic Wasserstein distance. Under a mild Gaussian approximation, this objective admits a closed-form expression that can be efficiently optimized during training. Importantly, we show that even when this approximation is relaxed, a nonparametric alternative based on maximum mean discrepancy (MMD) [14, 44, 12] yields comparable performance, suggesting that distributional matching effectively captures the essential structure required for effective vector quantization. These insights translate into consistent improvements in reconstruction fidelity across visual tokenization benchmarks.

2 A Distribution Matching Perspective on Vector Quantization

This section introduces a novel distributional perspective for understanding vector quantization. By defining three principled criteria for VQ evaluation, we provide empirical and theoretical evidence that aligning feature and code vector distributions yields a near-optimal VQ solution.

2.1 An Overview of Vector Quantization

As a core component of visual tokenizers [49, 28, 47], VQ acts as a compressor that discretizes continuous latent features into discrete visual tokens by mapping them to the nearest code vectors within a learnable codebook.

Figure 2 illustrates the classic VQ process [49], which consists of an encoder $E(\cdot)$, a decoder $D(\cdot)$, and an updatable codebook $\{\mathbf{e}_k\}_{k=1}^{K} \in \mathbb{R}^d$ containing a finite set of code vectors. Here, K denotes the codebook size and d the code vector dimension. Given an image $\boldsymbol{x} \in \mathbb{R}^{H\times W\times 3}$, the goal is to derive a spatial collection of codeword IDs $r \in \mathbb{N}^{h\times w}$ as image tokens. This is achieved by encoding the image to obtain $\boldsymbol{z}_e = E(\boldsymbol{x}) \in \mathbb{R}^{h\times w\times d}$, followed by a spatial quantizer $\mathcal{Q}(\cdot)$ that maps each spatial feature $\boldsymbol{z}_e^{ij}$ to its

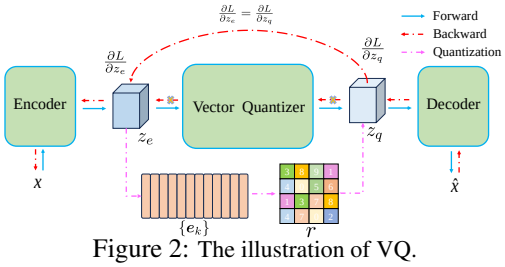


Figure 2: The illustration of VQ.

[1] Appendix. B discusses codebook collapse via Voronoi partitioning.

nearest code vector $\boldsymbol{e}_k$:

$$r^{ij} = \arg\min_k \|\boldsymbol{z}_e^{ij} - \boldsymbol{e}_k\|_2^2. \tag{1}$$

The resulting tokens retrieve the corresponding codebook entries $\boldsymbol{z}_q^{ij} = \mathcal{Q}(\boldsymbol{z}_e^{ij}) = \boldsymbol{e}_{r^{ij}}$, which are then passed through the decoder to reconstruct the image as $\widehat{\boldsymbol{x}} = D(\boldsymbol{z}_q)$. Despite its widespread adoption in visual tokenization, representation learning, and high-fidelity image synthesis [10], VQ still faces two key challenges: training instability and codebook collapse.

Training Instability This issue arises because during backpropagation, the gradient of $\boldsymbol{z}_q$ cannot flow directly to $\boldsymbol{z}_e$ due to the non-differentiable function $\mathcal{Q}$. To optimize the encoder's parameters, VQ-VAE [49] employs the straight-through estimator (STE) [3], which copies gradients from $\boldsymbol{z}_q$ to $\boldsymbol{z}_e$. However, this approach carries significant risks, especially when $\boldsymbol{z}_q$ and $\boldsymbol{z}_e$ are far apart. In such cases, the gradient gap can grow substantially, destabilizing training[2]. In this work, we tackle the training instability challenge from a distributional viewpoint. This perspective highlights that training instability is not merely an implementation artifact, but a consequence of systematic mismatch between feature and code distributions.

Codebook Collapse Codebook collapse occurs when only a small subset of code vectors receives gradients, while most remain unrepresentative and unupdated [9, 53, 28, 59]. Researchers have proposed solutions such as improved codebook initialization [60], reinitialization strategies [9, 51], and classical clustering algorithms like k-means [5] and k-means++[1] for codebook optimization [39, 59]. Beyond deterministic approaches that select the best-matching token, stochastic quantization strategies have also been explored [56, 38, 46]. However, these methods still exhibit low code vector utilization, particularly with large codebook sizes K [59, 33]. In this paper, we address this issue via distributional matching between feature and code vectors.

2.2 Evaluation Criteria

While these quantities have appeared individually in prior work, we emphasize that considering them jointly provides a unified lens for analyzing both optimization stability and codebook collapse.

Given feature vectors $\{\boldsymbol{z}_i\}_{i=1}^N$ from feature distribution $\mathcal{P}_A$ and code vectors $\{\boldsymbol{e}_k\}_{k=1}^K$ sampled from codebook distribution $\mathcal{P}_B$, vector quantization involves finding the nearest, and thus most representative, code vector for each feature:

$$\boldsymbol{z}_i' = \arg\min_{\boldsymbol{e}\in\{\boldsymbol{e}_k\}} \|\boldsymbol{z}_i - \boldsymbol{e}\|. \tag{2}$$

Each original feature vector $\boldsymbol{z}_i$ is then quantized to $\boldsymbol{z}_i'$. We denote the index of the selected code vector for the i-th feature as r_i, such that $\boldsymbol{z}_i' = \boldsymbol{e}_{r_i}$. We next introduce three key criteria to evaluate this process.

Criterion 1 (Quantization Error)**.** *The quantization error measures the average distortion introduced and is defined as:*

$$\mathcal{E}(\{\boldsymbol{e}_k\};\{\boldsymbol{z}_i\}) = \frac{1}{N}\sum_i \|\boldsymbol{z}_i - \boldsymbol{z}_i'\|^2. \tag{3}$$

A smaller $\mathcal{E}$ signifies a more accurate quantization of the original feature vectors. Beyond this geometric interpretation, $\mathcal{E}$ is also directly linked to training stability. As formally derived in Appendix C, the gradient discrepancy $\mathcal{G}_i$ for a feature $\boldsymbol{z}_i$ introduced by the straight-through estimator (STE) is theoretically bounded by its distance to the code vector:

$$\mathcal{G}_i \le \|\mathbf{H}\|_2 \cdot \|\boldsymbol{z}_i - \boldsymbol{z}_i'\|, \tag{4}$$

where $\mathbf{H} = \frac{\partial^2 \mathcal{L}}{\partial x^2}\Big|_{x=\boldsymbol{z}_i'}$ denotes the Hessian of the overall training loss $\mathcal{L}$ with respect to the latent input, evaluated at the quantized code $\boldsymbol{z}_i'$, characterizing the local curvature of the loss landscape. Since $\mathcal{E}$ is the mean squared magnitude of the term $\|\boldsymbol{z}_i - \boldsymbol{z}_i'\|$, minimizing $\mathcal{E}$ explicitly tightens this bound. This ensures that the approximated gradients remain faithful to the true gradients, effectively mitigating the gradient mismatch that causes training instability.

[2]Appendix C details the gradient-quantization relationship.

Criterion 2 (Codebook Utilization Rate)**.** *The codebook utilization rate is the fraction of code vectors used in VQ*

$$\mathcal{U}(\{\boldsymbol{e}_k\};\{\boldsymbol{z}_i\}) = \frac{1}{K}\sum_{k=1}^{K}\mathbf{1}\left(k \in \{r_i\}_{i=1}^{N}\right), \tag{5}$$

where the indicator evaluates to 1 if code vector $\boldsymbol{e}_k$ is selected by at least one feature.

A higher $\mathcal{U}$ reduces the risk of codebook collapse. Ideally, $\mathcal{U}$ should reach 100%, meaning all code vectors are utilized. As discussed in Appendix D, $\mathcal{U}$ measures only the *completeness* of codebook utilization and cannot evaluate the degree of collapse. This motivates introducing the codebook perplexity criterion.

Criterion 3 (Codebook Perplexity)**.** *The codebook perplexity measures the uniformity of codebook utilization in VQ*

$$\mathcal{C}(\{\boldsymbol{e}_k\};\{\boldsymbol{z}_i\}) = \exp(-\sum_{k=1}^{K} p_k \log p_k), \tag{6}$$

where $p_k = \frac{1}{N}\sum_{i=1}^{N}\mathbf{1}(\boldsymbol{z}_i' = \boldsymbol{e}_k)$. A higher $\mathcal{C}$ indicates more uniform selection of code vectors in VQ. Ideally, $\mathcal{C}$ reaches its maximum at $\mathcal{C}_0 = \exp(-\sum_{k=1}^{K}\frac{1}{K}\log\frac{1}{K}) = K$ when code vectors are completely uniformly utilized. Thus, as a complement to Criterion 2, $\mathcal{U}$ combined with $\mathcal{C}$ effectively evaluates codebook collapse.

We refer to $(\mathcal{E},\mathcal{U},\mathcal{C})$ as the criterion triple. When comparing extreme cases of distributional match and mismatch shown in Figure 1, we find that distributional matching significantly outperforms mismatching across all three criteria. Using this triple, we can analyze the benefits of distribution matching more systematically.

Quantization Error vs. Codebook Utilization Rate Quantization error ($\mathcal{E}$) is a more fundamental objective than codebook utilization ($\mathcal{U}$) in VQ. We theoretically show in [11] that minimizing $\mathcal{E}$ necessarily induces full codebook utilization, whereas the converse does not hold. This hierarchical relationship suggests treating $\mathcal{E}$ as the primary optimization target, with $\mathcal{U}$ serving as a complementary metric to monitor coverage.

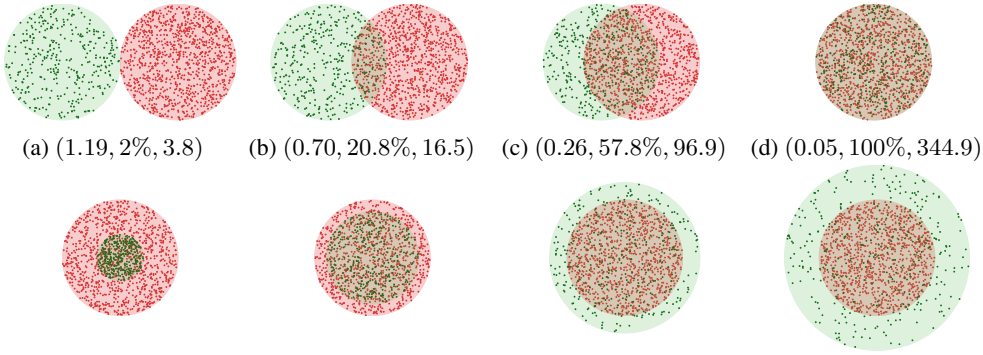
Remark Notably, $\mathcal{E}$ is sensitive to the variance of the latent feature distribution (Appendix E). Consequently, direct comparisons of raw $\mathcal{E}$ values across latent spaces with different variances can be misleading. To fairly evaluate quantization methods via criterion triple, all comparisons should be made under identical latent distributions.

2.3 The Effects of Distribution Matching

We conduct a deliberately simple synthetic experiment to isolate the geometric effects of distribution mismatch (see details in Appendix J.1). Specifically, we assume that $\mathcal{P}_A$ and $\mathcal{P}_B$ are uniform distributions within two distinct disks, as shown in Figure 3. We sample feature vectors $\{\boldsymbol{z}_i\}_{i=1}^{N}$ from the red disk and code vectors $\{\boldsymbol{e}_k\}_{k=1}^{K}$ from the green circle. The criterion triple $(\mathcal{E},\mathcal{U},\mathcal{C})$ is calculated by Criteria 1 to 3.

We examine two cases. The first involves disks with identical radii but different centers. As shown in Figures 3a to 3d, when the centers move closer, the criterion triple improves toward optimal values. Specifically, $\mathcal{E}$ decreases from 1.19 to 0.05, $\mathcal{U}$ rises from 2% to 100%, and $\mathcal{C}$ increases from 3.8 to 344.9. The second case shows distributions with identical centers but different radii. When the codebook support lies within the feature support (Figures 3e and 3f), $\mathcal{E}$ is larger, $\mathcal{U}$ slightly lower, and $\mathcal{C}$ smaller compared to the aligned distributions in Figure 3d. When the codebook support extends beyond the feature support, $\mathcal{E}$ increases modestly while $\mathcal{U}$ and $\mathcal{C}$ drop significantly (Figures 3g and 3h). Detailed explanations are provided in Appendix F.

Overall, VQ achieves the optimal criterion triple when feature and codebook distributions are identical. This is further supported by quantitative analyses in Appendix G.



(a) $(1.19, 2\%, 3.8)$ (b) $(0.70, 20.8\%, 16.5)$ (c) $(0.26, 57.8\%, 96.9)$ (d) $(0.05, 100\%, 344.9)$

(e) $(0.36, 93.3\%, 63.2)$ (f) $(0.10, 99.8\%, 250.5)$ (g) $(0.07, 61.3\%, 199.7)$ (h) $(0.08, 45.3\%, 151.5)$

Figure 3: Qualitative analyses of the criterion triple $(\mathcal{E}, \mathcal{U}, \mathcal{C})$: The red and green disks represent the uniform distributions of feature and code vectors, respectively.

2.4 Theoretical Analyses

In this section, we provide theoretical evidence to support our empirical observations. Let the code vectors $\{\boldsymbol{e}_k\}_{k=1}^K$ and feature vectors $\{\boldsymbol{z}_i\}_{i=1}^N$ be independently and identically drawn from $\mathcal{P}_B$ and $\mathcal{P}_A$, respectively. We say a codebook $\{\boldsymbol{e}_k\}_{k=1}^K$ attains full utilization asymptotically with respect to $\{\boldsymbol{z}_i\}_{i=1}^N$ if the codebook utilization rate $\mathcal{U}(\{\boldsymbol{e}_k\}_{k=1}^K; \{\boldsymbol{z}_i\}_{i=1}^N)$ tends to 1 in probability as N approaches infinity:

$$\mathcal{U}(\{\boldsymbol{e}_k\}_{k=1}^K; \{\boldsymbol{z}_k\}_{i=1}^N) \xrightarrow{p} 1, \quad \text{as } N \to \infty. \tag{7}$$

For the codebook distribution $\mathcal{P}_B$, we say it attains full utilization asymptotically with respect to $\mathcal{P}_A$ if, with probability 1, the randomly generated codebook $\{\boldsymbol{e}_k\}_{k=1}^K$ achieves full utilization asymptotically.

Additionally, a codebook distribution $\mathcal{P}_B$ is said to have vanishing quantization error asymptotically with respect to a domain $\Omega \subseteq \mathbb{R}^d$ if the quantization error over all data of size N tends to zero in probability as K approaches infinity:

$$\sup_{\{\boldsymbol{z}_i\} \subseteq \Omega} \mathcal{E}(\{\boldsymbol{e}_k\}_{k=1}^K; \{\boldsymbol{z}_i\}_{i=1}^N) \xrightarrow{p} 0, \quad \text{as } K \to \infty. \tag{8}$$

Our first theorem shows that $\overline{\text{supp}(\mathcal{P}_A)} = \overline{\text{supp}(\mathcal{P}_B)}$ is sufficient and necessary for the codebook distribution $\mathcal{P}_B$ to attain both full utilization and vanishing quantization error asymptotically. For simplicity, $\mathcal{P}_A$ is assumed to have a density function f_A with bounded support $\Omega \subseteq \mathbb{R}^d$.

Theorem 1. *Assume $\Omega = \text{supp}(\mathcal{P}_A)$ is a bounded open area. The codebook distribution $\mathcal{P}_B$ attains full utilization and vanishing quantization error asymptotically if and only if $\overline{\text{supp}(\mathcal{P}_B)} = \overline{\text{supp}(\mathcal{P}_A)}$, where $\overline{\mathcal{S}}$ denotes the closure of the set $\mathcal{S}$.*

Theorem 1 establishes the optimal support of the codebook distribution. The boundedness of Ω is required as we consider the worst case quantization error in equation 8. In real applications, when $\mathcal{P}_A$ follows an absolutely continuous distribution over an unbounded domain, then $\{z_i\}_{i=1}^N$ generated from $\mathcal{P}_A$ will be bounded with high probability. Thus, Theorem 1 also provides theoretical insights for a target distribution $\mathcal{P}_A$ with an unbounded domain.

Besides the optimal support, we also determine the optimal density of the codebook distribution by invoking existing results characterizing asymptotic optimal quantizers [13]. Specifically, we consider the case where N approaches to infinity and define the expected quantization error of a codebook $\{\boldsymbol{e}_k\}$ with respect to $\mathcal{P}_A$ as

$$\mathcal{E}(\{\boldsymbol{e}_k\}_{k=1}^K; \mathcal{P}_A) = \mathbb{E}_{\boldsymbol{z} \sim \mathcal{P}_A} \min_{\boldsymbol{e} \in \{\boldsymbol{e}_k\}} \|\boldsymbol{z} - \boldsymbol{e}\|^2. \tag{9}$$

A codebook $\{\boldsymbol{e}_k^*\}_{k=1}^K$ is called the set of optimal centers for $\mathcal{P}_A$ if it achieves the minimal quantization error:

$$\mathcal{E}(\{\boldsymbol{e}_k^*\}_{k=1}^K; \mathcal{P}_A) = \min_{\{\boldsymbol{e}_k\}_{k=1}^K} \mathcal{E}(\{\boldsymbol{e}_k\}_{k=1}^K; \mathcal{P}_A). \tag{10}$$

Intuitively, Theorem 1 shows that mismatched supports inevitably lead to either unused code vectors or large quantization error, whereas matching supports is both necessary and sufficient to avoid these failure modes.

Theorem 2 demonstrates that, under weak regularity conditions, the empirical measure of the optimal centers for $\mathcal{P}_A$ converges in distribution to a fixed distribution determined by $\mathcal{P}_A$. Notably, we do not assume a bounded domain in the following theorem.

Theorem 2 (Theorem 7.5, [13])**.** *Suppose $Z \sim \mathcal{P}_A$ is absolutely continuous w.r.t. the Lesbegue measure in $\mathbb{R}^d$ and $\mathbb{E}\|Z\|^{2+\delta} < \infty$ for some $\delta > 0$. Then the empirical measure of the optimal centers for $\mathcal{P}_A$,*

$$\frac{1}{K}\sum_{k=1}^{K}\delta_{\boldsymbol{e}_k^*}, \tag{11}$$

converges weakly to a fixed distribution $\mathcal{P}_A^*$, whose density function f_A^* is proportional to $f_A^{(d+2)/d}$.

High-dimensional Implication. Theorem 2 characterizes the asymptotically optimal codebook distribution $\mathcal{P}_A^*$, whose density satisfies $f_A^*(z) \propto f_A(z)^{(d+2)/d}$. A key implication arises in high-dimensional settings typical of visual tokenization: as the feature dimension d increases, the exponent converges to unity,

$$\lim_{d\to\infty}\frac{d+2}{d} = \lim_{d\to\infty}\left(1+\frac{2}{d}\right) = 1. \tag{12}$$

As a result, the optimal codebook density f_A^* becomes increasingly close to the feature density f_A in the high-dimensional regime. This observation provides a rigorous theoretical justification for our approach: explicitly aligning the codebook distribution with the feature distribution (i.e., $\mathcal{P}_B \approx \mathcal{P}_A$) serves as a principled and effective surrogate for the asymptotically optimal design in high-dimensional latent spaces.

3 Methodology

Based on the theoretical insights in Section 2, which show that matching the feature and codebook distributions is both necessary and sufficient for asymptotically optimal quantization, we propose a general framework for enhancing vector quantization via distributional matching. This framework can be instantiated using either parametric objectives (e.g., the Wasserstein distance for efficiency) or non-parametric alternatives (e.g., Maximum Mean Discrepancy, which offers robustness and adaptability to general distributions). In the main text, we focus primarily on the Wasserstein instantiation due to its computational efficiency arising from a closed-form solution. Finally, we integrate this objective into two representative VQ frameworks, namely VQ-VAE [49] and VQGAN [10].

3.1 A General Distributional Matching Framework

To resolve the fundamental mismatch between the feature distribution $\mathcal{P}_A$ and the codebook distribution $\mathcal{P}_B$, we introduce an auxiliary alignment loss $\mathcal{L}_{match}$ to the standard VQ objective. The goal is to minimize the divergence $\mathcal{D}$ between the two distributions:

$$\mathcal{L}_{match} = \mathcal{D}(\mathcal{P}_A, \mathcal{P}_B), \tag{13}$$

where $\mathcal{P}_A$ is empirically defined by the encoder outputs $\{\boldsymbol{z}_e\}$ and $\mathcal{P}_B$ by code vectors $\{\boldsymbol{e}_k\}$. Crucially, gradients from $\mathcal{L}_{match}$ are only back-propagated to the codebook, preserving encoder feature expressiveness.

3.2 Wasserstein-Based Distribution Matching

We consider a parametric instantiation of the proposed framework using a mild Gaussian approximation to enable computationally efficient distributional matching. Specifically, we approximate the feature and codebook distributions as $\mathcal{P}_A \approx \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $\mathcal{P}_B \approx \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$. Under this approximation, we employ the quadratic Wasserstein distance, as defined in Appendix H, which admits a computationally efficient closed-form solution. Although other statistical distances, such as the

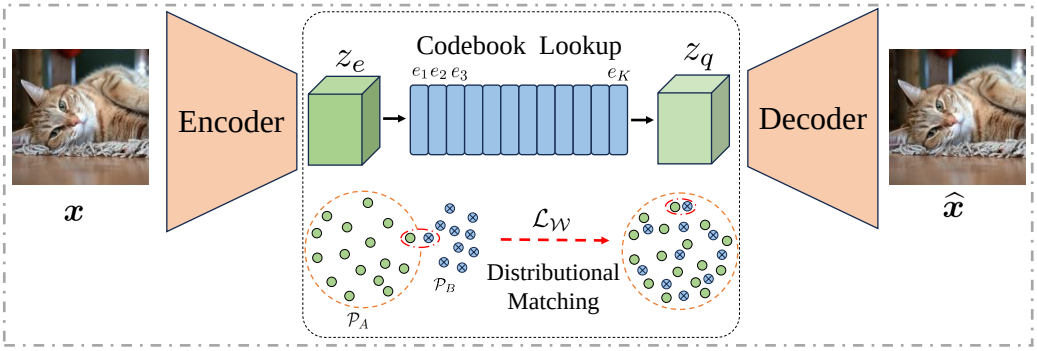


Figure 4: Illustration of the *Wasserstein VQ*. The architecture integrates an encoder-decoder network with a VQ module. In the VQ module, we augment the vanilla VQ framework [49] by incorporating our proposed Wasserstein loss $\mathcal{L}_{\mathcal{W}}$ to achieve distributional matching between features $\boldsymbol{z}_e$ ($\boldsymbol{z}_e^{ij} \sim \mathcal{P}_A$) and the codebook $\boldsymbol{e}_k$ ($\boldsymbol{e}_k \sim \mathcal{P}_B$).

Kullback-Leibler divergence [26, 17], are viable alternatives, they typically lack simple closed-form representations, rendering them computationally prohibitive in high-dimensional settings. Consequently, we adopt the quadratic Wasserstein distance, whose closed-form representation for Gaussian distributions is provided by the following lemma:

Lemma 3 ([36])**.** *The quadratic Wasserstein distance between* $\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ *and* $\mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$

$$\sqrt{\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\|_2^2 + \mathrm{tr}(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2 - 2(\Sigma_1^{\frac{1}{2}}\boldsymbol{\Sigma}_2\boldsymbol{\Sigma}_1^{\frac{1}{2}})^{\frac{1}{2}})}. \tag{14}$$

The lemma shows that the quadratic Wasserstein distance admits a closed-form expression in terms of the means and covariance matrices of the two distributions. In practice, we estimate these population quantities, $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$, $\boldsymbol{\Sigma}_1$, and $\boldsymbol{\Sigma}_2$, with their sample counterparts: $\widehat{\boldsymbol{\mu}}_1$, $\widehat{\boldsymbol{\mu}}_2$, $\widehat{\boldsymbol{\Sigma}}_1$, and $\widehat{\boldsymbol{\Sigma}}_2$. The empirical quadratic Wasserstein distance is then used as the optimization objective to align the feature and codebook distributions:

$$\mathcal{L}_{\mathcal{W}} = \sqrt{\|\widehat{\boldsymbol{\mu}}_1 - \widehat{\boldsymbol{\mu}}_2\|_2^2 + \mathrm{tr}(\widehat{\boldsymbol{\Sigma}}_1 + \widehat{\boldsymbol{\Sigma}}_2 - 2(\widehat{\boldsymbol{\Sigma}}_1^{\frac{1}{2}}\widehat{\boldsymbol{\Sigma}}_2\widehat{\boldsymbol{\Sigma}}_1^{\frac{1}{2}})^{\frac{1}{2}})}.$$

A smaller value of $\mathcal{L}_{\mathcal{W}}$ indicates stronger alignment between the feature distribution $\mathcal{P}_A$ and the codebook distribution $\mathcal{P}_B$. We refer to the VQ algorithm that employs $\mathcal{L}_{\mathcal{W}}$ as *Wasserstein VQ*.

Remark on the Gaussian Approximation The Gaussian approximation is introduced solely to obtain a closed-form and computationally efficient instantiation of the proposed distributional matching objective, simplifying the complex form in Definition 4 in Appendix H into Lemma 3. Importantly, this approximation does not constrain the learned feature representations: the Wasserstein loss $\mathcal{L}_{\mathcal{W}}$ is applied only to update the codebook parameters, while gradients to the encoder are detached. As a result, the encoder remains free to learn arbitrarily complex feature distributions.

In practice, latent representations in visual tokenization often exhibit approximately Gaussian statistics, a behavior commonly observed in deep representations (see Section 5 and Appendix I for empirical validation). One possible explanation is that multivariate normal distributions are information-theoretically advantageous, as they maximize entropy for a given covariance, thereby enabling efficient information transmission and robust signal reconstruction. As a result, multivariate normal distributions naturally emerge as both an ideal and empirically observed form for latent representations in visual tokenization and related signal recovery tasks.

3.3 MMD-Based Distribution Matching

To address scenarios where the Gaussian assumption may not hold (e.g., highly multi-modal distributions), we provide a non-parametric instantiation using Maximum Mean Discrepancy (MMD) [14, 44]. MMD measures the distance between distributions in a Reproducing Kernel Hilbert Space (RKHS) without assuming any specific parametric form:

$$\mathcal{L}_{\text{MMD}} = \frac{1}{N^2}\sum_{i,j} k(\boldsymbol{z}_i, \boldsymbol{z}_j) - \frac{2}{NK}\sum_{i,k} k(\boldsymbol{z}_i, \boldsymbol{e}_k) + \frac{1}{K^2}\sum_{k,l} k(\boldsymbol{e}_k, \boldsymbol{e}_l), \tag{15}$$

where $\boldsymbol{e}_k$ and $\boldsymbol{z}_i$ denote code vectors and encoder output spatial feature vectors, respectively, and $k(\cdot, \cdot)$ is a kernel function (e.g., a Gaussian RBF kernel). We refer to this approach as MMD VQ [12]. Similar to Wasserstein VQ, the MMD loss $\mathcal{L}_{\text{MMD}}$ is applied exclusively to update the codebook parameters, with gradients to the encoder detached.

Remark While $\mathcal{L}_{\text{MMD}}$ offers greater theoretical robustness by relaxing the Gaussian hypothesis, it typically incurs higher computational cost ($\mathcal{O}((N+K)^2)$) compared to the closed-form Wasserstein distance. In our experiments, we observe that Wasserstein VQ achieves performance comparable to MMD VQ (see Section 6), indicating that the Gaussian approximation sufficiently captures the essential structure for effective tokenization while remaining computationally efficient. Therefore, In the following experiments, we focus on Wasserstein instantiation for exploring the distributional matching framework.

3.4 Integration into VQ Architectures

Our distributional matching framework is agnostic to the underlying VQ architecture. We integrate it into two representative frameworks: VQ-VAE and VQGAN.

3.4.1 VQ-VAE Integration

We first examine *Wasserstein VQ* within the VQ-VAE framework [49]. As shown in Figure 4, the VQ-VAE model has three key components: an encoder $E(\cdot)$, a decoder $D(\cdot)$, and a quantizer $\mathcal{Q}(\cdot)$ with a learnable codebook $\{\mathbf{e}_k\}_{k=1}^K$. As described earlier in Section 2.1, for an input image $\boldsymbol{x}$, the encoder produces a spatial feature $\boldsymbol{z}_e = E(\boldsymbol{x}) \in \mathbb{R}^{h \times w \times d}$. The quantizer maps $\boldsymbol{z}_e$ to a quantized feature $\boldsymbol{z}_q$, which the decoder uses to reconstruct the image as $\widehat{\boldsymbol{x}} = D(\boldsymbol{z}_q)$. Incorporating our Wasserstein loss $\mathcal{L}_{\mathcal{W}}$ into the VQ-VAE framework, the overall loss is formulated as follows:

$$\mathcal{L}_{\text{VQ-VAE}} = \|\widehat{\boldsymbol{x}} - \boldsymbol{x}\|_2^2 + \beta\|\text{sg}(\boldsymbol{z}_q) - \boldsymbol{z}_e\|_2^2 + \|\text{sg}(\boldsymbol{z}_e) - \boldsymbol{z}_q\|_2^2 + \gamma\mathcal{L}_{\mathcal{W}}. \tag{16}$$

where sg denotes the stop-gradient. β and γ are hyper-parameters. We set $\gamma = 0.5$ for all VQ-VAE experiments.

By incorporating $\mathcal{L}_{\mathcal{W}}$, the codebook is encouraged to globally match the first- and second-order statistics of the feature distribution, complementing the local assignment enforced by the standard VQ loss.

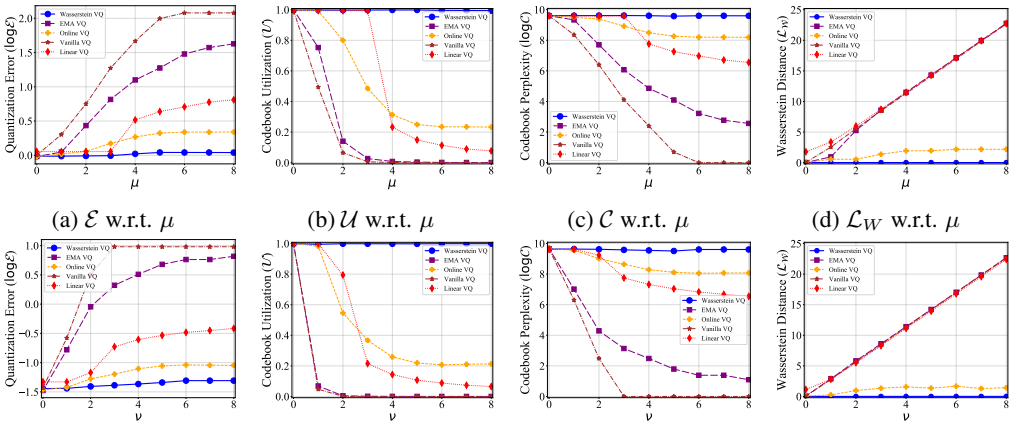
Remark Sections 2.3 and 2.4 provide empirical and theoretical evidence that minimizing quantization error is closely linked to aligning the codebook with the feature distribution. By encouraging the codebook to reflect the feature space structure and density, the alignment loss $\mathcal{L}_{\mathcal{W}}$ positions code vectors as cluster centers, minimizing the average squared distance between feature vectors and their assigned codes, $\|\boldsymbol{z}_e - \boldsymbol{z}_q\|_2^2$. In essence, distribution alignment arranges prototypes to cover the feature manifold and fill gaps, reducing overall quantization error.

3.4.2 VQGAN Integration

To ensure high perceptual quality in the reconstructed images, we further investigate *Wasserstein VQ* within the VQGAN framework [10]. VQGAN extends the VQ-VAE framework by integrating a VGG network [43] and a patch-based discriminator [10, 22]. The overall training objective of VQGAN can be written as follows:

$$\mathcal{L}_{\text{VQGAN}} = \mathcal{L}_{\text{VQ-VAE}} + \mathcal{L}_{\text{Per}} + \lambda\mathcal{L}_{\text{GAN}}. \tag{17}$$

Where $\mathcal{L}_{\text{Per}}$ and $\mathcal{L}_{\text{GAN}}$ denote the VGG-based perceptual loss [57] and the GAN loss [21, 30], respectively. Achieving state-of-the-art reconstruction fidelity via adversarial training typically incurs substantial computational cost. To reduce training overhead, we adopt the VQ-Transplant framework, which attains competitive reconstruction fidelity at a fraction of the cost [12]. We adopt the VQ-Transplant framework solely as a training-efficient backbone; the proposed Wasserstein VQ is orthogonal to this choice and can be integrated into standard VQGAN training as well. Specifically, it initializes the encoder and decoder from a state-of-the-art pretrained tokenizer (e.g., VAR [47]) rather than training from scratch, significantly lowering training cost while preserving reconstruction quality.



(a) $\mathcal{E}$ w.r.t. μ (b) $\mathcal{U}$ w.r.t. μ (c) $\mathcal{C}$ w.r.t. μ (d) $\mathcal{L}_W$ w.r.t. μ

(e) $\mathcal{E}$ w.r.t. ν (f) $\mathcal{U}$ w.r.t. ν (g) $\mathcal{C}$ w.r.t. ν (h) $\mathcal{L}_W$ w.r.t. ν

Figure 5: The performance metrics $(\mathcal{E}, \mathcal{U}, \mathcal{C})$ for various VQ approaches. For panels (a) to (d), the codebook distribution is initialized as a Gaussian distribution, while for panels (e) to (h), the codebook distribution is initialized as a uniform distribution.

4 An Atomic Setting for Evaluating the Criterion Triple

As discussed in Section 2.2, the variance of the latent distribution has a substantial impact on $\mathcal{E}$. Therefore, a fair evaluation of the criterion triple should be conducted under identical latent distributions. However, in practical scenarios, either in VQ-VAE [49] or VQGAN [10], the encoder outputs exhibit significantly different distributions across quantization methods due to intrinsic differences in their algorithmic mechanisms. Moreover, these distributional discrepancies are not static but evolve dynamically throughout VQ training. As a result, directly comparing quantization errors fails to accurately reflect the intrinsic effectiveness of different VQ algorithms and may even lead to misleading conclusions. To address this, we introduce a controlled simulated experimental setting that enables a principled evaluation of the intrinsic behavior of VQ algorithms.

Specifically, we assume that the encoder output feature distributions of all VQ methods follow the same Gaussian distribution, i.e., $\boldsymbol{z}_i \sim \mathcal{N}_d(\mu \cdot \mathbf{1}, \boldsymbol{I}_d)$. Although real-world feature distributions are often more complex, this simplification isolates the effect of the quantization mechanism itself and allows for controlled and comparable comparisons across methods. For the codebook distribution, we initialize all VQ methods with the same standard Gaussian distribution (by sampling $\boldsymbol{e}_k \sim \mathcal{N}_d(\mathbf{0}, \boldsymbol{I}_d)$), ensuring that the distribution variance is identical across all methods.

Our baselines include Vanilla VQ [49], EMA VQ [39], Online VQ [59], and Linear VQ, which employs a linear projection layer with frozen code vectors [60, 61]. For all VQ algorithms except Linear VQ, the sampled code vectors are treated as trainable parameters and optimized according to their respective update rules. In the case of Linear VQ, we focus on training only the linear projection layer. Detailed experimental specifications are provided in Appendix J.4.

As visualized in Figures 5a to 5c, *Wasserstein VQ* outperforms all baselines in terms of the criterion triplet $(\mathcal{E}, \mathcal{U}, \mathcal{C})$, particularly when the feature distribution and the initialized codebook distribution exhibit large deviations. While existing VQ methods perform well under the idealized setting of $\mu = 0$, this scenario is unrealistic: in practice, feature distributions are diverse and continuously evolving, making it infeasible to perfectly match the codebook distribution by codebook initialization. When a large initial distribution gap exists (e.g., $\mu = 5$), existing methods fail to achieve effective distribution alignment and perform poorly, as depicted in Figure 5d, highlighting their strong dependence on codebook initialization. In contrast, *Wasserstein VQ* overcomes this limitation through explicit distributional matching regularization, thereby achieving superior performance across all criterion metrics.

We observe the same behavior when the feature and codebook distributions are uniform, with feature vectors sampled from $\text{Unif}_d(\nu - 1, \nu + 1)$ and code vectors initialized from $\text{Unif}_d(-1, 1)$. As shown in Figures 5e to 5h, *Wasserstein VQ* again performs the best, suggesting that its effectiveness extends beyond Gaussian assumptions and exhibits distribution-agnostic behavior.

Table 1: Comparison of VQ-VAEs trained on FFHQ dataset following [49].

Approaches	Tokens	Codebook Size	$\mathcal{U}$ (↑)	$\mathcal{C}$ (↑)	PSNR(↑)	SSIM(↑)	Rec. Loss(↓)
Vanilla VQ	256	16384	3.8%	527.2	27.83	73.8	0.0119
STE++	256	16384	3.4%	476.7	27.54	72.3	0.0129
EMA VQ	256	16384	14.0%	1795.7	28.39	74.8	0.0106
Online VQ	256	16384	11.7%	1115.3	27.68	72.6	0.0125
Wasserstein VQ	256	16384	**100%**	**15713.3**	**29.03**	**76.6**	**0.0093**
Vanilla VQ	256	50000	1.2%	516.8	27.83	73.6	0.0120
STE++	256	50000	1.0%	447.2	27.49	72.4	0.0131
EMA VQ	256	50000	10.3%	4075.7	28.61	75.3	0.0101
Online VQ	256	50000	6.0%	1642.9	28.37	74.6	0.0107
Wasserstein VQ	256	50000	**100%**	**47496.4**	**29.24**	**77.0**	**0.0089**
Vanilla VQ	256	100000	0.6%	481.0	27.86	74.2	0.0118
STE++	256	100000	0.5%	450.7	27.52	72.4	0.0130
EMA VQ	256	100000	2.7%	2087.5	28.43	74.8	0.0105
Online VQ	256	100000	3.6%	1556.8	27.12	71.1	0.0142
Wasserstein VQ	256	100000	**100%**	**93152.7**	**29.53**	**78.0**	**0.0083**

Table 2: Comparison of VQ-VAEs trained on ImageNet dataset following [49].

Methods	**Tokens**	**Codebook Size**	$\mathcal{U}$ (↑)	$\mathcal{C}$ (↑)	**PSNR** (↑)	**SSIM** (↑)	**Rec. Loss** (↓)
Vanilla VQ	256	16,384	2.5%	360.7	24.44	57.5	0.0294
STE++	256	16,384	6.5%	889.9	24.88	58.9	0.0270
EMA VQ	256	16,384	14.5%	1861.5	24.98	59.2	0.0267
Online VQ	256	16,384	22.2%	1465.6	24.88	58.6	0.0273
Wasserstein VQ	256	16,384	**100.0%**	**15539.1**	**25.47**	**61.2**	**0.0242**
Vanilla VQ	256	50,000	0.9%	378.7	24.40	57.7	0.0295
STE++	256	50,000	2.0%	851.7	24.89	59.0	0.0270
EMA VQ	256	50,000	16.8%	6139.3	25.37	60.9	0.0246
Online VQ	256	50,000	9.9%	2241.7	25.09	59.7	0.0260
Wasserstein VQ	256	50,000	**100.0%**	**46133.2**	**25.72**	**62.3**	**0.0230**
Vanilla VQ	256	100,000	0.4%	337.0	24.43	57.4	0.0295
STE++	256	100,000	0.9%	730.6	24.86	59.1	0.0269
EMA VQ	256	100,000	3.0%	2170.0	25.13	60.1	0.0257
Online VQ	256	100,000	4.1%	1709.9	24.95	59.1	0.0267
Wasserstein VQ	256	100,000	**100.0%**	**93264.7**	**25.88**	**63.0**	**0.0223**

5 Experiments

In this section, we empirically demonstrate the effectiveness of the proposed distributional matching framework in visual tokenization tasks. Experiments are conducted within the VQ-VAE [49] and VQGAN [10] frameworks.

5.1 Evaluation on VQ-VAE Framework

Datasets and Baselines Experiments are conducted on four benchmark datasets: two low-resolution datasets, i.e., CIFAR-10 [27] and SVHN [34], and two high-resolution datasets FFHQ [23] and ImageNet [8]. We evaluate our approach against representative VQ methods: Vanilla VQ [49], EMA VQ [39], which uses exponential moving average updates and is also referred to as k-means, Online VQ, which employs k-means++ in CVQ-VAE [59], and enhanced straight-through estimator STE++ [20]. For experimental settings, see Appendix J.5.

Metrics We employ multiple evaluation metrics, including the codebook utilization rate ($\mathcal{U}$), codebook perplexity ($\mathcal{C}$), peak signal-to-noise ratio (PSNR), patch-level structural similarity index (SSIM), and pixel-level reconstruction loss (Rec. Loss). We exclude the raw quantization error ($\mathcal{E}$) from the main VQ-VAE experiments, as $\mathcal{E}$ is highly sensitive to feature variance, which is not controlled under different training dynamics. Consequently, a direct comparison of $\mathcal{E}$ does not faithfully reflect the intrinsic effectiveness of VQ algorithms, as discussed in Section 2.2 and Section 4.

Main Results As shown in Tables 1, 2 in the main text, and Tables 11, 12 in Appendix K.1, our proposed *Wasserstein VQ* consistently outperforms all baselines across datasets, achieving superior performance on nearly all metrics under a wide range of settings. The improvements are particularly pronounced in codebook utilization ($\mathcal{U}$) and codebook perplexity ($\mathcal{C}$), where *Wasserstein VQ* attains

Table 3: Analysis of generalization ability from ImageNet to FFHQ dataset.

Approaches	Tokens	Codebook Dim	$\mathcal{U}$ (↑)	$\mathcal{C}$ (↑)	PSNR(↑)	SSIM(↑)	Rec. Loss (↓)
Vanilla VQ	256	16384	1.7%	332.9	27.18	71.6	0.0138
EMA VQ	256	16384	12.5%	1292.1	27.69	72.3	0.0126
Online VQ	256	16384	33.2%	2794.9	27.91	72.7	0.0120
Wasserstein VQ	256	16384	99.2%	13975.4	28.62	75.0	0.0103

Table 4: The computational overhead of various VQ approaches.

Approaches	Codebook Size	Times (second)	Codebook Size	Times (second)	Codebook Size	Times (second)
Vanilla VQ	16384	**0.184**	50000	**0.408**	100000	**0.655**
EMA VQ	16384	0.265	50000	0.807	100000	1.502
Online VQ	16384	1.810	50000	5.438	100000	10.64
Wasserstein VQ	16384	0.294	50000	0.525	100000	0.774

near-complete codebook utilization and substantially higher codebook perplexity. This indicates that the learned codebooks more faithfully reflect the structure and density of the feature space. As vector quantization fundamentally acts as a compression mechanism from continuous latent representations to a discrete codebook, improved alignment between feature and code distributions directly translates into reduced information loss. Consistent with this interpretation, *Wasserstein VQ* achieves the lowest reconstruction loss and improved PSNR/SSIM across datasets.

Gaussian Approximation Validation To empirically assess the Gaussian approximation underlying our Wasserstein objective, we analyze the distributional properties of the learned latent features. Using the FFHQ dataset, we extract latent vectors from a subset of 1,600 images ($d = 8$), resulting in a total of $N = 409{,}600$ samples. Figure 6 presents Quantile–Quantile (Q–Q) plots for the feature dimensions exhibiting the highest (z_6) and lowest (z_7) agreement with a Gaussian reference. Under an ideal Gaussian model, the samples (blue points) align with the theoretical reference line ($y = x$). As shown, the best-fitting dimension (z_6) demonstrates near-perfect alignment, with a coefficient of determination of $R^2 = 0.9988$. Importantly, even for the least-fitting dimension (z_7), the Q–Q plot evaluated over the distribution achieves a high goodness-of-fit score ($R^2 > 0.979$). While mild deviations appear in the extreme tails, the central mass of the distribution, which dominates the probability mass and governs the Wasserstein objective, is well approximated by a Gaussian. These results support the use of a Gaussian-based Wasserstein formulation as an effective approximation for distributional matching. A more comprehensive univariate and multivariate analysis, including Mahalanobis distance based normality assessment across all dimensions, is provided in Appendix I.

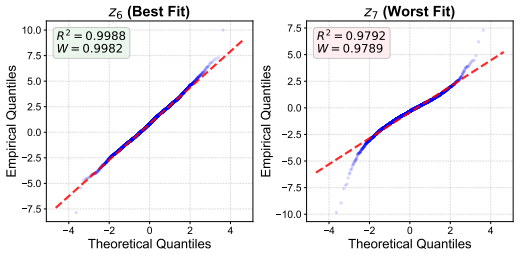
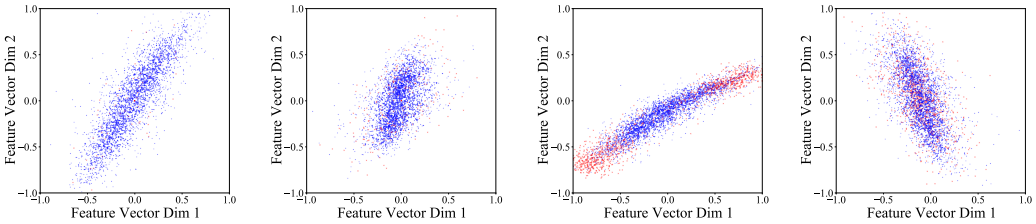


Figure 6: **Gaussianity Validation via Q-Q Plots.** We visualize the feature dimensions with the highest z_6 (Left) and lowest (z_7, Right) conformance to the Gaussian hypothesis. **Note:** For visual clarity, plots display a random subset of 5,000 points, while the reported statistics (R^2, W) are computed on the **full validation set** ($N = 409,600$). The strong alignment with the red identity line ($y = x$) confirms that the learned features are predominantly Gaussian, with deviations confined to sparse heavy tails.

Representation Visualization To examine the learned distribution of feature and code vectors across different VQ methods trained on the FFHQ dataset (with a fixed codebook size of $K = 8192$), we randomly sample 3,000 feature vectors and 1,000 code vectors and visualize them via scatter plots. As shown in Figures 7a and 7b, in both Vanilla VQ and EMA VQ, most code vectors are concentrated near the origin, rendering them largely inactive. While Online VQ mitigates this central clustering, its code vectors are pushed toward the extremes of the feature space, as illustrated in Figure 7c. This distributional mismatch leads to increased information loss and reduced codebook utilization. By contrast, *Wasserstein VQ* achieves substantially better alignment between feature and code vector distributions, significantly reducing information loss and improving codebook utilization.

Cross-Dataset Generalization To evaluate the model's generalization capability, we conducted a cross-dataset transfer experiment. Specifically, we used the pre-trained checkpoint from Table 2 (trained on ImageNet) and evaluated its performance on the FFHQ dataset, which is out-of-distribution relative to ImageNet. The results are summarized in Table 3.



(a) Vanilla VQ (b) EMA VQ (c) Online VQ (d) Wasserstein VQ

Figure 7: Visualization of feature and codebook distributions. Blue $\cdot$ and red $\times$ represent the feature and code vectors, respectively.

Among all methods, Wasserstein VQ achieves the highest performance in this transfer setting, indicating that the proposed distribution alignment objective does not impair generalization. Moreover, Wasserstein VQ's strong performance on the out-of-distribution FFHQ dataset suggests that enforcing alignment may contribute to improved robustness under distribution shifts. Overall, these findings provide evidence that models trained with the proposed regularization generalize effectively to out-of-distribution data.

Computational Overhead Analyses We evaluate the runtime efficiency by measuring the forward and backward pass times of the VQ module. As shown in Table 4 in Appendix K.4, the runtime of Wasserstein VQ is comparable to Vanilla VQ, with only negligible overhead arising from the covariance computation. This confirms that the closed-form Wasserstein objective offers a highly efficient pathway to optimal codebook utilization without incuring a significant time overhead. More implementation details see Appendix K.4.

Sensitivity Analysis on γ We conduct a sensitivity analysis with respect to γ on the FFHQ dataset by varying $\gamma \in \{0, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$. As reported in Table 15 in Appendix K.3, when $\gamma = 0$, Wasserstein VQ yields the worst performance, as it degenerates into the vanilla VQ formulation without distributional matching. As γ increases from 10^{-5} to 10^{-3}, the performance of Wasserstein VQ consistently improves. When γ reaches 10^{-2}, Wasserstein VQ achieves full (100%) codebook utilization along with competitive quantitative results. Moreover, within the range $\gamma \in [10^{-2}, 1]$, the performance of Wasserstein VQ remains stable, indicating that the method is not sensitive to the precise choice of γ once it exceeds a moderate threshold.

Effect of Codebook Size on FFHQ To investigate the impact of codebook size on VQ performance, we vary the codebook size K across a wide range: $K \in \{1024, 2048, 4096, 8192, 16384, 50000, 100000\}$. As shown in Table 1 in the main text and Table 13 in Appendix K.2, the vanilla VQ model suffers from severe codebook collapse even at relatively small sizes, such as $K = 1024$. In contrast, improved variants like EMA VQ and Online VQ handle smaller codebooks effectively, but still exhibit codebook collapse when K becomes very large (e.g., $K \geq 50000$). In contrast, *Wasserstein VQ* consistently maintains 100% codebook utilization across all codebook sizes, highlighting the effectiveness of enforcing distributional alignment via the quadratic Wasserstein distance in mitigating codebook collapse.

Effect of Codebook Dimensionality We investigate the impact of codebook dimensionality on VQ performance by conducting experiments on CIFAR-10 with d ranging from 2 to 32. As reported in Table 14 (Appendix K.2), our proposed Wasserstein VQ consistently outperforms all baselines across all dimensionalities. We also observe a manifestation of the curse of dimensionality: performance generally degrades as d increases. Vanilla VQ experiences the most severe decline, followed by EMA VQ and Online VQ, whereas Wasserstein VQ exhibits only minimal reduction in codebook utilization.

5.2 Evaluation on the VQGAN Framework

Datasets and Baselines We evaluate our proposed method against a comprehensive set of baselines on the ImageNet and FFHQ datasets. For ImageNet, the compared methods include DQVAE [18], DF-VQGAN [35], DiVAE [42], RQVAE [28], VQGAN [10], VQGAN-FC [53], VQGAN-EMA [39], VQGAN-LC [60], Llama GEN [45], and VAR [47]. For FFHQ, we compare against RQVAE [28],

Table 5: Reconstruction performance on the ImageNet-1K dataset. The suffixes "–a," "–b," and "–c" correspond to decoder adaptation for 5, 10, and 15 epochs, respectively. The best-performing result is highlighted in bold. Results marked with † are cited from VQGAN-LC [60], those with ⋆ from Llama GEN [45], and those with ‡ from VQ-Transplant [12].

Methods	Tokens	Codebook Size	$\mathcal{U}$ (↑)	r-FID (↓)	r-IS (↑)	LPIPS (↓)	PSNR (↑)	SSIM (↑)
DQVAE†	256	1,024	-	4.08	-	-	-	-
DiVAE†	256	16,384	-	4.07	-	-	-	-
RQVAE†	256	16,384	-	3.20	-	-	-	-
RQVAE†	512	16,384	-	2.69	-	-	-	-
RQVAE†	1,024	16,384	-	1.83	-	-	-	-
DF-VQGAN†	256	12,288	-	5.16	-	-	-	-
DF-VQGAN†	1,024	8,192	-	1.38	-	-	-	-
Llama GEN⋆	256	16,384	97.0%	2.19	-	-	20.79	67.5
VQGAN†	256	16,384	3.4%	5.96	-	0.17	23.3	52.4
	256	50,000	1.1%	5.44	-	0.17	22.5	52.5
	256	100,000	0.5%	5.44	-	0.17	22.3	52.5
VQGAN-FC†	256	16,384	11.2%	4.29	-	0.17	22.8	54.5
	256	50,000	3.6%	4.96	-	0.15	23.1	54.7
	256	100,000	1.9%	4.65	-	0.15	22.9	55.1
VQGAN-EMA†	256	16,384	83.2%	3.41	-	0.14	23.5	56.6
	256	50,000	40.2%	3.88	-	0.14	23.2	55.9
	256	100,000	24.2%	3.46	-	0.13	23.4	56.2
VQGAN-LC†	256	16,384	**99.9%**	3.01	-	0.13	23.2	56.4
	256	50,000	**99.9%**	2.75	-	0.13	23.8	58.4
	256	100,000	**99.9%**	2.62	-	0.12	23.8	58.9
	1,024	100,000	99.5%	1.29	-	**0.07**	**27.0**	**71.6**
Wasserstein VQ-a‡	512	16,384	99.8%	1.04	191.3	0.114	24.36	64.0
	512	32,768	99.7%	0.98	193.9	0.111	24.37	64.3
	512	65,536	99.6%	0.92	195.5	0.106	24.68	65.4
Wasserstein VQ-b	512	16,384	99.8%	0.98	192.9	0.114	24.34	63.7
	512	32,768	99.8%	0.88	196.2	0.109	24.60	64.7
	512	65,536	99.6%	0.81	**198.7**	0.105	**24.77**	**65.5**
Wasserstein VQ-c	512	16,384	99.8%	0.90	194.1	0.114	24.27	63.4
	512	32,768	99.8%	0.85	196.4	0.109	24.43	64.1
	512	65,536	99.6%	**0.79**	198.5	**0.104**	24.73	65.2

Table 6: Reconstruction performance on the ImageNet-1K dataset for multi-scale quantization algorithms. The suffixes "–a," "–b," and "–c" correspond to decoder adaptation for 5, 10, and 15 epochs, respectively. For each codebook size, the best-performing result is highlighted in bold. ‡: results are cited from VQ-Transplant [12].

Methods	Tokens	Codebook Size	$\mathcal{E}$ (↓)	$\mathcal{U}$ (↑)	$\mathcal{C}$ (↑)	r-FID (↓)
VAR‡	680	4,096	0.283	**100%**	2,981.4	0.92
Vanilla VAR-a‡	680	4,096	0.305	38.2%	1,300.4	1.25
	680	8,192	0.309	22.9%	1,422.6	1.30
EMA VAR-a‡	680	4,096	0.321	99.9%	1,806.6	1.69
	680	8,192	0.312	99.8%	2,498.8	1.15
Online VAR-a‡	680	4,096	0.276	99.0%	1,950.9	1.05
	680	8,192	0.269	73.9%	3,588.6	1.00
Wasserstein VAR-a‡	680	4,096	**0.255**	**100%**	3,286.2	0.93
	680	8,192	**0.240**	**100%**	6,518.2	0.83
Wasserstein VAR-b	680	4,096	**0.255**	**100%**	3,286.2	0.88
	680	8,192	**0.240**	**100%**	6,518.2	0.79
Wasserstein VAR-c	680	4,096	**0.255**	**100%**	3,286.2	**0.81**
	680	8,192	**0.240**	**100%**	6,518.2	**0.73**

VQGAN [10], VQGAN-FC [53], VQGAN-EMA [39], VQ-WAE [50], MQVAE [19], and VQGAN-LC [60]. Implementation details see Appendix J.5.

Table 7: Reconstruction performance on the FFHQ dataset. Results marked with † are cited from VQGAN-LC [60], and those with ‡ from VQ-Transplant [12].

VQs	Tokens	Codebook Size K	$\mathcal{E}$(↓)	$\mathcal{U}$ (↑)	PSNR(↑)	SSIM(↑)	LPIPS (↓)	r-FID(↓)
RQVAE†	256	2048	-	-	22.9	67.0	0.13	7.04
VQ-WAE†	256	1024	-	-	22.5	66.5	0.12	4.20
MQVAE†	256	1024	-	78.2%	-	-	-	4.55
VQGAN†	256	16384	-	2.3%	24.4	63.3	0.12	5.25
VQGAN-FC†	256	16384	-	10.9%	24.8	64.6	0.11	4.86
VQGAN-EMA†	256	16384	-	68.2%	25.4	66.1	0.10	4.79
VQGAN-LC†	256	100000	-	99.5%	26.1	69.4	0.08	3.81
Wasserstein VQ‡	512	16384	**0.153**	**99.7%**	**27.25**	**75.4**	**0.075**	**1.81**
Wasserstein VQ‡	512	32768	**0.142**	**99.7%**	**27.33**	**75.7**	**0.072**	**1.21**

Figure 8: Visualization of reconstructed ImageNet Images. The top row displays the original input images with a resolution of 256×256 pixels, while the bottom row shows the reconstructed images from the *Wasserstein VAR*.

Figure 9: Visualization of reconstructed FFHQ Images. The top row displays the original input images with a resolution of 256×256 pixels, while the bottom row shows the reconstructed images from the *Wasserstein VQ*.

Metrics We report standard reconstruction and perceptual quality metrics, including Fréchet Inception Distance (r-FID) [16], Reconstruction Inception Score (r-IS) [41], Learned Perceptual Image Patch Similarity (LPIPS) [57], Peak Signal-to-Noise Ratio (PSNR), and Structural Similarity Index Measure (SSIM).

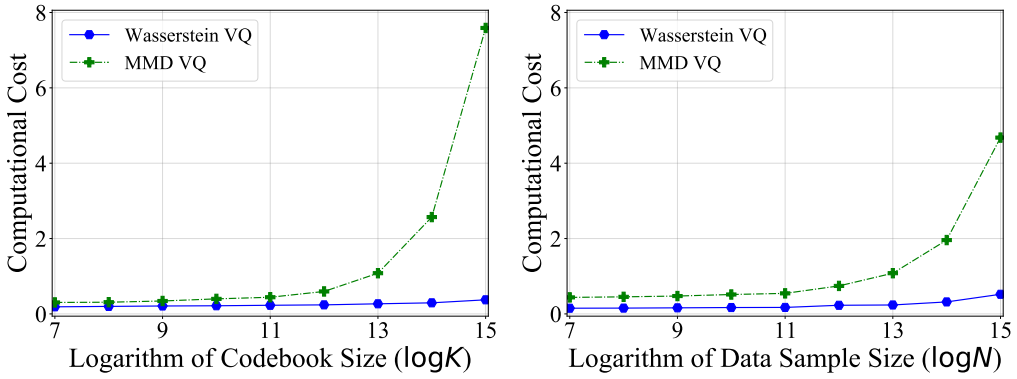
Main Results As shown in Tables 5 and 7, our proposed *Wasserstein VQ* achieves the lowest r-FID within the VQGAN framework. We further evaluate multi-scale quantization algorithms in Table 6. All experiments in this table are conducted under fully controlled conditions: the encoder–decoder architecture is fixed to VAR [47], and, importantly, the latent features are kept identical across all methods. This ensures that measurements of codebook utilization, perplexity, and quantization error are not influenced by variations in the latent features, as discussed in Section 2.2.

Under these controlled conditions, *Wasserstein VAR* consistently demonstrates the strongest overall performance, with particularly notable improvements in quantization error. Reconstruction quality further improves as the number of decoder adaptation epochs increases, reflecting stronger adversarial refinement. For instance, when $K = 8192$, the r-FID decreases from **0.83** (Wasserstein VAR-a) to **0.73** (Wasserstein VAR-c).

Reconstruction Visualization To further demonstrate the reconstruction performance, we visualize the reconstructed FFHQ and ImageNet images in Figure 9 and 8. As shown, the images reconstructed by our method are nearly identical to the original inputs. In particular, on the FFHQ dataset, only very minor differences can be observed in fine details.

Table 8: Reconstruction performance on the ImageNet-1K dataset for multi-scale quantization algorithms. The suffixes "–a," "–b," and "–c" correspond to decoder adaptation for 5, 10, and 15 epochs, respectively. For each codebook size, the best-performing result is highlighted in bold.

Methods	Tokens	Codebook Size K	$\mathcal{E}$ (↓)	$\mathcal{U}$ (↑)	$\mathcal{C}$ (↑)	r-FID (↓)
MMD VAR-a	680	4096	**0.255**	**100%**	**3757.3**	**0.91**
	680	8192	**0.234**	**100%**	**7539.4**	**0.81**
Wasserstein VAR-a	680	4096	**0.255**	**100%**	3286.2	0.93
	680	8192	0.240	**100%**	6518.2	0.83
MMD VAR-b	680	4096	**0.255**	**100%**	**3757.3**	**0.87**
	680	8192	**0.234**	**100%**	**7539.4**	**0.78**
Wasserstein VAR-b	680	4096	**0.255**	**100%**	3286.2	0.88
	680	8192	0.240	**100%**	6518.2	0.79
MMD VAR-c	680	4096	**0.255**	**100%**	**3757.3**	0.82
	680	8192	**0.234**	**100%**	**7539.4**	0.75
Wasserstein VAR-c	680	4096	**0.255**	**100%**	3286.2	**0.81**
	680	8192	0.240	**100%**	6518.2	**0.73**



(a) Codebook Size (b) Data Sample Size

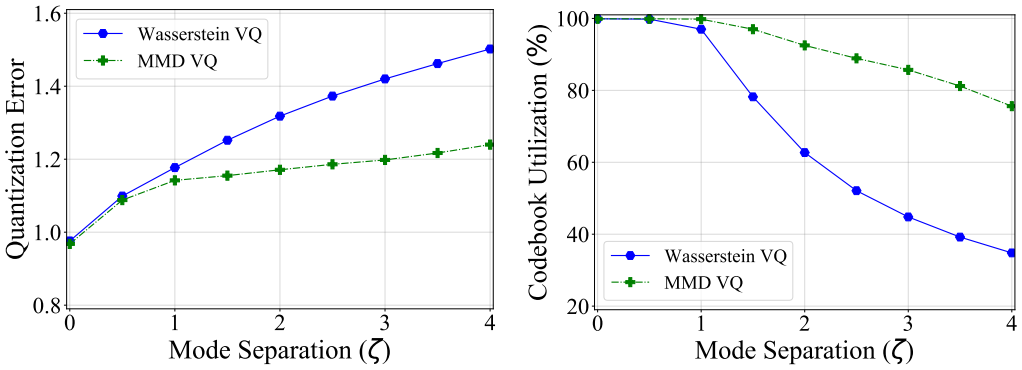
Figure 10: Computational overhead (seconds) comparison between Wasserstein VQ and MMD VQ.

6 Discussion on Two Distributional Matching Approaches: Wasserstein VQ vs. MMD VQ

We compare Wasserstein VQ and MMD VQ from three perspectives: visual tokenization performance, computational efficiency, and robustness to non-Gaussian distributions. First, we assess visual tokenization performance under the VQGAN framework. As shown in Table 8, we evaluate performance with decoder adaptation conducted for 5, 10, and 15 epochs. Overall, the two methods achieve very similar performance, with Wasserstein VQ slightly outperforming MMD VQ when the decoder adaptation is extended to 15 epochs.

Second, we analyze the computational cost of the two approaches. Following the setup in Appendix K.4, we measure runtime efficiency by recording forward and backward pass times over 100 iterations. We consider two scenarios: (i) fixing the number of data samples at $N = 8192$ while varying the codebook size K from 128 to 32,768, and (ii) fixing the codebook size at $K = 8192$ while varying the number of data samples N from 128 to 32,768, providing complementary comparisons. As illustrated in Figures 10a and 10b, MMD VQ exhibits a rapidly increasing computational cost, reflecting its inefficiency. This difference arises because Wasserstein VQ performs distribution matching by aligning first- and second-order statistics, whereas MMD VQ relies on pairwise distances between all elements. Consequently, the computational complexity of MMD VQ grows substantially, especially when both the number of data samples and the codebook size are large.

Third, we evaluate the robustness of Wasserstein VQ and MMD VQ under non-Gaussian latent distributions. While Section 5.1 shows that the latent distributions in visual tokenization tasks are approximately Gaussian, this experiment allows us to explicitly assess robustness beyond the



(a) Quantization error

(b) Codebook utilization

Figure 11: comparison between Wasserstein VQ and MMD VQ.

Gaussian regime. To simulate a non-Gaussian distribution, we follow the setting in Section 4 and assume the encoder output features follow a bimodal mixture:

$$z_i \sim 0.5 \cdot \mathcal{N}(\zeta \mathbf{1}, I) + 0.5 \cdot \mathcal{N}(-\zeta \mathbf{1}, I),$$

where the mode separation parameter $\zeta \in \{0.0, 0.5, \ldots, 4.0\}$ controls deviation from Gaussianity, with larger ζ indicating stronger non-Gaussianity. The codebook is initialized with a standard Gaussian distribution, and code vectors are treated as trainable parameters. After 10,000 training steps, we evaluate Wasserstein VQ and MMD VQ in terms of quantization error and codebook utilization rate. As shown in Figures 11a and 11b, for small ζ (nearly Gaussian), the two methods perform comparably. As ζ increases, MMD VQ consistently achieves lower quantization error and higher codebook utilization, whereas Wasserstein VQ degrades due to its limited moment-matching assumption. These synthetic experiments empirically demonstrate that MMD VQ is more robust to non-Gaussian feature distributions.

Summary. Wasserstein VQ and MMD VQ exhibit complementary strengths and limitations. Wasserstein VQ is computationally efficient and performs stably under nearly Gaussian latent distributions, benefiting from its first- and second-order moment-matching formulation, which scales well with large datasets and codebooks. However, it is less robust when feature distributions deviate from Gaussianity. In contrast, MMD VQ excels under non-Gaussian distributions, consistently achieving lower quantization error and higher codebook utilization, but at the cost of substantially higher computational overhead. In practice, Wasserstein VQ is preferred for efficiency and standard Gaussian-like scenarios, while MMD VQ provides an advantage when robustness to non-Gaussian features is critical.

7 Conclusion

We identify a fundamental distributional mismatch between feature and code vectors as a common cause of training instability and codebook collapse in vector quantization. To address this issue, we propose a general distributional matching framework that formalizes desirable VQ behavior through principled criteria. Within this framework, we show that aligning feature and code distributions provides a unified mechanism for improving training stability, codebook utilization, and representation efficiency. We instantiate the framework using a Wasserstein-based objective with an efficient closed-form solution under a mild Gaussian approximation, and further demonstrate that a nonparametric alternative based on maximum mean discrepancy achieves comparable performance. More broadly, this work offers a unified perspective on vector quantization and opens new avenues for designing stable and efficient discrete representation learning methods.

8 Limitation and Discussion

One limitation of this work is that we do not directly validate the effectiveness of the proposed distributional matching framework through downstream generative results, due to limited computational resources. Recent studies have shown that improved reconstruction performance does not necessarily translate into better generative performance. For instance, increasing the codebook size

or token length in discrete visual tokenizers [54], or increasing the latent resolution or dimensionality in continuous tokenizers, can significantly improve reconstruction quality while simultaneously degrading generative performance [52].

In these cases, larger codebooks or longer token sequences in discrete tokenizers substantially increase the complexity of training autoregressive models, due to longer sequences and larger vocabularies. Similarly, higher latent resolution or dimensionality in continuous tokenizers makes the training of diffusion models more challenging, leading to reduced generative quality.

In contrast, distributional matching improves tokenizer quality without introducing additional burdens to downstream generative models. For example, compared with VAR [47], Wasserstein VAR achieves improved reconstruction without increasing the codebook size or token length. As a result, the improved reconstruction performance induced by distributional matching is more likely to translate into improved generative performance.

References

[1] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *ACM-SIAM Symposium on Discrete Algorithms*, 2007.

[2] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *ArXiv*, 2013.

[3] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *ArXiv*, 2013.

[4] A. Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distributions. *Bulletin of the Calcutta Mathematical Society*, 1943.

[5] Paul S. Bradley and Usama M. Fayyad. Refining initial points for k-means clustering. In *ICML*, 1998.

[6] Mathilde Caron, Hugo Touvron, Ishan Misra, Herv'e J'egou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.

[7] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. Maskgit: Masked generative image transformer. In *CVPR*, 2022.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, K. Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[9] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music. *ArXiv*, 2020.

[10] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021.

[11] Xianghong Fang, Wenlong Mou, Yuan Yuan, Dehan Kong, and Tim G. J. Rudner. A unifying view of vector, product, and scalar quantization: An information-theoretic perspective. In *ICLR submission*, 2026.

[12] Xianghong Fang, Yuan Yuan, Dehan Kong, and Tim G. J. Rudner. VQ-transplant: Efficient plug-and-play vq-module integration for pre-trained visual tokenizers. In *ICLR*, 2026.

[13] Siegfried Graf and Harald Luschgy. *Foundations of quantization for probability distributions*. Springer Science & Business Media, 2000.

[14] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Scholkopf, and Alex Smola. A kernel two-sample test. *JMLR*, 2012.

[15] Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. Vector quantized diffusion model for text-to-image synthesis. In *CVPR*, 2022.

[16] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, 2017.

[17] Jonathan Ho, Ajay Jain, and P. Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

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

[19] Mengqi Huang, Zhendong Mao, Quang Wang, and Yongdong Zhang. Not all image regions matter: Masked vector quantization for autoregressive image generation. In *CVPR*, 2023.

[20] Minyoung Huh, Brian Cheung, Pulkit Agrawal, and Phillip Isola. Straightening out the straight-through estimator: Overcoming optimization challenges in vector quantized networks. In *ICML*, 2023.

[21] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[22] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016.

[23] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.

[24] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.

[25] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *CVPR*, 2020.

[26] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[27] Alex Krizhevsky. Learning multiple layers of features from tiny images. *ArXiv*, 2009.

[28] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *CVPR*, 2022.

[29] Xiang Li, Kai Qiu, Hao Chen, Jason Kuen, Jiuxiang Gu, Jindong Wang, Zhe Lin, and Bhiksha Raj. Xq-gan: An open-source image tokenization framework for autoregressive generation. *ArXiv*, 2024.

[30] Jae Hyun Lim and J. C. Ye. Geometric gan. *ArXiv*, 2017.

[31] David Lindley and Solomon Kullback. Information theory and statistics. *Journal of the American Statistical Association*, 1959.

[32] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

[33] Fabian Mentzer, David C. Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: Vq-vae made simple. In *ICLR*, 2024.

[34] Yuval Netzer, Tao Wang, Adam Coates, A. Bissacco, Bo Wu, and A. Ng. Reading digits in natural images with unsupervised feature learning. *ArXiv*, 2011.

[35] Minheng Ni, Chenfei Wu, Haoyang Huang, Daxin Jiang, Wangmeng Zuo, and Nan Duan. Nüwa-lip: Language-guided image inpainting with defect-free vqgan. In *CVPR*, 2022.

[36] Ingram Olkin and Friedrich Pukelsheim. The distance between two random vectors with given dispersion matrices. *Linear Algebra and its Applications*, 1982.

[37] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Q. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russ Howes, Po-Yao (Bernie) Huang, Shang-Wen Li, Ishan Misra, Michael G. Rabbat, Vasu Sharma, Gabriel Synnaeve, Huijiao Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. *TMLR*, 2024.

[38] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *ICML*, 2021.

[39] Ali Razavi, Aäron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. In *NeurIPS*, 2019.

[40] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

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

[42] Jie Shi, Chenfei Wu, Jian Liang, Xiang Liu, and Nan Duan. Divae: Photorealistic images synthesis with denoising diffusion decoder. *ArXiv*, 2022.

[43] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[44] Bharath K. Sriperumbudur, Arthur Gretton, Kenji Fukumizu, Bernhard Scholkopf, and Gert R. G. Lanckriet. Hilbert space embeddings and metrics on probability measures. *JMLR*, 2009.

[45] 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*, 2024.

[46] Yuhta Takida, Takashi Shibuya, Wei-Hsiang 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. In *ICML*, 2022.

[47] Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. In *NeurIPS*, 2024.

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

[49] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *NeurIPS*, 2017.

[50] Tung-Long Vuong, Trung-Nghia Le, He Zhao, Chuanxia Zheng, Mehrtash Harandi, Jianfei Cai, and Dinh Q. Phung. Vector quantized wasserstein auto-encoder. In *ICML*, 2023.

[51] Will Williams, Sam Ringer, Tom Ash, John Hughes, David Macleod, and Jamie Dougherty. Hierarchical quantized autoencoders. In *NeurIPS*, 2020.

[52] Jingfeng Yao and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In *CVPR*, 2025.

[53] Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. Vector-quantized image modeling with improved vqgan. In *ICLR*, 2022.

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

[55] Han Zhang, Zizhao Zhang, Augustus Odena, and Honglak Lee. Consistency regularization for generative adversarial networks. *ArXiv*, 2019.

[56] Jiahui Zhang, Fangneng Zhan, Christian Theobalt, and Shijian Lu. Regularized vector quantization for tokenized image synthesis. In *CVPR*, 2023.

[57] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[58] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient gan training. In *NeurIPS*, 2020.

[59] Chuanxia Zheng and Andrea Vedaldi. Online clustered codebook. In *ICCV*, 2023.

[60] Lei Zhu, Fangyun Wei, Yanye Lu, and Dong Chen. Scaling the codebook size of vqgan to 100,000 with a utilization rate of 99%. *ArXiv*, 2024.

[61] Yongxin Zhu, Bocheng Li, Yifei Xin, and Linli Xu. Addressing representation collapse in vector quantized models with one linear layer. *ArXiv*, 2024.

A Optimal Support of The Codebook Distribution

Proof of Theorem 1. First, we assume $\overline{\text{supp}(\mathcal{P}_B)} = \overline{\text{supp}(\mathcal{P}_A)}$. Then for any $\boldsymbol{z} \in \text{supp}(\mathcal{P}_A)$, there exist a sequence of points in $\text{supp}(\mathcal{P}_B)$ that converge to $\boldsymbol{z}$. Let $\{\boldsymbol{e}_k\}_{k=1}^K$ be K code vectors independently generated from $\mathcal{P}_B$. Then the empirical distribution of $\{\boldsymbol{e}_k\}_{k=1}^K$ tends to $\mathcal{P}_B$ as the size K tends to infinity. Since $\Omega = \text{supp}(\mathcal{P}_A)$ is a bounded region, we have the following:

$$\sup_{\boldsymbol{z} \in \overline{\text{supp}(\mathcal{P}_A)}} \min_k \|\boldsymbol{z} - \boldsymbol{e}_k\|^2 = \sup_{\boldsymbol{z} \in \overline{\text{supp}(\mathcal{P}_B)}} \min_k \|\boldsymbol{z} - \boldsymbol{e}_k\|^2 \xrightarrow{p} 0, \quad \text{as } K \to \infty. \tag{18}$$

This quantity is an upper bound on the quantization error $\mathcal{E}(\{\boldsymbol{z}_i\}; \{\boldsymbol{e}_k\})$. Thus,

$$\sup_{\{\boldsymbol{z}_i\} \subseteq \Omega} \mathcal{E}\left(\{\boldsymbol{z}_i\}_{i=1}^N; \{\boldsymbol{e}_k\}_{k=1}^K\right) \leq \sup_{\boldsymbol{z} \in \overline{\Omega}} \min_k \|\boldsymbol{z} - \boldsymbol{e}_k\|^2 \xrightarrow{p} 0, \quad \text{as } K \to \infty. \tag{19}$$

This demonstrates that $\mathcal{P}_B$ has vanishing quantization error asymptotically. Furthermore, for any K code vectors $\{\boldsymbol{e}_k\}_{k=1}^K$ independently drawn from $\mathcal{P}_B$, we have $\{\boldsymbol{e}_k\}_{k=1}^K \subseteq \overline{\Omega}$. Since the empirical distribution of $\{\boldsymbol{z}_i\}_{i=1}^N$ tends to $\mathcal{P}_A$ as the feature sample size N tends to infinity, we can easily show that for any fixed $\{\boldsymbol{e}_k\}_{k=1}^K \subseteq \overline{\Omega}$, the codebook utility rate satisfies

$$\mathcal{U}\left(\{\boldsymbol{z}_i\}_{i=1}^N, \{\boldsymbol{e}_k\}_{k=1}^K\right) \xrightarrow{p} 1, \quad \text{as } N \to \infty. \tag{20}$$

This shows that $\{\boldsymbol{e}_k\}_{k=1}^K$ attains full utilization asymptotically, and thus $\mathcal{P}_B$ attains full utilization asymptotically.

On the other hand, we assume $\mathcal{P}_B$ attains full utilization and vanishing quantization error asymptotically. Then we first claim that $\overline{\text{supp}(\mathcal{P}_A)} \subseteq \overline{\text{supp}(\mathcal{P}_B)}$. Since $\mathcal{P}_B$ has vanishing quantization error asymptotically, then for any $\boldsymbol{z} \in \text{supp}(\mathcal{P}_A)$, there exist a sequence of points in $\text{supp}(\mathcal{P}_B)$ that converge to $\boldsymbol{z}$. This implies that $\text{supp}(\mathcal{P}_A) \subseteq \overline{\text{supp}(\mathcal{P}_B)}$ and thus $\overline{\text{supp}(\mathcal{P}_A)} \subseteq \overline{\text{supp}(\mathcal{P}_B)}$.

To show $\overline{\text{supp}(\mathcal{P}_B)} = \overline{\text{supp}(\mathcal{P}_A)}$, it remains to show $\text{supp}(\mathcal{P}_B) \subseteq \overline{\text{supp}(\mathcal{P}_A)}$. In fact, if $\text{supp}(\mathcal{P}_B) \subseteq \overline{\text{supp}(\mathcal{P}_A)}$ does not hold, then there exists an open region $\mathcal{R} \subseteq \text{supp}(\mathcal{P}_B) - \overline{\text{supp}(\mathcal{P}_A)}$ such that $\mathcal{P}_B(\mathcal{R}) > 0$ and

$$\min_{\boldsymbol{z} \in \text{supp}(\mathcal{P}_A), \boldsymbol{z}' \in \mathcal{R}} \|\boldsymbol{z} - \boldsymbol{z}'\| \geq \epsilon_0 \tag{21}$$

for some $\epsilon_0 > 0$. Since $\text{supp}(\mathcal{P}_A) \subseteq \overline{\text{supp}(\mathcal{P}_B)}$, then there exists a sufficiently large K_0 such that the event

$$\left\{ \text{Generating} \{\boldsymbol{e}_k\}_{k=1}^{K_0} \text{ i.i.d. from } \mathcal{P}_B \text{ s.t. } \{\boldsymbol{e}_k\} \subseteq \text{supp}(\mathcal{P}_A), \sup_{\boldsymbol{z} \in \text{supp}(\mathcal{P}_A)} \min_k \|\boldsymbol{z} - \boldsymbol{e}_k\| < \epsilon_0 \right\} \tag{22}$$

has some positive probability $C > 0$. Then with a positive probability of at least $C \cdot \mathcal{P}_B(\mathcal{R})$, we can pick the first K_0 code vectors from Equation (22) and the (K_0+1)th code vector from $\mathcal{R}$. For any such codebook of size $K_0 + 1$, we know the (K_0+1)th code vector will never be used regardless of the choice of the feature set $\{\boldsymbol{z}_i\}$. Therefore, the codebook utilization

$$\sup_{\{\boldsymbol{z}_i\}} \mathcal{U}\left(\{\boldsymbol{e}_k\}_{k=1}^{K_0+1}; \{\boldsymbol{z}_i\}\right) \leq \frac{K_0}{K_0+1} < 1. \tag{23}$$

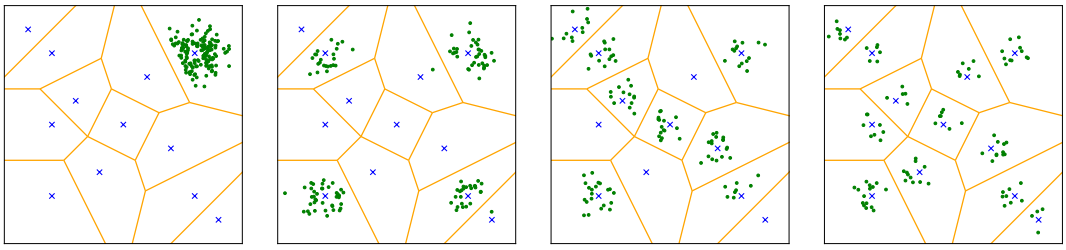
This contradicts the property that $\mathcal{P}_B$ attains full utilization asymptotically. Thus, $\text{supp}(\mathcal{P}_B) \subseteq \overline{\text{supp}(\mathcal{P}_A)}$ must hold. This concludes the proof. □

B Understanding Codebook Collapse Through the Lens of Voronoi Partition

B.1 Voronoi Partition: Definition and Connection to Codebook Collapse

Let $\mathcal{X}$ be a metric space with distance function $d(\cdot, \cdot)$, and let $\{\boldsymbol{e}_k\}_{k=1}^K$ denote a set of code vectors. The Voronoi cell (or Voronoi region) $\mathcal{R}_k$ associated with code vector $\boldsymbol{e}_k$ is the set of all points in $\mathcal{X}$ whose distance to $\boldsymbol{e}_k$ is no greater than their distance to any other code vector $\boldsymbol{e}_j$ ($j \neq k$):

$$\mathcal{R}_k = \{x \in \mathcal{X} \mid d(x, \boldsymbol{e}_k) \leq d(x, \boldsymbol{e}_j), \forall j \neq k\}. \tag{24}$$



(a) (b) (c) (d)

Figure 12: Visualization of the Voronoi partition. The symbols $\cdot$ and $\times$ represent the feature and code vectors, respectively.

The Voronoi diagram is the collection of all cells $\{\mathcal{R}_k\}_{k=1}^K$. Figure 12 illustrates a Voronoi diagram for 12 code vectors, partitioning the metric space into 12 regions according to $\mathcal{R}_k$. When d is the ℓ_2 distance, vector quantization (VQ) can equivalently be expressed in terms of Voronoi regions:

$$\forall \boldsymbol{z}_i \in \mathcal{R}_j, \quad \boldsymbol{z}_i' = \underset{\boldsymbol{e} \in \{\boldsymbol{e}_k\}}{\arg\min} \|\boldsymbol{z}_i - \boldsymbol{e}\| = \boldsymbol{e}_j, \tag{25}$$

where $\boldsymbol{z}_i$ is an arbitrary feature vector. Equation 25 provides an alternative viewpoint for nearest neighbor search: identify the region $\mathcal{R}_j$ containing $\boldsymbol{z}_i$ and select the corresponding code vector $\boldsymbol{e}_j$.

Relation to Codebook Collapse Codebook collapse occurs in its most severe form when all feature vectors fall within the same Voronoi region. As illustrated in Figure 12a, all features occupy a single region, leading to the utilization of only one code vector. To prevent collapse, feature vectors should be distributed across all regions as uniformly as possible (Figure 12d).

B.2 Limitations of Existing Vector Quantization Methods

We analyze why standard VQ methods often fail to prevent codebook collapse, using Vanilla VQ [49] and k-means-based VQ [39] as examples. Both share a similar assignment step but differ in their update mechanisms.

Assignment Step Given feature vectors $\{\boldsymbol{z}_i\}_{i=1}^N$ and code vectors $\{\boldsymbol{e}_k\}_{k=1}^K$, at iteration t, both methods partition the feature space into Voronoi cells and assign features to the nearest code vectors:

$$\mathcal{R}_k^{(t-1)} = \{x \in \mathcal{X} \mid \left\|x - \boldsymbol{e}_k^{(t-1)}\right\|_2^2 \leq \left\|x - \boldsymbol{e}_j^{(t-1)}\right\|_2^2, \forall j \neq k\}, \quad \mathcal{S}_k^{(t)} = \{\boldsymbol{z}_i \mid \boldsymbol{z}_i \in \mathcal{R}_k^{(t-1)}\}.$$

Update Step in Vanilla VQ Code vectors are updated via gradient descent on the reconstruction loss:

$$\mathcal{L} = \frac{1}{N} \sum_{k=1}^K \sum_{\boldsymbol{z}_m \in \mathcal{S}_k^{(t)}} \left\|\boldsymbol{z}_m - \boldsymbol{e}_k^{(t-1)}\right\|_2^2. \tag{26}$$

Update Step in k-means-based VQ Code vectors are updated using an exponential moving average:

$$\boldsymbol{e}_k^{(t)} = \alpha \boldsymbol{e}_k^{(t-1)} + (1-\alpha) \frac{1}{|\mathcal{S}_k^{(t)}|} \sum_{\boldsymbol{z}_m \in \mathcal{S}_k^{(t)}} \boldsymbol{z}_m. \tag{27}$$

Codebook Collapse in Vanilla and k-means VQ Despite different update strategies, both methods can suffer from codebook collapse because the assignment step does not guarantee that all Voronoi cells receive feature vectors (Figures 12a–12c). Larger codebooks exacerbate the issue, leaving some cells unassigned and their corresponding code vectors underutilized.

Connection to Distribution Matching and Mitigation As demonstrated in Section 4, both VQ methods are sensitive to codebook initialization. Collapse is mitigated only if the codebook distribution approximates the feature distribution. However, in practice, feature distributions are typically unknown and dynamically evolving. To address this, we introduce a **distributional matching constraint** that aligns the codebook distribution with the feature distribution, ensuring complete codebook utilization.

C The Relationship Between Gradient Gap and Quantization Error

In this section, we provide a formal derivation of the relationship between the **Quantization Error** (Criterion 1) and the **Gradient Gap** caused by the Straight-Through Estimator (STE). This analysis theoretically supports our claim that minimizing quantization error via distribution matching inherently mitigates training instability.

Problem Setup: Let $\mathcal{L}$ denote the global loss function. In the VQ process, the encoder outputs a continuous latent vector z_e, which is discretized to a code z_q. During backpropagation, the non-differentiable quantization operation is bypassed using the STE, which approximates the gradient of the encoder output $\frac{\partial \mathcal{L}}{\partial z_e}$ using the gradient of the quantized code $\frac{\partial \mathcal{L}}{\partial z_q}$:

$$\text{STE Approximation:} \quad \frac{\partial \mathcal{L}}{\partial z_e} \leftarrow \frac{\partial \mathcal{L}}{\partial z_q} \tag{28}$$

We define the **Gradient Gap** ($\mathcal{G}$) as the magnitude of the discrepancy between the true gradient and the STE approximated gradient:

$$\mathcal{G} := \left\| \frac{\partial \mathcal{L}}{\partial z_e} - \frac{\partial \mathcal{L}}{\partial z_q} \right\| \tag{29}$$

Taylor Expansion Analysis: To analyze this gap, we perform a Taylor expansion of the gradient function around the quantized point z_q. Let us consider the gradient function with respect to the latent variable x. Expanding $\frac{\partial \mathcal{L}}{\partial z_e}$ at the point $x = z_q$, we obtain:

$$\frac{\partial \mathcal{L}}{\partial z_e} = \frac{\partial \mathcal{L}}{\partial x}\bigg|_{x=z_q} + \frac{\partial^2 \mathcal{L}}{\partial x^2}\bigg|_{x=z_q} (z_e - z_q) + \mathcal{O}(\|z_e - z_q\|^2) \tag{30}$$

where $\frac{\partial \mathcal{L}}{\partial x}\big|_{x=z_q}$ is exactly the gradient backpropagated from the quantized code, i.e., $\frac{\partial \mathcal{L}}{\partial z_q}$, and $\mathbf{H} = \frac{\partial^2 \mathcal{L}}{\partial x^2}\Big|_{x=z_q}$ is the Hessian matrix, representing the local curvature of the loss function at z_q.

First-Order Approximation and Upper Bound: By retaining only the first-order term (as is common in gradient gap analysis) and ignoring the higher-order terms $\mathcal{O}(\|z_e - z_q\|^2)$, the relationship can be approximated as:

$$\frac{\partial \mathcal{L}}{\partial z_e} \approx \frac{\partial \mathcal{L}}{\partial z_q} + \mathbf{H}(z_e - z_q) \tag{31}$$

Substituting this back into the definition of the Gradient Gap $\mathcal{G}$, we get:

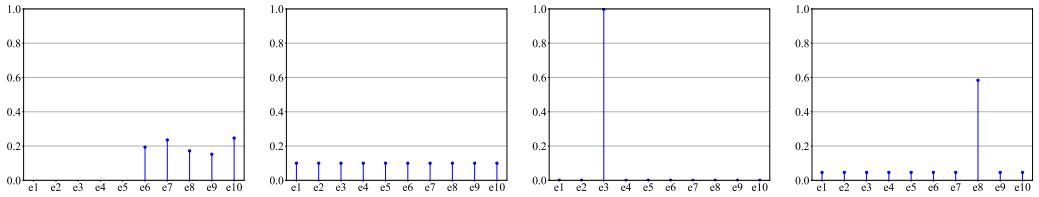
$$\mathcal{G} = \left\| \frac{\partial \mathcal{L}}{\partial z_e} - \frac{\partial \mathcal{L}}{\partial z_q} \right\| \approx \|\mathbf{H}(z_e - z_q)\| \tag{32}$$

By applying the definition of the consistent matrix norm (specifically, the spectral norm for the symmetric Hessian), we derive the upper bound:

$$\mathcal{G} \leq \|\mathbf{H}\|_2 \cdot \|z_e - z_q\| \tag{33}$$

where $\|\mathbf{H}\|_2$ represents the spectral norm of the Hessian (indicative of the maximum curvature) and $\|z_e - z_q\|$ corresponds to the **Quantization Error**.

The derivation in Eq. 33 explicitly demonstrates that the Gradient Gap $\mathcal{G}$ is linearly bounded by the Quantization Error. In methods like Vanilla VQ, a large Quantization Error implies a looser bound on $\mathcal{G}$, leading to inaccurate gradient signals passed to the encoder. This large discrepancy causes the "Training Instability" observed in previous works. By minimizing the Wasserstein distance, our method explicitly minimizes the Quantization Error (Criterion 1). This tightens the upper bound on $\mathcal{G}$, ensuring that the gradient passed to the encoder ($\frac{\partial \mathcal{L}}{\partial z_q}$) is a faithful approximation of the true gradient ($\frac{\partial \mathcal{L}}{\partial z_e}$).



(a) $(50\%, 4.92)$ (b) $(100\%, 10.00)$ (c) $(100\%, 1.02)$ (d) $(100\%, 4.92)$

Figure 13: Visualization of the evaluation criteria $(\mathcal{U}, \mathcal{C})$.

D Complementary Roles of Criterion 2 and 3 in Assessing Codebook Collapse

To clarify the complementary roles of Criterion 2 and Criterion 3, which are introduced in Section 2.2, we provide visual illustrations to facilitate intuitive understanding. The metric $\mathcal{U}$ is designed to quantify the *completeness* of codebook utilization. As shown in Figures 13a and 13b, the corresponding values of $\mathcal{U}$ are 50% and 100%, respectively[3].

However, $\mathcal{U}$ alone is insufficient to characterize the severity of codebook collapse, since it does not account for imbalance among utilized code vectors. This limitation is illustrated in Figure 13c. Although all code vectors are utilized, resulting in $\mathcal{U} = 100\%$, the code vector e_3 overwhelmingly dominates the assignment distribution. Such extreme imbalance constitutes a degenerate form of codebook collapse, even though $\mathcal{U}$ attains its maximum value. This observation motivates the introduction of Criterion 3, which explicitly measures the uniformity, or conversely the imbalance, of codebook utilization.

A comparison between Figures 13b and 13c further highlights the discriminative capability of Criterion 3. While both cases share the same utilization completeness, namely $\mathcal{U} = 100\%$, their corresponding values of $\mathcal{C}$ differ substantially, being 10.00 and 1.02, respectively. This contrast demonstrates that Criterion 3 effectively captures differences in the distribution of p_k that are not reflected by $\mathcal{U}$. In particular, the near minimal value of $\mathcal{C}$ in Figure 13c correctly identifies this scenario as a collapsed codebook, which aligns with our intended interpretation.

Nevertheless, Criterion 3 alone is also insufficient for a comprehensive evaluation of codebook collapse. As illustrated by Figures 13a and 13d, identical values of $\mathcal{C}$ can correspond to markedly different levels of utilization completeness, as indicated by their substantially different values of $\mathcal{U}$. This discrepancy shows that $\mathcal{C}$ by itself cannot quantify the proportion of actively utilized code vectors.

Consequently, in this work, we adopt the joint use of Criterion 2 and Criterion 3 to quantitatively assess codebook collapse. Effective mitigation is achieved only when both metrics attain sufficiently large values, indicating that the codebook is both fully utilized and balanced in its assignment distribution.

E The Significant Impact of Distribution Variance on Quantization Error

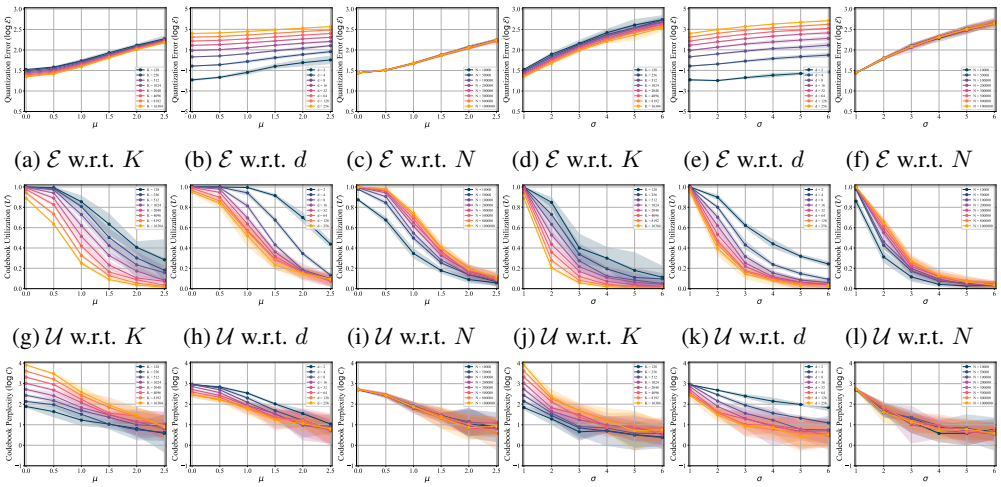
As discussed in Section 2.3 and 2.4, the optimal criterion triple is achieved when the distributions $\mathcal{P}_A$ and $\mathcal{P}_B$ are identical. In this section, we further analyze the criterion triple by the lens of distribution variance under the condition that both distributions are identical. Specifically, we first sample a set of feature vectors $\{\boldsymbol{z}_i\}_{i=1}^N$ along with a set of code vectors $\{\boldsymbol{e}_k\}_{k=1}^K$ from the distribution $\mathcal{N}_d(\mathbf{0}, \sigma^2\boldsymbol{I})$ or the distribution $\text{Unif}_d(-\zeta, \zeta)$. We then calculate the evaluation criteria according to their definitions in Section 2.2. As demonstrated in Table 9, σ and ζ have a substantial impact on $\mathcal{E}$, while $\mathcal{U}$ and $\mathcal{C}$ remains largely unaffected.

Table 9: The criterion triple influence by the distribution variance.

Evaluation Criteria	σ					ζ				
	0.0001	0.001	0.01	0.1	1.0	0.0001	0.001	0.01	0.1	1.0
$\mathcal{E}$	1.25e-8	1.25e-6	1.25e-4	1.24e-2	1.25	3.27e-9	3.27e-7	3.27e-5	3.27e-3	0.327
$\mathcal{U}$	0.9934	0.9938	0.9940	0.9934	0.9941	0.9993	0.9986	0.9990	0.9992	0.9989
$\mathcal{C}$	7265.3	7260.3	7267.7	7255.0	7275.8	7380.2	7372.2	7387.9	7397.5	7391.6

This experimental finding suggests that when the distribution variance of the feature vectors is uncontrollable or unknown, reporting a comparison of quantization error among various VQ methods is unreasonable. This is because the improvement in quantization error is predominantly attributed to

[3]This difference arises because, in Figure 13a, only half of the code vectors have nonzero utilization probabilities p_k, as defined in Criterion 3, whereas in Figure 13b, all code vectors satisfy $p_k > 0$.



(a) $\mathcal{E}$ w.r.t. K (b) $\mathcal{E}$ w.r.t. d (c) $\mathcal{E}$ w.r.t. N (d) $\mathcal{E}$ w.r.t. K (e) $\mathcal{E}$ w.r.t. d (f) $\mathcal{E}$ w.r.t. N

(g) $\mathcal{U}$ w.r.t. K (h) $\mathcal{U}$ w.r.t. d (i) $\mathcal{U}$ w.r.t. N (j) $\mathcal{U}$ w.r.t. K (k) $\mathcal{U}$ w.r.t. d (l) $\mathcal{U}$ w.r.t. N

(m) $\mathcal{C}$ w.r.t. K (n) $\mathcal{C}$ w.r.t. d (o) $\mathcal{C}$ w.r.t. N (p) $\mathcal{C}$ w.r.t. K (q) $\mathcal{C}$ w.r.t. d (r) $\mathcal{C}$ w.r.t. N

Figure 14: Quantitative analyses of the criterion triple when $\mathcal{P}_A$ and $\mathcal{P}_B$ are Gaussian distributions.

the reduction in distribution variance rather than the effectiveness of the VQ methods. To evaluate various VQ methods in terms of the criterion triple, we establish an atomic and fair experimental setting in Section 4, where the feature distributions for all VQ methods are identical.

F Interpretation of Qualitative Distributional Matching Results

This section interprets the experimental results presented in Figure 3. The VQ process relies on nearest neighbor search for code vector selection. As evident from Figure 3a to 3d, actively selected code vectors are predominantly those located in close proximity to or within the feature distribution, while distant ones remain unselected. This leads to highly uneven code vector utilization p_k, with those closer to the feature distribution being excessively used. This elucidates the significantly low $\mathcal{U}$ and $\mathcal{C}$ observed in Figure 3a. Furthermore, a notable quantization error, e.g., $\mathcal{E} = 1.19$ in Figure 3a, arises when the codebook and feature distributions are mismatched, forcing feature vectors outside the codebook to settle for distant code vectors. Conversely, as the disk centers align, leading to a closer match between the two distributions, an increased number of code vectors become actively engaged. Additionally, code vectors are utilized more uniformly, and feature vectors can select nearer counterparts. This accounts for the improvement of criterion triple values towards optimality as the distributions align.

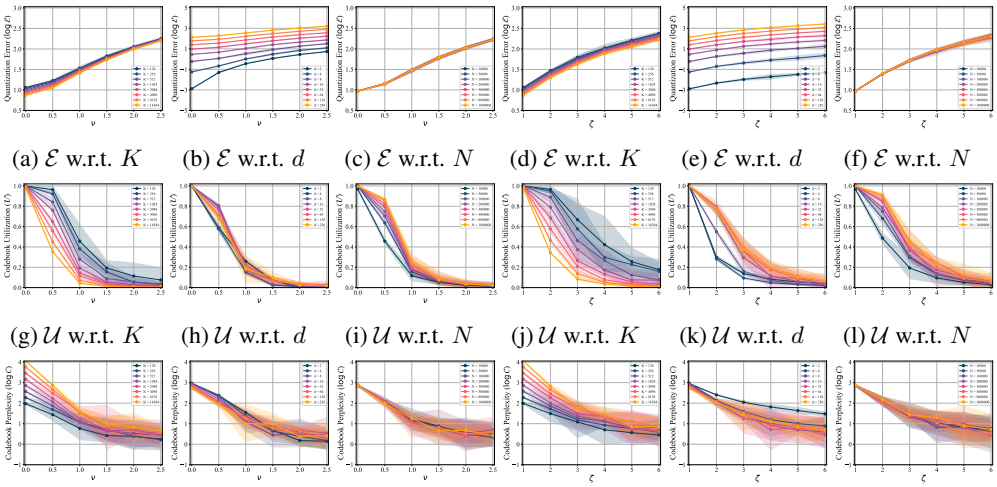
Analogously, we can employ nearest neighbor search to interpret the second case. When code vectors are distributed within the range of feature vectors, as illustrated in Figure 3e and Figure 3f, the majority of code vectors would be actively utilized, ensuring high $\mathcal{U}$. However, the utilization of these code vectors is not uniform; code vectors on the periphery of the codebook distribution are more frequently used, leading to relatively low $\mathcal{C}$. Feature vectors on the periphery will have larger distances to their nearest code vectors, resulting in higher $\mathcal{E}$. Conversely, when feature vectors fall within the range of code vectors, as depicted in Figure 3g and Figure 3h, outer code vectors remain largely unused, leading to a lower $\mathcal{U}$ and $\mathcal{C}$. Since only inner code vectors are active, each feature vector can find a nearby counterpart, maintaining low $\mathcal{E}$.

G Supplementary Quantitative Analyses on Distribution Matching: Further Supporting the Main Findings in Section 2.3

To further elucidate the effects of the distributional matching, we conduct more quantitative analyses centered around the criterion triple $(\mathcal{E}, \mathcal{U}, \mathcal{C})$.

G.1 Codebook Distribution and Feature Distribution are Gaussian Distributions

We begin by assuming that the distributions $\mathcal{P}_A$ and $\mathcal{P}_B$ are Gaussian. We generate a set of feature vectors $\{\boldsymbol{z}_i\}_{i=1}^N$ from $\mathcal{N}_d(\mathbf{0}, \boldsymbol{I})$ and a set of code vectors $\{\boldsymbol{e}_k\}_{k=1}^K$ from $\mathcal{N}_d(\mu \cdot \mathbf{1}, \boldsymbol{I})$, with μ varying



(a) $\mathcal{E}$ w.r.t. K (b) $\mathcal{E}$ w.r.t. d (c) $\mathcal{E}$ w.r.t. N (d) $\mathcal{E}$ w.r.t. K (e) $\mathcal{E}$ w.r.t. d (f) $\mathcal{E}$ w.r.t. N

(g) $\mathcal{U}$ w.r.t. K (h) $\mathcal{U}$ w.r.t. d (i) $\mathcal{U}$ w.r.t. N (j) $\mathcal{U}$ w.r.t. K (k) $\mathcal{U}$ w.r.t. d (l) $\mathcal{U}$ w.r.t. N

(m) $\mathcal{C}$ w.r.t. K (n) $\mathcal{C}$ w.r.t. d (o) $\mathcal{C}$ w.r.t. N (p) $\mathcal{C}$ w.r.t. K (q) $\mathcal{C}$ w.r.t. d (r) $\mathcal{C}$ w.r.t. N

Figure 15: Quantitative analyses of the criterion triple when $\mathcal{P}_A$ and $\mathcal{P}_B$ are uniform distributions.

within $\{0.0, 0.5, 1.0, 1.5, 2.0, 2.5\}$. The criterion triple results are presented in Figures 14a to 14c, Figures 14g to 14i, and Figures 14m to 14o. Across all tested configurations of K, d, N, we consistently observe that when $\mu = 0$ — indicating identical distributions between $\mathcal{P}_A$ and $\mathcal{P}_B$ — the criterion triple achieves the lowest $\mathcal{E}$, highest $\mathcal{U}$, and largest $\mathcal{C}$. This empirical evidence reinforces the effectiveness of aligning feature and codebook distributions in VQ.

Additionally, we further analyze the criterion triple by varying the covariance matrix. We sample a set of feature vectors $\{\boldsymbol{z}_i\}_{i=1}^N$ from the distribution $\mathcal{N}_d(\mathbf{0}, \boldsymbol{I})$ and a set of code vectors $\{\boldsymbol{e}_k\}_{k=1}^K$ from $\mathcal{N}_d(\mathbf{0}, \sigma^2\boldsymbol{I})$, where σ is selected from $\{1, 2, 3, 4, 5, 6\}$. The results for the criterion triple are shown in Figures 14d to 14f, Figures 14j to 14l, and Figures 14p to 14r. When $\sigma = 1$, indicating identical distributions between $\mathcal{P}_A$ and $\mathcal{P}_B$, all three evaluation criteria reach their optimal values: the lowest $\mathcal{E}$, highest $\mathcal{U}$, and largest $\mathcal{C}$ across all tested values of K, d, N. This result corroborates our earlier findings.

G.2 Codebook Distribution and Feature Distribution are Unifrom Distributions

The above conclusion holds when $\mathcal{P}_A$ and $\mathcal{P}_B$ are other types of distributions, such as the uniform distribution. As shown in Figure 15, we sample a set of feature vectors $\{\boldsymbol{z}_i\}_{i=1}^N$ from the distribution $\text{Unif}_d(-1, 1)$ and a set of code vectors $\{\boldsymbol{e}_k\}_{k=1}^K$ from $\text{Unif}_d(\nu - 1, \nu + 1)$, where ν is selected from the set $\{0.0, 0.5, 1.0, 1.5, 2.0, 2.5\}$ or from $\text{Unif}_d(-\zeta, \zeta)$, with ζ drawn from the set $\{1, 2, 3, 4, 5, 6\}$. We observe that when $\mu = 0$ or $\zeta = 1$—indicating that $\mathcal{P}_A$ and $\mathcal{P}_B$ have identical distributions—the performance in terms of the criterion triple is optimal, achieving the lowerest $\mathcal{E}$, the highest $\mathcal{U}$, and the largest $\mathcal{C}$ across all tested values of K, d, N. Therefore, we conclude that our quantitative analyses are distribution-agnostic and can be generalized to other distributions.

H Statistical Distances over Gaussian Distributions

We first introduce the definition of Wasserstein distance.

Definition 4. *The Wasserstein distance or earth-mover distance with p norm is defined as below:*

$$W_p(\mathbb{P}_r, \mathbb{P}_g) = \left(\inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\left[\|x - y\|^p\right]\right)^{1/p}. \tag{34}$$

where $\Pi(\mathcal{P}_r, \mathcal{P}_g)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are $\mathcal{P}_r$ and $\mathcal{P}_g$ respectively. Intuitively, when viewing each distribution as a unit amount of earth/soil, the Wasserstein distance (also known as earth-mover distance) represents the minimum cost of transporting "mass" from x to y to transform distribution $\mathcal{P}_r$ into distribution $\mathcal{P}_g$. When $p = 2$, this is called the quadratic Wasserstein distance.

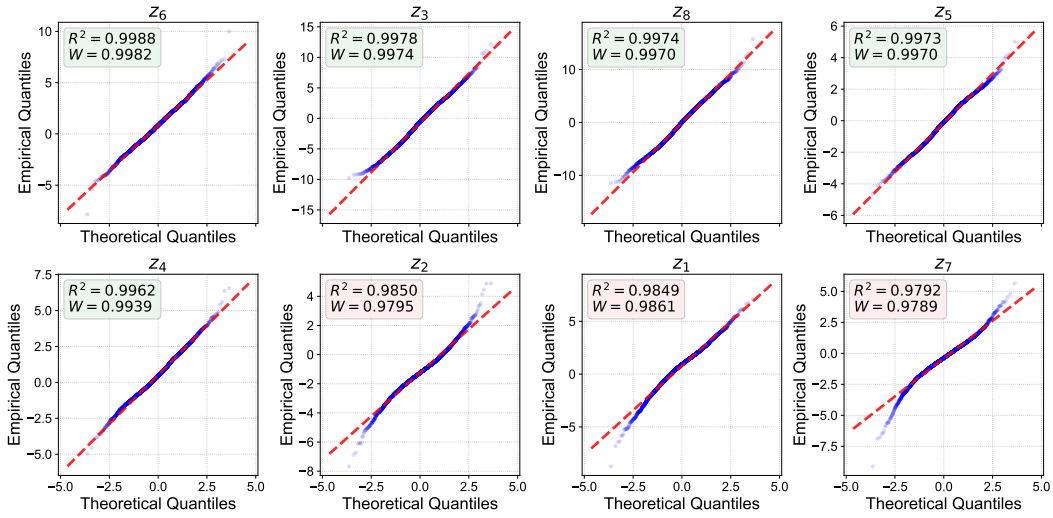


Figure 16: Comprehensive Q–Q plots for all latent dimensions. Subplots are ordered from the best-fitting dimension z_6 (top-left) to the least-fitting dimension z_7 (bottom-right) according to the R^2 score. **Visualization:** For visual clarity, blue points show a random subset of 5,000 samples. **Statistics:** The annotated R^2 and Shapiro–Wilk W statistics (green/red boxes) are computed on the **full validation set** ($N = 409{,}600$). Across all dimensions, the Q–Q plots indicate a strong Gaussian fit for the central mass of the distributions, with only mild deviations in the extreme tails.

In this paper, we achieve distributional matching using the quadratic Wasserstein distance under Gaussian distribution assumptions. We also examine other statistical distribution distances as potential loss functions for distributional matching and compare them with the Wasserstein distance. Specifically, we provide the Kullback-Leibler divergence and the Bhattacharyya distance over Gaussian distributions in Lemma 5 and Lemma 6. It can be observed that the KL divergence for two Gaussian distributions involves calculating the determinant of covariance matrices, which is computationally expensive in moderate and high dimensions. Moreover, the calculation of the determinant is sensitive to perturbations and it requires full rank (In the case of not full rank, the determinant is zero, rendering the logarithm of zero undefined), which can be impractical in many cases. Other statistical distances like Bhattacharyya Distance suffer from the same issue. In contrast, quadratic Wasserstein distance does not require the calculation of the determinant and full-rank covariance matrices.

Lemma 5 (Kullback-Leibler divergence [31])**.** *Suppose two random variables* $\mathbf{Z}_1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ *and* $\mathbf{Z}_2 \sim \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ *obey multivariate normal distributions, then Kullback-Leibler divergence between* $\mathbf{Z}1$ *and* $\mathbf{Z}_2$ *is:*

$$D_{\mathrm{KL}}(\mathbf{Z}_1, \mathbf{Z}_2) = \frac{1}{2}((\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) + \mathrm{tr}(\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1 - \mathbf{I}) + \ln \frac{\det \boldsymbol{\Sigma}_2}{\det \boldsymbol{\Sigma}_1}).$$

Lemma 6 (Bhattacharyya Distance [4])**.** *Suppose two random variables* $\mathbf{Z}_1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ *and* $\mathbf{Z}_2 \sim \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ *obey multivariate normal distributions,* $\boldsymbol{\Sigma} = \frac{1}{2}(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2)$*, then bhattacharyya distance between* $\mathbf{Z}1$ *and* $\mathbf{Z}_2$ *is:*

$$\mathcal{D}_B(\mathbf{Z}_1, \mathbf{Z}_2) = \frac{1}{8}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) + \frac{1}{2} \ln \frac{\det \boldsymbol{\Sigma}}{\sqrt{\det \boldsymbol{\Sigma}_1 \det \boldsymbol{\Sigma}_2}}.$$

I Statistical Assessment of Gaussianity

To further validate the Gaussian assumption discussed in Section 5.1, we conduct a comprehensive set of univariate and multivariate normality tests on the validation set. Specifically, we analyze $N = 409{,}600$ latent feature vectors ($d = 8$), extracted from 1,600 images in the FFHQ dataset.

Univariate Analysis. Figure 16 shows the Q–Q plots for all eight dimensions of the latent feature vector, sorted by their R^2 goodness-of-fit scores. Consistent with the observations in the main paper, the majority of dimensions (z_6, z_3, z_8, z_5, z_4) demonstrate an excellent fit to the Gaussian distribution, with $R^2 > 0.99$. Although the remaining dimensions exhibit mild deviations in the extreme tails, their

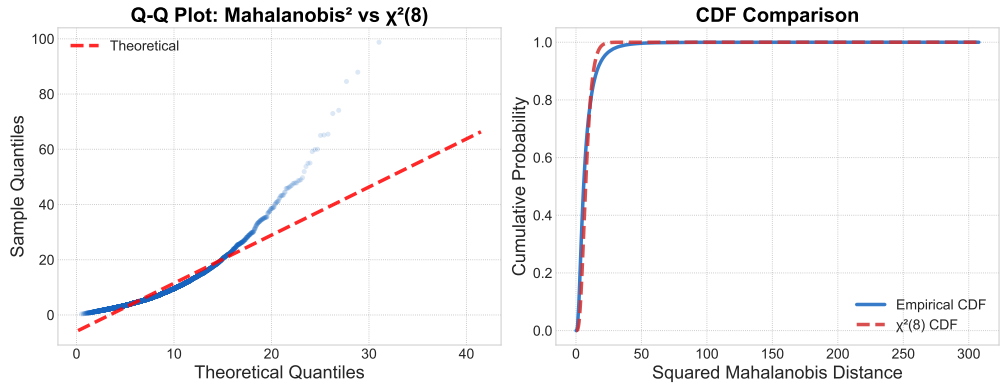


Figure 17: Multivariate normality assessment using Mahalanobis distance. Left (Q–Q plot): The squared Mahalanobis distances of the latent features are compared against the theoretical χ^2_8 quantiles. Deviations are primarily observed in the extreme tails, while the bulk of the samples closely follows the reference line. Right (CDF): The empirical cumulative distribution function (blue solid) closely matches the theoretical χ^2_8 CDF (red dashed) over the primary probability mass, indicating strong agreement in the high-density region.

Shapiro–Wilk statistics remain high ($W > 0.97$), indicating that the central mass of the distributions is well approximated by a Gaussian.

Multivariate Analysis. To assess joint normality, we compute the squared Mahalanobis distance for each validation sample,

$$d_i^2 = (z_i - \mu)^\top \Sigma^{-1} (z_i - \mu).$$

Under the multivariate Gaussian hypothesis, d_i^2 should follow a Chi-squared distribution with $d = 8$ degrees of freedom (χ^2_8). The results are shown in Figure 17. The Q–Q plot (Left) indicates strong agreement between the empirical and theoretical quantiles for the majority of samples, particularly in the high-density region near the origin, while heavier tails emerge only in the extreme quantiles. To further quantify this alignment, Figure 17 (Right) compares the cumulative distribution functions (CDFs). The empirical CDF (blue solid line) closely matches the theoretical χ^2_8 CDF (red dashed line) across the primary probability mass.

Overall, while deep feature representations naturally exhibit heavy-tailed behavior due to outliers, both univariate and multivariate tests consistently indicate that the bulk of the distribution is well modeled by a Gaussian. This provides strong empirical justification for using the Wasserstein distance, which is derived under Gaussian assumptions, as an efficient and effective optimization surrogate. Moreover, the comparable performance of MMD VQ, which does not impose any parametric distributional assumptions, further suggests that the Gaussian approximation captures the essential statistical structure required for high-fidelity tokenization.

Table 10: Hyperparameters for the experiments in Table 1, 2, 5, 11, 12, 7.

Frameworks	VQ-VAE	VQ-VAE	VQGAN	VQGAN
Dataset	CIFAR-10/SVHN	FFHQ/ImageNet	FFHQ	ImageNet
Input size	$32 \times 32 \times 3$	$256 \times 256 \times 3$	$256 \times 256 \times 3$	$256 \times 256 \times 3$
Latent size	$8 \times 8 \times 8$	$16 \times 16 \times 8$	$16 \times 16 \times 32$	$16 \times 16 \times 32$
encoder/decoder channels	64	64	160	160
encoder/decoder channel mult.	$[1, 1, 2]$	$[1, 1, 2, 2, 4]$	$[1, 1, 2, 2, 4]$	$[1, 1, 2, 2, 4]$
Batch size	128	32	32	32
Initial Learning rate lr	5×10^{-5}	5×10^{-5}	1×10^{-5}	1×10^{-5}
Perceptual loss Coefficient	0	0	1.0	1.0
Adversarial loss Coefficient	0	0	0.4	0.4
Codebook dimensions	8	8	32	32
Training Epochs	50	30/4	30	5/10/15
GPU Resources	1 V100 16GB	1 A100 40GB	2 H100 80GB	2 H100 80GB

J The Experimental Details

J.1 Synthetic Experimental Details in Section 2.3

As depicted in Figure 3 in Section 2.3, we conduct a qualitative analyses of the criterion triple. Specifically, we sample a set of feature vectors $\{\boldsymbol{z}_i\}_{i=1}^N$ from within the red circle, and a collection of code vectors $\{\boldsymbol{e}_k\}_{k=1}^K$ from within the green circle, with parameters set to $K = 400$, $N = 10000$ and $d = 2$ for the calculation of the criterion triple $(\mathcal{E}, \mathcal{U}, \mathcal{C})$. For the visualization, we select 10% of the feature vectors and 90% of the code vectors for plotting.

J.2 Synthetic Experimental Details in Appendix G

As illustrate in Figure 14 in Appendix G.1, we undertake comprehensive quantitative analyses centered around the criterion triple $(\mathcal{E}, \mathcal{U}, \mathcal{C})$. In these analyses, we assume that $\mathcal{P}_A$ and $\mathcal{P}_B$ are Gaussian distributions, from which we sample a set of feature vectors $\{\boldsymbol{z}_i\}_{i=1}^N$ and a collection of code vectors $\{\boldsymbol{e}_k\}_{k=1}^K$. The default parameters are set to $N = 200,000$, $K = 1024$, and $d = 32$ for all figures unless otherwise specified. For instance, in Figure 14a, N and d are taken at their default values, while the K is varied within the set $\{128, 256, 512, 1024, 2048, 4096, 8192, 16284\}$. Additionally, each synthetic experiment is repeated five times, and the average results are reported, along with the calculation of 95% confidence intervals. In all figures, mean results are represented by points, while the confidence intervals are shown as shaded areas. Identical parameter settings are employed when $\mathcal{P}_A$ and $\mathcal{P}_B$ are uniform distributions, as illustrated in Figure 15 in Appendix G.2.

J.3 Synthetic Experimental Details in Appendix E

We set $K = 8192, d = 8, N = 100000$ when calculating the criterion triple $(\mathcal{E}, \mathcal{U}, \mathcal{C})$ in Appendix E. Each synthetic experiment is repeated five times, and the average results are reported in Table 9.

J.4 Synthetic Experimental Details in Section 4

We provide experimental details of Figure 5 in Section 4. In our experimental setup, we evaluate five distinct VQ algorithms using the criterion triple $(\mathcal{E}, \mathcal{U}, \mathcal{C})$. All experiments run on a single NVIDIA A100 GPU, with a codebook size K of 16,384 and dimensionality d of 8 across all algorithms. Each algorithm trains for 2,000 steps, with 50,000 feature vectors sampled from the specified Gaussian distribution at each step. For *Wasserstein VQ*, Vanilla VQ, and VQ + MLP, we use the SGD optimizer for training. For VQ EMA and Online Clustering, we use classical clustering algorithms—k-means [5] and k-means++[1]—to update code vectors.

J.5 Experimental Details in Section 5

Data Augmentation For FFHQ and ImageNet-1k datasets, we follow LLama Gen [45] and apply iterative box downsampling to resize images to 256×256 resolution. For CIFAR-10 and SVHN, the images are kept at their original resolution.

Encoder-Decoder Architecture In VQ-VAE For the ImageNet and FFHQ datasets, our proposed Wasserstein VQ and all baseline methods adopt identical encoder-decoder architectures and parameter configurations. Across all baselines in these frameworks, the encoder—a U-Net [40]—downscales the input image by a factor of 16. For CIFAR-10 and SVHN datasets, the encoder reduces the input resolution by a factor of 4. Detailed hyperparameter configurations are summarized in Table 10.

Encoder-Decoder Architecture In VQGAN To accelerate adversarial training in VQGAN, our proposed Wasserstein VQ-a/b/c and Wasserstein VAR-a/b/c models on the ImageNet dataset, as well as the Wasserstein VQ model on the FFHQ dataset, are built upon the VQ-Transplant framework. Within this framework, we deploy the pretrained VAR tokenizer [47] for initialization. Consequently, the encoder–decoder architecture is identical to that used in the VAR tokenizer.

Training Details in VQ-VAE All experiments employ identical training settings: we use the AdamW optimizer [32] with $\beta_1 = 0.9$ and $\beta_1 = 0.95$, an initial learning rate lr, and apply a half-

cycle cosine decay schedule following a linear warm-up phase. For specific details on training epochs and batch sizes, refer to Table 10.

Training Details in VQ-GAN Our proposed Wasserstein VQ and Wasserstein VAR were trained on two NVIDIA H100 GPUs using the AdamW optimizer [32] with $\beta_1 = 0.9$ and $\beta_1 = 0.95$. During VQ module substitution, we used an initial learning rate of 10^{-4} with linear decay to 10^{-5}. For decoder adaptation, the learning rate remained constant at $lr = 10^{-5}$. For specific details on training epochs and batch sizes, refer to Table 10. For adversarial training, we follow the VQ-Transplant framework [12], which employs a frozen DINO-S [6, 37] discriminator with an architecture reminiscent of StyleGAN [25, 24]. Consistent with VQ-Transplant [12], we further incorporate DiffAug [58], consistency regularization [55], and LeCAM regularization [48] to stabilize discriminator training.

Loss Weight in VQ-VAE For all three baselines, β is typically set to a value within the range $[0.25, 2]$. In our experiments, β is set to a fixed value of 1.0. For our proposed *Wasserstein VQ* model, we set β to a much smaller value, e.g., $\beta = 0.1$. The smaller β values enable the Wasserstein distance to dominate the loss function, thereby more effectively narrowing the gap between the distributions.

Loss Weight in VQGAN For our proposed Wasserstein VQ and Wasserstein VAR models, the perceptual loss weight λ_P is fixed at 1. In multi-scale quantization experiments, we set $\lambda_G = 0.5$, whereas in fixed-scale quantization experiments, $\lambda_G = 0.4$. The coefficient γ is set to 0.2 for all configurations incorporating the Wasserstein distance (i.e., Wasserstein VQ and Wasserstein VAR).

K Supplementary Results and Analyses in VQ-VAE Framework

K.1 VQ-VAE Performance on CIFAR-10 and SVHN datasets

Due to space limitations in the main text, we have relocated the VQ-VAE evaluation on CIFAR-10 and SVHN datasets to the appendix. As demonstrated in Table 11, 12, our Wasserstein VQ consistently outperforms all baselines across both datasets, achieving superior results on nearly all evaluation metrics regardless of codebook size. Notably, we observe that Wasserstein VQ fails to reach 100% codebook utilization on SVHN, which may be attributed to the dataset's limited diversity.

Table 11: Comparison of VQ-VAEs trained on CIFAR-10 dataset following [49].

Approaches	Tokens	Codebook Size	$\mathcal{U}$ (↑)	$\mathcal{C}$ (↑)	PSNR(↑)	SSIM(↑)	Rec. Loss (↓)
Vanilla VQ	64	8192	2.7%	186.9	27.15	0.83	0.0147
EMA VQ	64	8192	99.7%	6416.1	29.43	0.88	0.0095
Online VQ	64	8192	22.1%	995.4	28.20	0.85	0.0123
Wasserstein VQ	64	8192	**100.0%**	**7781.8**	**29.88**	**0.90**	**0.0085**
Vanilla VQ	64	16384	1.6%	220.3	27.36	0.84	0.0141
EMA VQ	64	16384	80.8%	10557.3	29.43	0.88	0.0093
Online VQ	64	16384	13.4%	798.5	27.54	0.82	0.0141
Wasserstein VQ	64	16384	**100.0%**	**15583.7**	**30.19**	**0.90**	**0.0080**
Vanilla VQ	64	32768	0.5%	154.8	27.10	0.83	0.0150
EMA VQ	64	32768	54.4%	14427.0	29.57	0.88	0.0091
Online VQ	64	32768	7.2%	1556.0	28.84	0.87	0.0106
Wasserstein VQ	64	32768	**99.0%**	**29845.1**	**30.63**	**0.91**	**0.0071**

K.2 Analyses on Codebook Size and Dimensionality

Analyses of Codebook Size We investigate the impact of the codebook size K on the performance of VQ by varying across a wide range: $K \in [1024, 2048, 4096, 8192, 16384, 50000, 100000]$. As shown in Table 1, 13, the vanilla VQ model suffers from severe codebook collapse even with a relatively small K, such as $K = 1024$. In contrast, improved algorithms like EMA VQ and Online VQ can handle smaller codebook sizes effectively, but they still experience codebook collapse when K is very large, e.g., $K \geq 50000$. Notably, the *Wasserstein VQ* model consistently maintains 100% codebook utilization, irrespective of the codebook size. This underscores the effectiveness of distributional matching via the quadratic Wasserstein distance in mitigating the issue of codebook collapse.

Table 12: Comparison of VQ-VAEs trained on SVHN dataset following [49].

Approaches	Tokens	Codebook Size	$\mathcal{U}$ (↑)	$\mathcal{C}$ (↑)	PSNR(↑)	SSIM(↑)	Rec. Loss (↓)
Vanilla VQ	64	8192	8.1%	533.1	37.81	0.97	0.0018
EMA VQ	64	8192	56.8%	3363.0	40.38	**0.98**	0.0010
Online VQ	64	8192	27.8%	1325.1	39.04	0.97	0.0016
Wasserstein VQ	64	8192	**88.2%**	**6154.5**	**41.04**	**0.98**	**0.0009**
Vanilla VQ	64	16384	3.4%	446.0	37.87	0.97	0.0017
EMA VQ	64	16384	22.2%	2593.8	40.19	**0.98**	0.0011
Online VQ	64	16384	13.5%	1090.5	39.12	0.97	0.0014
Wasserstein VQ	64	16384	**87.5%**	**11967.2**	**41.49**	**0.98**	**0.0008**
Vanilla VQ	64	32768	1.8%	467.5	37.87	0.97	0.0017
EMA VQ	64	32768	35.8%	7662.9	40.25	**0.98**	0.0010
Online VQ	64	32768	7.0%	1334.8	39.26	0.97	0.0014
Wasserstein VQ	64	32768	**88.7%**	**24376.3**	**41.84**	**0.98**	**0.0008**

Table 13: Supplementary comparison of VQ-VAEs trained on FFHQ dataset following [49] w.r.t codebook size K.

Approaches	Tokens	Codebook Size	$\mathcal{U}$ (↑)	$\mathcal{C}$ (↑)	PSNR(↑)	SSIM(↑)	Rec. Loss (↓)
Vanilla VQ	256	1024	51.7%	446.2	27.64	73.0	0.0125
EMA VQ	256	1024	74.1%	618.9	27.66	72.7	0.0125
Online VQ	256	1024	**100.0%**	759.3	28.08	74.0	0.0114
Wasserstein VQ	256	1024	**100.0%**	**977.4**	**28.11**	**74.4**	**0.0112**
Vanilla VQ	256	2048	27.6%	453.0	27.78	73.8	0.0121
EMA VQ	256	2048	**100%**	1608.0	**28.39**	74.9	**0.0107**
Online VQ	256	2048	**100%**	1462.6	28.34	74.6	0.0108
Wasserstein VQ	256	2048	**100%**	**1840.5**	28.32	**75.3**	**0.0107**
Vanilla VQ	256	4096	12.5%	435.0	27.84	73.7	0.0119
EMA VQ	256	4096	76.7%	2443.1	28.49	75.0	0.0104
Online VQ	256	4096	70.7%	1600.0	28.25	74.1	0.0110
Wasserstein VQ	256	4096	**100%**	**3895.4**	**28.54**	**75.1**	**0.0102**
Vanilla VQ	256	8192	5.6%	398.1	27.69	73.5	0.0122
EMA VQ	256	8192	28.9%	1839.2	28.39	74.8	0.0106
Online VQ	256	8192	34.9%	1474.4	28.15	73.9	0.0113
Wasserstein VQ	256	8192	**100%**	**7731.5**	**28.81**	**76.2**	**0.0099**

Table 14: Analysis On codebook dimension by the comparison of VQ-VAEs trained on CIFAR-10 dataset following [49]. (The codebook size K is fixed to 16384)

Approaches	Tokens	Codebook Dim	$\mathcal{U}$ (↑)	$\mathcal{C}$ (↑)	PSNR(↑)	SSIM(↑)	Rec. Loss (↓)
Vanilla VQ	256	2	3.8%	532.2	27.00	0.80	0.0162
EMA VQ	256	2	97.6%	**14460.3**	27.25	0.80	**0.0155**
Online VQ	256	2	9.0%	611.8	26.62	0.79	0.0178
Wasserstein VQ	256	2	**99.3%**	12278.9	**27.30**	**0.81**	**0.0155**
Vanilla VQ	256	4	1.3%	176.7	27.15	0.83	0.0149
EMA VQ	256	4	99.8%	13153.9	29.57	0.89	0.0092
Online VQ	256	4	11.1%	877.7	26.69	0.79	0.0173
Wasserstein VQ	256	4	**100.0%**	**15724.7**	**29.93**	**0.89**	**0.0087**
Vanilla VQ	256	8	1.6%	220.3	27.36	0.84	0.0141
EMA VQ	256	8	80.8%	10557.3	29.43	0.88	**0.0009**
Online VQ	256	8	13.4%	798.5	27.54	0.82	0.0141
Wasserstein VQ	256	8	**100.0%**	**15583.7**	**30.19**	**0.90**	0.0080
Vanilla VQ	256	16	1.1%	150.8	27.05	0.83	0.0152
EMA VQ	256	16	32.5%	4169.2	29.31	0.88	0.0099
Online VQ	256	16	18.2%	2051.0	28.29	0.85	0.0122
Wasserstein VQ	256	16	**99.2%**	**14832.2**	**30.27**	**0.91**	**0.0078**
Vanilla VQ	256	32	0.7%	94.37	26.67	0.81	0.0165
EMA VQ	256	32	7.0%	942.7	28.24	0.85	0.0122
Online VQ	256	32	18.8%	2278.0	28.92	0.87	0.0104
Wasserstein VQ	256	32	**96.4%**	**14056.9**	**30.39**	**0.91**	**0.0076**

Analyses of Codebook Dimensionality We further investigate the impact of codebook dimensionality d on VQ performance. Conducting experiments on CIFAR-10 with dimensionality d ranging from 2 to 32, our proposed Wasserstein VQ consistently outperforms all baselines regardless of dimensionality, as shown in Table 14. Notably, we observe the curse of dimensionality phe-

Table 15: Sensitivity analysis of γ trained on FFHQ dataset following [49].

Approaches	γ	Tokens	Codebook Size	$\mathcal{U}$ (↑)	$\mathcal{C}$ (↑)	PSNR(↑)	SSIM(↑)	Rec. Loss (↓)
Wasserstein VQ	0.0	256	16384	3.8%	527.2	27.83	73.8	0.0119
Wasserstein VQ	0.00001	256	16384	24.8%	903.0	27.97	74.2	0.0114
Wasserstein VQ	0.0001	256	16384	52.3%	6764.3	28.70	75.5	0.0100
Wasserstein VQ	0.001	256	16384	90.6%	9988.0	28.93	76.7	0.0094
Wasserstein VQ	0.01	256	16384	**100%**	14952.5	29.06	76.7	**0.0092**
Wasserstein VQ	0.1	256	16384	**100%**	**15943.5**	**29.07**	76.7	**0.0092**
Wasserstein VQ	0.5	256	16384	**100%**	15713.3	29.03	76.6	0.0093
Wasserstein VQ	1.0	256	16384	**100%**	14712.4	29.02	**76.9**	0.0093

nomenon—performance degrades as dimensionality increases. Vanilla VQ exhibits the most severe degradation, followed by EMA VQ and Online VQ, while our Wasserstein VQ shows only minimal codebook utilization reduction.

K.3 Sensitivity Analysis on γ.

We conduct a sensitivity analysis with respect to γ on the FFHQ dataset by varying $\gamma \in \{0, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$. As reported in Table 15, when $\gamma = 0$, Wasserstein VQ yields the worst performance, as it degenerates into the vanilla VQ formulation without distributional matching. As γ increases from 10^{-5} to 10^{-3}, the performance of Wasserstein VQ consistently improves. When γ reaches 10^{-2}, Wasserstein VQ achieves full (100%) codebook utilization along with competitive quantitative results. Moreover, within the range $\gamma \in [10^{-2}, 1]$, the performance of Wasserstein VQ remains stable, indicating that the method is not sensitive to the precise choice of γ once it exceeds a moderate threshold.

K.4 Computational Overhead Comparison among Various VQ Approaches

To evaluate the runtime efficiency of different VQ approaches, we measure the forward and backward pass times of the VQ module over 100 iterations across three different codebook sizes, with the feature dimension set to $d = 8$ and the number of data samples $N = 8192$. As shown in Table 4, even at large codebook sizes (specifically, $K \geq 50,000$), the runtime of Wasserstein VQ is only slightly longer than that of Vanilla VQ. This demonstrates that Wasserstein VQ maintains high computational efficiency, and incorporating the quadratic Wasserstein distance does not introduce significant time overhead. Notably, while Online VQ exhibits substantial runtime increases at large codebook sizes, Wasserstein VQ remains considerably more efficient, further underscoring its scalability.

K.5 Discussion with VQ-WAE [50]

VQ-WAE [50] introduces an alternative approach to distributional matching by employing Optimal Transport to optimize codebook vectors. Compared with our proposed distributional matching method, there are three key differences.

First, regarding theoretical contributions: VQ-WAE [50] claims that achieving optimal transport (OT) between code vectors and feature vectors yields the best reconstruction performance. Their notion of optimality encompasses both the VQ process and the encoder-decoder reconstruction pipeline. While we contend that incorporating complex encoder-decoder functions renders rigorous theoretical analysis intractable, VQ-WAE nevertheless asserts this conclusion. In contrast, our work deliberately excludes encoder-decoder components, focusing solely on the VQ process, which admits rigorous mathematical modeling. Through our proposed criterion triple, we theoretically prove that distributional matching guarantees optimal performance.

Second, regarding distribution modeling: VQ-WAE [50] assumes both code vectors and feature vectors follow uniform discrete distributions, whereas our method models them as continuous distributions. Specifically, VQ-WAE [50] represents the distributions of feature vectors $\{\boldsymbol{z}_i\}_{i=1}^N$ and code vectors $\{\boldsymbol{e}_k\}_{k=1}^K$ as empirical measures:

$$\mathcal{P}_A = \frac{1}{N}\sum_{i=1}^{N}\delta_{\boldsymbol{z}_i}, \quad \mathcal{P}_B = \frac{1}{N}\sum_{k=1}^{K}\delta_{\boldsymbol{e}_k} \tag{35}$$

Table 16: Reconstruction performance (↓: the lower the better and ↑: the higher the better). †:Results cited from VQ-WAE [50]. Codebook size K is fixed to 512.

Dataset	Model	Tokens	SSIM (↑)	PSNR (↑)	LPIPS (↓)	Rec. Loss (↓)	Perplexity (↑)
CIFAR10	VQ-VAE†	64	70	23.14	0.35		69.8
	SQ-VAE†	64	80	26.11	0.23		434.8
	VQ-WAE†	64	80	25.93	0.23		497.3
	VQ-WAE (Our run)	64	13	14.60	0.41	0.247	1.0
	Vanilla VQ	64	83	27.19	0.03	0.015	192.5
	EMA VQ	64	84	27.97	0.04	0.013	436.1
	Online VQ	64	84	27.87	0.04	0.013	451.4
	Wasserstein VQ	64	86	28.26	0.03	0.012	481.7
SVHN	VQ-VAE†	64	88	26.94	0.17		114.6
	SQ-VAE†	64	96	35.37	0.06		389.8
	VQ-WAE†	64	96	34.62	0.07		485.1
	VQ-WAE (Our run)	64	25	15.87	0.26	0.2026	1.0
	Vanilla VQ	64	97	38.18	0.01	0.0016	407.1
	EMA VQ	64	97	38.35	0.01	0.0017	408.9
	Online VQ	64	97	38.54	0.01	0.0017	421.5
	Wasserstein VQ	64	97	38.25	0.01	0.0016	423.5

where $\delta_{\boldsymbol{z}_i}$ and $\delta_{\boldsymbol{e}_k}$ denote Dirac delta functions centered at $\boldsymbol{z}_i$ and $\boldsymbol{e}_k$, respectively. To align $\mathcal{P}_A$ and $\mathcal{P}_B$, VQ-WAE formulates the OT problem as:

$$\begin{aligned} &\min_{\mathbf{P}\in\Pi(\mathcal{P}_A,\mathcal{P}_B)} \sum_{i=1}^{N}\sum_{k=1}^{K} P_{ik}\|\boldsymbol{z}_i-\boldsymbol{e}_k\|^2, \\ &\text{s.t.} \quad \mathbf{P}\mathbf{1}_K=\frac{1}{N}\mathbf{1}_N, \quad \mathbf{P}^\top\mathbf{1}_N=\frac{1}{K}\mathbf{1}_K, \quad P_{ik}\geq 0 \quad \forall i,k, \end{aligned} \tag{36}$$

where $\mathbf{P}$ is the transport plan, and the feasible set is:

$$\Pi(\mathcal{P}_A,\mathcal{P}_B)=\left\{\mathbf{P}\in\mathbb{R}_+^{N\times K}\;\middle|\;\mathbf{P}\mathbf{1}_K=\frac{1}{N}\mathbf{1}_N,\ \mathbf{P}^\top\mathbf{1}_N=\frac{1}{K}\mathbf{1}_K\right\} \tag{37}$$

In contrast, we simplify the distributional assumption by modeling $\mathcal{P}_A$ and $\mathcal{P}_B$ as Gaussian distributions.

Third, regarding computational efficiency, The OT problem in VQ-WAE is prohibitively complex, whereas our quadratic Wasserstein distance incurs minimal overhead. To mitigate complexity, VQ-WAE employs a Kantorovich potential network. However, upon reproducing their code (no official implementation was released; we derived it from their ICLR 2023 supplementary material, [4]), we observed severe non-convergence—the method degenerated to using a single code vector, failing to achieve distributional matching. Notably, VQ-WAE underperformed all other VQ baselines (Table 16).

In comparison, our quadratic Wasserstein distance (Equation 15) requires only low-dimensional matrix operations (e.g., $d = 8$), achieving superior performance and effective matching (Figure 7).

[4] See https://openreview.net/forum?id=Z8qk2iM5uLI. We includes the reproduced code and training logs of VQ-WAE in our supplementary materials.