PUSHING TOWARD THE SIMPLEX VERTICES: A SIMPLE REMEDY FOR CODE COLLAPSE IN SMOOTHED VECTOR QUANTIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Vector quantization, which discretizes a continuous vector space into a finite set of representative vectors (a *codebook*), has been widely adopted in modern machine learning. Despite its effectiveness, vector quantization poses a fundamental challenge: the non-differentiable quantization step blocks gradient backpropagation. *Smoothed* vector quantization addresses this issue by relaxing the discrete selection of a codebook vector into a weighted combination of codebook entries, represented as the matrix product of a simplex vector and the codebook. Effective smoothing requires two properties: (1) smoothed code-selection vectors should remain close to a onehot vector, ensuring tight approximation, and (2) all codebook entries should be utilized, preventing *code collapse*. Existing methods typically address these desiderata separately. By contrast, the present study introduces a simple and intuitive regularization that promotes both simultaneously by minimizing the distance between each simplex vertex and its K-nearest smoothed code-selectors. Representative benchmarks on image encoding demonstrate that the proposed method achieves more effective codebook utilization and improves performance over prior approaches.

1 INTRODUCTION

Vector quantization is a method for discretizing a continuous vector space (Gray, 1984; van den Oord et al., 2017). It maps each vector in the continuous space to the nearest element of a finite set of representative vectors (a.k.a. a *codebook*). The resulting discrete representations are easier to manipulate and interpret than the original continuous forms, and have proved effective across diverse applications, including image generation (Esser et al., 2021; Ramesh et al., 2021; Rombach et al., 2022; Yu et al., 2022b), speech recognition (Baevski et al., 2020a;b), and speech/music generation (Hadjeres & Crestel, 2020; Dhariwal et al., 2020; Zhang et al., 2023; Chen et al., 2025).

When integrated into deep neural networks, however, vector quantization introduces a fundamental challenge: quantization is a non-differentiable operation that blocks gradient backpropagation (van den Oord et al., 2017). Accordingly, some approximation is required to enable learning through quantization.

One effective workaround is to *smooth* vector quantization (Jang et al., 2017). The selection of a codebook vector can be expressed as multiplying the codebook matrix by its corresponding onehot vector $\mathbf{e}_m := (0, \ldots, 1, \ldots, 0)^\mathsf{T}$, whose nonzero entry at m indexes the chosen vector. *Smoothed* vector quantization relaxes this onehot vector to lie within the simplex $\Delta^{M-1} := \{(p_1, \ldots, p_M) \in \mathbb{R}^M \mid \sum_{m=1}^{M} p_m = 1, p_m \geq 0 (\forall m)\}$, where M denotes the number of codebook vectors. Consequently, differentiable mappings (e.g., softmax) become available and can be incorporated into neural networks.

To successfully approximate onehot code-selection vectors, smoothed vectors must be distributed around the vertices of the simplex (orange "o" in Figure 1A). At the same time, they should not concentrate around a few vertices, leaving other codebook entries unused (blue "x"). This latter issue—called *code collapse*—has been identified as a major challenge in vector quantization (Dieleman et al., 2018; Baevski et al., 2020b; Dhariwal et al., 2020; Fifty et al., 2025). A widely adopted

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

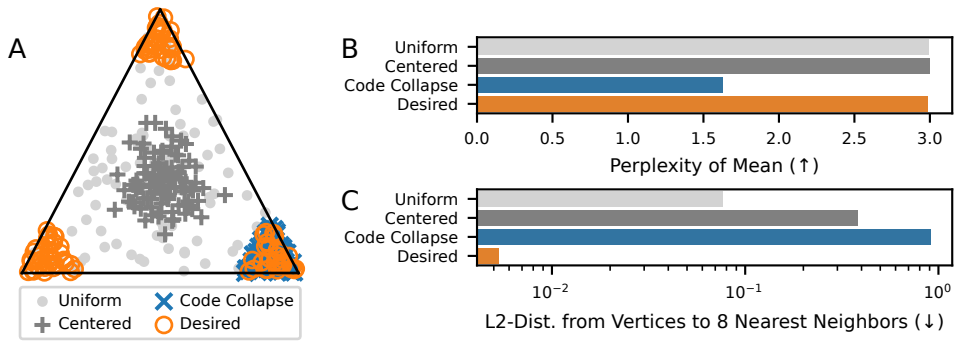


Figure 1: (A) Four different distributions on the simplex Δ^{3-1}. For effective smoothed vector quantization, samples should be concentrated near the vertices of the simplex (i.e., onehot-like vectors; orange), rather than centered (dark gray) or uniformly spread across the simplex (light gray). At the same time, each vertex must have some samples in its neighborhood to avoid code collapse (blue). (B) Maximizing the perplexity of the sample mean (Baevski et al., 2020b) penalizes code collapse but cannot discriminate among the other three distributions. (C) The proposed K-nearest neighbor (KNN) distance minimization ($K = 8$) favors the desired vertex-concentrated distribution while also preventing code collapse.

workaround for code collapse (in smoothed quantization) is to introduce an auxiliary learning objective that maximizes the entropy or perplexity of the *mean* of the smoothed code-selection vectors (Dieleman et al., 2018; Baevski et al., 2020b, see §2.3 for the formal definition). However, maximizing the entropy/perplexity of the mean can be achieved by various distributions, not only the desired vertex-neighboring ones. For example, both uniform and center-concentrated distributions have their mean at the simplex center (Figure 1A), which also maximizes this objective (Figure 1B). Accordingly, an additional mechanism is needed to tighten the smoothing (e.g., by adjusting the temperature parameter of the (Gumbel-)softmax; see §2.2).

Beyond this standard approach, however, there exists a simple and unified strategy for simultaneously tightening smoothed quantization and maximizing codebook usage: *Why not directly encourage clustering around all simplex vertices?* Specifically, minimizing the distance between each simplex vertex and its K-nearest neighbors (KNNs) satisfies both desiderata at once (Figure 1C). The present study investigates this intuitive yet underexplored approach, comparing it against existing alternatives on representative benchmarks. The results indicate that the proposed method enables the exhaustive usage of the entire codebook, even when other approaches suffer from code collapse.

The remainder of this paper is organized as follows. §2 reviews related studies on vector quantization. §3 introduces the proposed method, which is then evaluated on representative benchmarks in §4. Finally, §5 discusses the results and the limitations of the proposed method.

2 Related Studies

The central challenge of vector quantization lies in its non-differentiability, which disrupts the backpropagation of gradients in neural networks. To address this, either the gradient computation (backward pass) or the quantization itself (forward pass) must be approximated.

2.1 Approximation in the Backward Pass (Gradient Estimation)

One line of work retains the original non-differentiable quantization in the forward pass but replaces the gradient computation in the backward pass. Let $\mathbf{z} \in \mathbf{R}^D$ denote a pre-quantized feature vector, and $\{\mathbf{q}_1, \ldots, \mathbf{q}_M\} \subset \mathbf{R}^D$ the set of quantized vectors. Quantization maps $\mathbf{z}$ to its "closest" codebook entry according to a distance metric $\mathcal{D}$: i.e., $\mathbf{z} \mapsto \mathbf{q}_{\iota(\mathbf{z})}$ where $\iota(\mathbf{z}) := \operatorname{argmin}_m \mathcal{D}(\mathbf{z}, \mathbf{q}_m)$. Consequently, the partial derivatives $\frac{\partial q_{\iota(\cdot),i}}{\partial z_j}$ are ill-defined and must be approximated.

A canonical approximation—known as the *straight-through estimation* (STE)—replaces the ill-defined Jacobian with the identity matrix (van den Oord et al., 2017):

$$\frac{\partial q_{\iota(\cdot),i}}{\partial z_j} \approx \begin{cases} 1 & i = j \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Algorithmically, the STE is implemented using the `detach` operation, which excludes its argument from gradient computation:

$$\mathrm{STE}(\mathbf{q}_{\iota(\mathbf{z})}, \mathbf{z}) := \mathbf{q}_{\iota(\mathbf{z})} + \mathbf{z} - \mathtt{detach}(\mathbf{z}) \tag{2}$$

Here, $z - \mathtt{detach}(z)$ evaluates to zero, while its gradient with respect to z is the identity.

More recently, Fifty et al. (2025) proposed an alternative gradient approximation and demonstrated its empirical superiority over the STE:

$$\mathrm{RE}(\mathbf{q}_{\iota(\mathbf{z})}, \mathbf{z}) := \mathtt{detach}\left(\frac{\|\mathbf{q}_{\iota(\mathbf{z})}\|}{\|\mathbf{z}\|} R\right) \mathbf{z} \tag{3}$$

where R is the rotation matrix aligning $\mathbf{z}$ to $\mathbf{q}_{\iota(\mathbf{z})}$,[1] and $\frac{\|\mathbf{q}_{\iota(\mathbf{z})}\|}{\|\mathbf{z}\|}$ rescales the rotated vector to match the amplitude of $\mathbf{q}_{\iota(\mathbf{z})}$. In this formulation, the Jacobian of the quantization is approximated by the scaled rotation matrix:

$$\frac{\partial q_{\iota(\cdot),i}}{\partial z_j} \approx \frac{\|\mathbf{q}_{\iota(\mathbf{z})}\|}{\|\mathbf{z}\|} R \tag{4}$$

2.2 Approximation in the Forward Pass (Smoothing)

An alternative approach approximates the forward quantization itself. Using the onehot representation $\mathbf{e}_m$ of the code index m, quantization can be expressed as:

$$\mathbf{q}_{\iota(\mathbf{z})} = Q\mathbf{e}_{\iota(\mathbf{z})} \tag{5}$$

where $Q := (\mathbf{q}_1, \ldots, \mathbf{q}_M)$. *Smoothed* quantization extends the possible range of $\mathbf{e}_m$ to the simplex Δ^{M-1}. As noted in §1, effective learning requires smoothed code-selectors $\mathbf{p} \in \Delta^{M-1}$ to concentrate near the vertices of the simplex (i.e., $\mathbf{p} \approx \mathbf{e}_m$ for some m).

A widely studied instance of smoothed quantization is Gumbel-softmax sampling (Jang et al., 2017). Given assignment probabilities π_m of $\mathbf{z}$ to the m-th code—typically log-proportional to their dot-product $\mathbf{q}_m^\mathsf{T}\mathbf{z}$—categorical sampling can be implemented using Gumbel samples $g_m = -\log(-\log u_m)$ with $u_m \sim \mathrm{Uniform}(0, 1)$:

$$\begin{aligned} &\iota(\mathbf{z}) \sim \mathrm{Categorical}(\pi_1, \ldots, \pi_M) \\ \Leftrightarrow &\iota(\mathbf{z}) = \operatorname*{argmax}_m (g_m + \log \pi_m) \end{aligned} \tag{6}$$

Replacing argmax above with $\mathrm{softmax}$ yields a smoothed quantization:

$$p_m = \frac{\exp\left(\left(g_m + \log \pi_m\right)/\tau\right)}{\sum_{m'} \exp\left(\left(g_{m'} + \log \pi_{m'}\right)/\tau\right)} \tag{7}$$

where lowering the temperature parameter τ produces a tighter approximation of categorical sampling.

The Gumbel-softmax sampling can also be combined with hard quantization using the STE:

$$\mathrm{STE}(\mathbf{e}_{\iota(\mathbf{z})}, \mathbf{p}) = \mathbf{e}_{\iota(\mathbf{z})} + \mathbf{p} - \mathtt{detach}(\mathbf{p}) \tag{8}$$

[1] The rotation matrix is given by $R = I - 2\hat{\mathbf{r}}\hat{\mathbf{r}}^\mathsf{T} + 2\hat{\mathbf{q}}_{\iota(\mathbf{z})}\hat{\mathbf{z}}^\mathsf{T}$, where $\hat{\mathbf{v}} := \mathbf{v}/\|\mathbf{v}\|$ is the L2-normalization of vector $\mathbf{v}$, and $\mathbf{r} := \hat{\mathbf{q}}_{\iota(\mathbf{z})} + \hat{\mathbf{z}}$.

2.3 REGULARIZATION

Beyond approximation strategies, vector quantization also requires auxiliary regularization losses to ensure effective training. For example, the STE alone does not guarantee alignment between pre-quantized features and codebook entries. This alignment is instead fostered by the following regularization loss, $\mathcal{L}_{\text{reg}}$ (van den Oord et al., 2017; Fifty et al., 2025):

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{main}} + \mathcal{L}_{\text{reg}} \tag{9}$$

$$\mathcal{L}_{\text{reg}} = \mathcal{L}_{\text{hard}} := N^{-1} \sum_{i=1}^{N} \left(\beta \underbrace{\|\mathbf{z}_i - \texttt{detach}(\mathbf{q}_{m_i})\|_2^2}_{\text{Commitment Loss}} + \underbrace{\|\texttt{detach}(\mathbf{z}_i) - \mathbf{q}_{m_i}\|_2^2}_{\text{Codebook Loss}} \right) \tag{10}$$

where $\mathcal{L}_{\text{main}}$ is the primary task loss (e.g., L2 regression in autoencoding), and $\beta > 0$ is a weighting hyperparameter. As defined in Equation 10, $\mathcal{L}_{\text{hard}}$ consists of two components. The first term (known as the *commitment loss*) aligns each pre-quantized feature $\mathbf{z}_i$ in a batch ($i = 1, \ldots, N$) with its nearest codebook entry $\mathbf{q}_{m_i}$. The second term (called the *codebook loss*) moves each codebook vector $\mathbf{q}_m$ toward the centroid of its assigned features whose nearest neighbor is $\mathbf{q}_m$ (i.e., $\{\mathbf{z}_i : \operatorname{argmin}_{m'} \|\mathbf{z}_i - \mathbf{q}_{m'}\| = m\}$).

It should be noted, however, that the codebook loss in Equation 10 does not inherently prevent code collapse. Some codebook vectors may never serve as the nearest neighbor of any pre-quantized feature and therefore receive no updates (Zhu et al., 2025). Accordingly, previous studies have resorted to additional workarounds; for example, unused codebook vectors may be reset to the positions of pre-quantized features (Dhariwal et al., 2020). More recently, Zhu et al. (2025) proposed another remedy called ***SimVQ***, which reparameterizes the codebook Q as the product of a randomly frozen matrix $Q' \in \mathbb{R}^{M \times D}$ and a learnable matrix $W \in \mathbb{R}^{D \times D}$ (i.e., $Q = Q'W$). In this formulation, all codebook vectors share learnable parameters with one another ($\mathbf{q}_m = q'_{m,1}\mathbf{w}_1 + \cdots + q'_{m,D}\mathbf{w}_D$), so updates to one vector propagate to the others.

Similarly, smoothed quantization also requires auxiliary regularization to avoid code collapse. A widely adopted option is to maximize the normalized perplexity of the mean assignment probability (Dieleman et al., 2018; Baevski et al., 2020b):

$$\mathcal{L}_{\text{reg}} = \mathcal{L}_{\text{ppl}} := -\frac{\exp\left(-\sum_m \bar{\pi}_m \log \bar{\pi}_m\right)}{M} \tag{11}$$

$$\bar{\pi}_m := N^{-1} \sum_{i=1}^{N} \pi_{i,m} \tag{12}$$

As noted in §1, this perplexity-based regularization does not promote the onehotness of $\boldsymbol{\pi}_i$, failing to distinguish onehot-like samples from uniform or centered ones (Figure 1B). Onehotness can instead be induced by annealing the temperature parameter of the Gumbel-softmax sampling ($\tau \to 0$). However, manually scheduling this annealing is empirically challenging. The next section therefore proposes an alternative regularization loss that automatically encourages onehotness within the framework of gradient-based learning, while simultaneously preventing code collapse.

3 METHODS

As noted in previous sections, ideal smoothed code-selection vectors $\mathbf{p} \in \Delta^{M-1}$ are distributed near the vertices of the simplex (Figure 1A). Moreover, each vertex should have at least some smoothed code-selectors in its neighborhood; otherwise, the quantization suffers from code collapse. A simple way to achieve these objectives simultaneously is to impose a loss penalizing the deviation of the KNNs from each vertex (Figure 1C):[2]

$$\mathcal{L}_{\text{reg}} = (MK)^{-1} \sum_{m=1}^{M} \sum_{k=1}^{K} \mathcal{D}(\mathbf{e}_m, \mathbf{p}^{(m,k)}) \tag{13}$$

where $\mathbf{p}^{(m,k)}$ denotes the k-th nearest neighbor of the simplex vertex (onehot vector) $\mathbf{e}_m$ according to a distance/deviation metric $\mathcal{D}$. Two options for $\mathcal{D}$ are considered in this study: the squared L2 distance, $\|\mathbf{e}_m - \mathbf{p}^{(m,k)}\|_2^2$, and cross-entropy, $-\log p_m^{(m,k)}$.

[2] A possible alternative formalization and its limitations are discussed in Appendix D.

Table 1: Performance of discrete autoencoding on the ImageNet validation set. Reported metrics are codebook usage and reconstruction quality scores: root mean squared error (rMSE), Fréchet Inception Distance (FID), Inception Score (IS), and Structural Similarity Index Measure (SSIM). The proposed method is denoted as "KNN-L2/CE". The baseline methods are abbreviated as follows: "Cos" denotes cosine-distance quantization (contrasted with Euclidean distance, 'Euclid'). "HG" and "SG" denote Hard-Gumbel and Soft-Gumbel, i.e., Gumbel-softmax sampling with and without straight-through hard quantization in the forward pass, respectively. Best scores across all methods are highlighted in boldface, while underlined values indicate the best-performing value of $K/4$.

			Feature Map Size; Codebook Size											
			16×16×32; 1024				64×64×3; 8196				64×64×64; 8196			
	Method	$K/4$	Usage (↑)	rMSE (↓)	FID (↓)	IS (↑)	Usage (↑)	rMSE (↓)	FID (↓)	IS (↑)	Usage (↑)	rMSE (↓)	FID (↓)	IS (↑)
STE	Euclid	—	4.5%	0.108	101.84	10.67	**100.0%**	**0.045**	**10.41**	**142.12**	1.7%	0.067	25.65	82.61
STE	Cos	—	3.0%	0.102	108.09	10.03	70.9%	0.053	18.13	107.38	1.7%	0.054	20.08	101.74
STE	SimVQ	—	**100.0%**	**0.091**	86.94	15.93	**100.0%**	0.046	10.82	141.17	**100.0%**	0.045	12.54	134.84
RE	Euclid	—	3.1%	0.123	127.40	7.40	78.8%	0.046	14.04	123.30	0.2%	0.077	42.12	51.86
RE	Cos	—	2.8%	0.113	130.47	7.69	99.5%	0.052	19.49	102.55	2.9%	0.050	20.11	101.75
HG	PPL	—	**100.0%**	0.094	94.81	12.26	**100.0%**	0.051	24.41	87.22	**100.0%**	**0.044**	14.77	124.10
HG	**KNN-CE**	1	**100.0%**	0.099	77.21	18.41	**100.0%**	0.061	18.26	111.14	**100.0%**	0.045	4.56	179.47
SG	PPL	—	55.0%	0.224	161.12	3.76	**100.0%**	0.049	14.34	121.54	49.8%	0.320	292.60	1.85
SG	**KNN-CE**	1	**100.0%**	0.153	161.97	6.16	**100.0%**	0.063	19.24	106.64	**100.0%**	0.048	4.62	180.80
Softmax	PPL	—	**100.0%**	0.250	260.30	2.75	81.6%	0.164	57.66	49.21	98.4%	0.149	149.05	13.71
Softmax	**KNN-L2**	1	**100.0%**	0.096	74.53	22.33	**100.0%**	0.060	16.16	119.58	**100.0%**	0.092	14.38	117.34
Softmax	**KNN-L2**	2	**100.0%**	0.098	75.07	20.38	**100.0%**	0.061	15.78	116.06	**100.0%**	0.070	7.91	151.99
Softmax	**KNN-L2**	4	**100.0%**	0.102	78.16	18.28	**100.0%**	0.055	13.15	132.91	**100.0%**	0.051	5.96	168.61
Softmax	**KNN-L2**	8	99.9%	0.092	**71.81**	**22.71**	**100.0%**	0.054	16.88	114.00	**100.0%**	0.056	8.71	152.68
Softmax	**KNN-CE**	1	**100.0%**	0.098	75.16	21.10	**100.0%**	0.060	18.59	110.88	**100.0%**	**0.044**	**2.86**	**194.38**
Softmax	**KNN-CE**	2	**100.0%**	0.102	95.95	13.69	**100.0%**	0.061	19.71	107.15	**100.0%**	0.046	3.55	187.71
Softmax	**KNN-CE**	4	**100.0%**	0.200	227.82	3.28	**100.0%**	0.061	21.58	99.60	**100.0%**	0.050	6.12	166.37
Softmax	**KNN-CE**	8	**100.0%**	0.327	260.67	1.96	**100.0%**	0.059	18.92	107.23	**100.0%**	0.053	6.37	162.22

The proposed regularization no longer requires Gumbel-softmax sampling; smoothed code-selectors can be obtained directly as $\mathbf{p} = \boldsymbol{\pi} = \mathrm{softmax}(Q^{\mathsf{T}}\mathbf{z})$.[3] At the same time, the proposed regularization is fully compatible with Gumbel-softmax sampling; one can simply replace $\mathbf{p}$ (Gumbel-softmax samples) in Equation 13 with $\boldsymbol{\pi}$ (assignment probabilities).

During inference, hard quantization is applied by taking $\mathrm{argmax}_m\, p_m$ in the onehot representation. In the following section, both the deterministic and stochastic approaches are evaluated on representative benchmarks.

4 Experiments

The proposed regularization for smoothed vector quantization was benchmarked on representative tasks of discrete autoencoding (§4.1) and more sophisticated adversarial training (VQGAN; §4.2).

4.1 Discrete Autoencoding

The first benchmark assessed the proposed regularization in the context of discrete autoencoding on the ImageNet dataset (Deng et al., 2009). Input images were convolutionally encoded into latent feature maps, whose pixels were subsequently quantized (Esser et al., 2021; Fifty et al., 2025). A decoder convolutional network reconstructed the input images from these quantized feature maps, and the entire model was trained to minimize the L2 reconstruction loss (L_{main} in Equation 9). Further details of the network architecture and training setup are provided in Appendix B.1.

Three feature-map configurations were considered. Two followed the configurations used in previous studies (Esser et al., 2021; Fifty et al., 2025), with $H \times W \times C = 16 \times 16 \times 32$ ($M = 1024$)

[3] The exact implementation of $\mathbf{p}$ and $\boldsymbol{\pi}$ involves normalization and rescaling; see Appendix A.1 for details.

Table 2: Tightness of softmax-based smoothing (without Gumbel sampling), measured by the individual perplexity of smoothed code-selection vectors, $\exp(-\sum_{m=1}^{M} p_m \log p_m)$. Reported values are the 75th, 90th, and 99th percentiles, as well as the maximum, computed across all feature-map pixels in the ImageNet validation set.

		Feature Map Size; Codebook Size											
		16×16×32; 1024				64×64×3; 8192				64×64×64; 8192			
Method	$K/4$	75%	90%	99%	Max	75%	90%	99%	Max	75%	90%	99%	Max
PPL	—	910.40	933.12	963.29	995.91	7559.89	7561.99	7563.10	7565.02	8186.32	8186.40	8186.55	8186.76
KNN-L2	1	1.17	1.59	2.36	6.26	1.00	1.00	1.00	4.00	9.63	18.10	51.04	123.62
	2	1.00	1.08	1.86	27.37	1.00	1.00	1.06	4.65	3.85	6.36	16.53	43.40
	4	1.00	1.06	1.86	62.99	1.00	1.00	1.00	4.61	1.06	1.06	1.06	1.07
	8	1.00	1.10	1.90	120.68	3.33	3.87	5.22	12.03	1.06	1.06	1.06	1.06
KNN-CE	1	1.18	1.60	2.38	6.28	3.96	4.51	5.81	11.96	1.21	1.21	1.21	1.22
	2	1.23	1.72	2.73	8.07	4.04	4.66	6.16	11.77	1.18	1.18	1.18	1.19
	4	1.42	1.78	2.38	5.45	4.16	4.86	6.57	13.04	1.21	1.22	1.22	1.23
	8	2.38	2.80	3.71	7.78	4.46	5.31	7.39	13.81	1.43	1.43	1.44	1.46

and $64 \times 64 \times 3$ ($M = 8196$). For the latter configuration, the channel dimensionality was further increased to $C = 64$, resulting in feature maps of size $64 \times 64 \times 64$ while retaining $M = 8196$.

Each model was trained using four GPUs in parallel. Accordingly, the $K/4$-nearest neighbors of each simplex vertex were computed independently on each GPU, where $K/4 \in \{1, 2, 4, 8\}$.[4] Crucially, performing the K-nearest neighbor search locally on each device avoids expensive multi-GPU communication (such as `all_gather` operations in PyTorch), which would otherwise be required to share assignment probabilities across GPUs. This design makes the proposed regularization highly parallelizable and computationally efficient (see Appendix C for a detailed analysis of wall-clock time).

The weight β on the commitment loss (Equation 10) for STE-based methods (including SimVQ) and rotational gradient estimation was set to 1.0, following Fifty et al. (2025) and Zhu et al. (2025).

Table 1 reports the codebook utilization and reconstruction performance—including root mean squared error (rMSE), Inception Score (IS; Salimans et al., 2016), and Fréchet Inception Distance (FID; Heusel et al., 2017)—for the proposed method ("KNN-L2/CE") and the baselines. Reconstruction quality deteriorated substantially when softmax-smoothed quantization (i.e., without Gumbel randomness) was combined with the perplexity-based regularization ("PPL"; Equation 11). This degradation arises from a mismatch between soft quantization during training and hard quantization at inference; the smoothed code-selectors deviated from the simplex vertices, as reflected by their high individual perplexity, $\exp(-\sum_{m=1}^{M} p_m \log p_m)$ (Table 2). These results support the argument made in the Introduction that perplexity-based regularization alone does not promote tight smoothing.

Perplexity-based regularization was effective only when combined with the Gumbel-softmax sampling, and onehot quantization in the forward pass via STE was necessary to ensure full codebook utilization ("Hard-Gumbel"). Without hard quantization, code collapse was not prevented at higher channel dimensionalities ($C \geq 32$; "Soft-Gumbel").

By contrast, the proposed KNN-based regularization consistently prevented code collapse and achieved near-complete codebook utilization without resorting to Gumbel-softmax sampling. Reconstruction performance was also superior to, or competitive with, the Gumbel-softmax plus perplexity baseline across all feature map sizes. Moreover, eliminating the Gumbel-softmax sampling reduces the runtime as well (see Appendix C).

The choice of deviation metric (L2 distance vs. cross-entropy, CE) did not exhibit a consistent global advantage in reconstruction quality. However, cross-entropy was additionally favorable in terms of computational efficiency. Its performance was highest when using the minimum number of neighbors ($K/4 = 1$) for all the evaluation metrics (underlined entries in Table 1) except rMSE under the

[4]The value of K was upper-bounded at 8×4 by available computational resources; with a maximum batch size of 64, the total number of latent pixels was $64 \times 64 \times 64 = 8 \times 4 \times 8196$, allowing only 8×4 neighbors per vertex of Δ^{8196-1}. This implementation constraint is further discussed in §4.2.

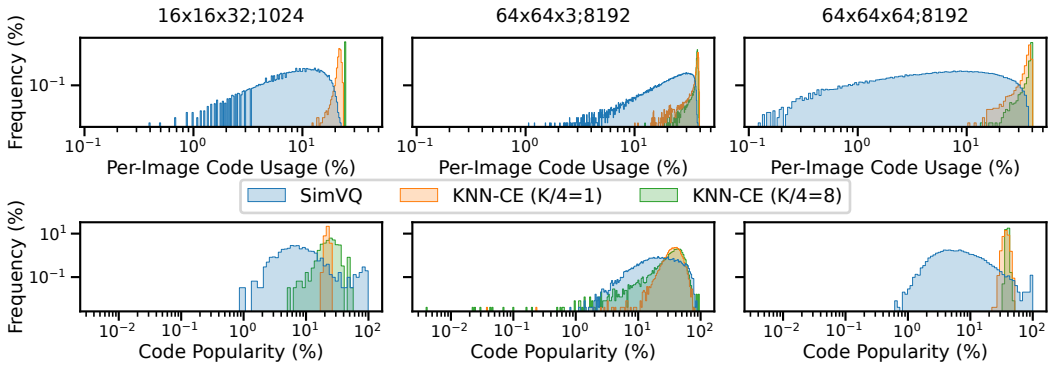


Figure 2: (Top) Histogram of per-image codebook utilization on the validation dataset, defined as the percentage of codes used at least once within each image. (Bottom) Histogram of code popularity, defined as the proportion of validation images in which each code occurs.

$64 \times 64 \times 3$ setting; increasing K yielded no further gains. Although most smoothed code-selectors were formally left unregularized under this setting, they nonetheless achieved low individual perplexity after training (Table 2), indicating a tight approximation to hard quantization. While the exact mathematical mechanism remains an open question, these consistent empirical findings suggest that anchoring only the single closest sample is sufficient to drive the entire distribution toward the simplex vertices. By contrast, minimizing the L2 distance to only a single nearest neighbor per GPU led to performance degradation for feature maps of size $64 \times 64 \times 64$. Although reconstruction performance alone does not favor either metric consistently, this asymmetry in the reliability of small-K behavior matters for scalability, as increasing the number of neighbors requires a larger batch size (see §4.2 for details).

Finally, both STE and rotational gradient estimation (RE) exhibited severe code collapse at higher channel dimensionalities ($C \geq 32$), reaffirming prior findings that reducing channel dimensionality is critical for stable training (Yu et al., 2022a; 2024). SimVQ was more robust, maintaining full codebook utilization under all settings. Looking at individual images, however, more than 5% of them were encoded by using only 1% or a smaller subset of the codes under the $64 \times 64 \times 64$ condition (Figure 2). This lack of code diversity per image resulted in performance degradation compared to the low-dimensional setting ($64 \times 64 \times 3$), even though the best scores (achieved by the KNN-CE with $K/4 = 1$) improved by increasing the channel dimensionality.

4.2 VQGAN and Product Quantization

A primary limitation of the proposed method is the empirical upper bound on the codebook size M. The KNN-based regularization identifies K nearest smoothed code-selectors ($\mathbf{p}$ or $\boldsymbol{\pi}$) per simplex vertex (on each GPU). Consequently, each batch must contain at least KM latent pixels/frames. This requirement limits the scalability of the codebook size M and, consequently, the expressiveness of the quantized representations under standard VRAM constraints.

The present section introduces a possible workaround for this problem and demonstrate its effectiveness based on an advanced quantization task requiring a larger codebook size than naive autoencoding. Specifically, the proposed method is combined with *product quantization*, which achieves diverse discrete representations by combining multiple codebooks (Jégou et al., 2011; Baevski et al., 2020b). By splitting a codebook matrix of size $M \times D$ into G smaller codebooks of size $M \times D/G$, product quantization can represent M^G distinct quantized vectors. Thus, a huge effective vocabulary (of size M^G) is achievable without increasing the size M of individual codebooks, thereby circumventing the scalability constraint of the proposed method.

To demonstrate the effectiveness of product quantization, the proposed method was evaluated on the VQGAN task (Esser et al., 2021). While the network architecture remains identical to the discrete autoencoder (§4.1), VQGAN uses additional learning objectives besides the pixel-wise error minimization in order to improve the quality of reconstructed images (see Appendix B.2 for details). To fully exploit these advanced objectives, however, increasing the effective codebook size and thereby

Table 3: Performance of VQGAN on the ImageNet validation set. Reported metrics are codebook usage and reconstruction quality scores. Best scores across all methods are highlighted in boldface. For the single-codebook STE/RE Euclid baselines ($G = 1$), scores from the original publications are reported, since attempts to reproduce these results were unsuccessful.[5] All product-quantized configurations ($G = 8$), including these baselines, were newly implemented and evaluated.

Method	Evaluated in	D/G	M^G	Usage % (↑)	FID (↓)	IS (↑)
STE Euclid	Esser et al. (2021)	$256/1$	1024^1	—	7.94	—
		$256/1$	16384^1	—	4.98	—
	This Work	$256/8$	1024^8	0.1–0.2	398.52	1.00
RE Euclid	Fifty et al. (2025)	$256/1$	1024^1	98	4.6	146.5
	This Work	$256/8$	1024^8	0.2–0.8	506.60	1.00
SimVQ	This Work[6]	$32/1$	1024^1	1.0	436.34	1.00
		$128/1$	262144^1	2.3×10^{-5}	389.26	1.00
		$256/8$	1024^8	1.1–2.8	410.60	1.00
Hard-Gumbel PPL	This Work	$32/1$	1024^1	**100.0**	15.84	78.59
		$256/8$	1024^8	**100.0–100.0**	2.17	193.53
Softmax KNN-CE ($K/4 = 1$)	This Work	$32/1$	1024^1	**100.0**	13.46	88.59
		$256/8$	1024^8	**100.0–100.0**	**1.57**	**194.51**

the granularity of discrete latent representations is crucial. Thus, VQGAN is a representative benchmark for evaluating the combination of product quantization and KNN-based regularization.

Following the original study (Esser et al., 2021), quantization was performed on the latent feature map of size $16 \times 16 \times 256$, derived from the original input image of size $256 \times 256 \times 3$. Product quantization was implemented with $G = 8$ groups and a codebook size $M = 1024$, thereby splitting the latent channels into $D/G = 256/8 = 32$ while increasing the effective vocabulary to $M^G = 1024^8$.

Table 3 reports the codebook utilization and reconstruction performance for the proposed method ("KNN-CE") and the baselines. The reconstruction scores reported in the original studies are listed alongside those obtained using the corresponding methods in the present experimental setup.

Just as in the discrete autoencoding, the proposed regularization successfully prevented code collapse, ensuring 100% utilization of the codebook(s). It also outperformed the existing methods when combined with the product quantization and equipped with a huge effective codebook size of 1024^8. Note that product quantization does not require a larger codebook matrix and the number of learnable parameters remains identical to the single codebook setting.

5 Discussions

This study introduced a simple and unified regularization method that simultaneously tightens smoothed vector quantization and promotes effective code utilization. The proposed method successfully prevented code collapse in both discrete autoencoding (§4.1) and VQGAN (§4.2). This robustness is noteworthy, as many prior approaches were effective only in specific settings and remained vulnerable to code collapse in others.

[5] Attempts to replicate STE Euclid results using the official repository yielded highly degraded scores; for example, the FID scores were 172.78 for $M = 1024$ and 349.18 for $M = 16384$. These results are consistent with community-reported issues (e.g., `https://github.com/CompVis/taming-transformers/issues/87`). Furthermore, the pre-trained weights for the RE Euclid model (for replicating Table 2 in Fifty et al., 2025) are unavailable in the official repository (`https://github.com/cfifty/rotation_trick/tree/main`). Due to these reproducibility challenges, the "optimal scores" published in the original papers are reported for the single-codebook baselines.

[6] The original SimVQ paper evaluated the method on 128×128 inputs while keeping a 16×16 latent map (Zhu et al., 2025), which severely reduces the compression rate and makes reconstruction significantly easier. Since those scores are not directly comparable to the standard 256×256 VQGAN benchmark, SimVQ is evaluated here on the same 256×256 pipeline to ensure a rigorous and direct comparison.

The proposed method is geometrically intuitive and straightforward, yet appears to be unaddressed in the existing literature. Research on neural vector quantization has traditionally been rooted in variational autoencoding (Kingma & Welling, 2014), with the primary aim of extending this stochastic framework to discrete variables (Jang et al., 2017; van den Oord et al., 2017). The issue of code collapse was recognized (or documented) later, and workarounds were developed independently of the quantization methods themselves (Dieleman et al., 2018; Baevski et al., 2020b; Dhariwal et al., 2020).

By contrast, the present work reformulates neural vector quantization as a simple smoothing problem: onehot vectors are approximated by elements of the simplex. Within this perspective, concentrating the approximators near the simplex vertices naturally arises as a desirable property. This reformulation, together with the proposed regularization strategy, represents a key conceptual contribution of the study.

The KNN-based regularization can also overcome its limitation in the available codebook size by utilizing product quantization (§4.2). While the maximum size of a single codebook is bounded by that of the batch and latent feature map, combining multiple codebooks can enlarge the effective vocabulary without requiring extra learnable parameters. This workaround broadens the applicability of the proposed method.

AI USE STATEMENT

In this work, we used generative AI tools for none of the tasks with required disclosure. We have not used generative AI tools for "generate synthetic data sets", "help develop theoretical models or conceptual frameworks", "formulate mathematical claims", "propose or refine hypotheses", "design or provide feedback on research methodology or experiments", "implement methods", "assist with translation", "clean and reformat dataset", "support qualitative and thematic data analysis", or "interpret results"; and "provide critical ingredients for proving mathematical claims" and "assist in the writing of proofs" are not applicable to this work. Additionally, we used generative AI tools for [tasks with recommended disclosure] "edit a research paper to improve readability". We have reviewed all AI-assisted work. We reviewed all LLM-proposed revisions to ensure that the final manuscript accurately reflects our original arguments. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work introduces a fundamental method for representation learning. It does not involve human subjects or live data collection. No specific ethical risks or societal harms were identified. The research adheres fully to the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of the results, the complete Python source code used for all experiments is available as part of the supplementary material. Furthermore, comprehensive documentation is provided in the appendix: Appendix A outlines the detailed implementation of the core quantization module, while Appendix B lists all hyperparameters and training configurations. All data preprocessing steps and experimental setups are fully detailed in these sections to facilitate reproducibility of the reported experiments.

ACKNOWLEDGMENTS

Hidden for double-blind review.

REFERENCES

Alexei Baevski, Steffen Schneider, and Michael Auli. vq-wav2vec: Self-supervised learning of discrete speech representations. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2020a.

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 12449–12460. Curran Associates, Inc., 2020b.

Sanyuan Chen, Chengyi Wang, Yu Wu, Ziqiang Zhang, Long Zhou, Shujie Liu, Zhuo Chen, Yanqing Liu, Huaming Wang, Jinyu Li, Lei He, Sheng Zhao, and Furu Wei. Neural codec language models are zero-shot text to speech synthesizers. *IEEE Transactions on Audio, Speech and Language Processing*, 33:705–718, 2025. doi: 10.1109/TASLPRO.2025.3530270.

François Chollet. Xception: Deep learning with depthwise separable convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1800–1807, 2017. doi: 10.1109/CVPR.2017.195.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

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

Sander Dieleman, Aaron van den Oord, and Karen Simonyan. The challenge of realistic music generation: modelling raw audio at scale. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming Transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12868–12878, Los Alamitos, CA, USA, jun 2021. IEEE Computer Society. doi: 10.1109/CVPR46437.2021.01268.

Christopher Fifty, Ronald Guenther Junkins, Dennis Duan, Aniketh Iyengar, Jerry Weihong Liu, Ehsan Amid, Sebastian Thrun, and Christopher Ré. Restructuring vector quantization with the rotation trick. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

V. Fomin, J. Anmol, S. Desroziers, J. Kriss, and A. Tejani. High-level library to help with training neural networks in pytorch. `https://github.com/pytorch/ignite`, 2020.

Robert Gray. Vector quantization. *IEEE ASSP Magazine*, 1(2):4–29, 1984. doi: 10.1109/MASSP.1984.1162229.

Gaëtan Hadjeres and Léopold Crestel. Vector quantized contrastive predictive coding for template-based music generation, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, June 2016. doi: 10.1109/CVPR.2016.90.

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 I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5967–5976, 2017. doi: 10.1109/CVPR.2017.632.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with Gumbel-softmax. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

Herve Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011. doi: 10.1109/TPAMI.2010.57.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (eds.), *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 694–711, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46475-6.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. The International Conference on Learning Representations (ICLR) 2014, 2014.

Fernando Pérez-Cruz. Kullback-leibler divergence estimation of continuous distributions. In *2008 IEEE International Symposium on Information Theory*, pp. 1666–1670, 2008. doi: 10.1109/ISIT.2008.4595271.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8821–8831. PMLR, 18–24 Jul 2021.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10674–10685, 2022. doi: 10.1109/CVPR52688.2022.01042.

G. Ronning. Maximum likelihood estimation of dirichlet distributions. *Journal of Statistical Computation and Simulation*, 32(4):215–221, 1989. doi: 10.1080/00949658908811178.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training GANs. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Max Sklar. Fast MLE computation for the dirichlet multinomial, 2014.

Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 30*, pp. 6306–6315. Curran Associates, Inc., 2017.

Nicolas Wicker, Jean Muller, Ravi Kiran Reddy Kalathur, and Olivier Poch. A maximum likelihood approximation method for dirichlet's parameter estimation. *Computational Statistics & Data Analysis*, 52(3):1315–1322, 2008. ISSN 0167-9473. doi: 10.1016/j.csda.2007.07.011.

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 *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022a.

Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, Ben Hutchinson, Wei Han, Zarana Parekh, Xin Li, Han Zhang, Jason Baldridge, and Yonghui Wu. Scaling autoregressive models for content-rich text-to-image generation. *Transactions on Machine Learning Research*, 2022b. ISSN 2835-8856. doi: 10.48550/ARXIV.2206.10789.

Lijun Yu, José Lezama, Nitesh Bharadwaj Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Agrim Gupta, Xiuye Gu, Alexander G. Hauptmann, Boqing Gong, Ming-Hsuan Yang, Irfan Essa, David A. Ross, and Lu Jiang. Language model beats diffusion - tokenizer is key to visual generation. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 586–595, 2018. doi: 10.1109/CVPR.2018.00068.

Ziqiang Zhang, Long Zhou, Chengyi Wang, Sanyuan Chen, Yu Wu, Shujie Liu, Zhuo Chen, Yanqing Liu, Huaming Wang, Jinyu Li, Lei He, Sheng Zhao, and Furu Wei. Speak foreign languages with your own voice: Cross-lingual neural codec language modeling. arXiv, 2023.

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

A Implementation of the Quantization Methods

This section provides details on the implementation of the quantization methods.

A.1 Smoothed Quantization

Smoothed code-selection vectors $\mathbf{p}$ were computed as $\mathbf{p} = \mathrm{softmax}(\hat{Q}^{\mathsf{T}}\hat{\mathbf{z}}/\mathrm{t})$. In other words, the codebook vectors $(\mathbf{q}_1, \ldots, \mathbf{q}_M) = Q$ and the feature vectors $\mathbf{z}$ were first L2-normalized, and their product (i.e., cosine similarity) was scaled by a learnable temperature t. This temperature was shared across the codebook so that all the logits had the same amplitude. Assignment probabilities π for Gumbel-softmax sampling were computed in the same way, while the additional temperature parameter—τ in Equation 7—was fixed as 1.0.[7]

A.2 Hard Quantization

The weight β on the commitment loss in Equation 10 was set to 1.0 (Fifty et al., 2025; Zhu et al., 2025), based on previous observations that its value does not significantly affect learning outcomes within the range 0.1–2.0. (van den Oord et al., 2017).

B Details of the Experiments

B.1 Discrete Autoencoding

This section provides implementation details for the autoencoding experiment described in §4.1.

The network architecture followed prior work on discrete autoencoding of ImageNet (Esser et al., 2021; Fifty et al., 2025). Input images were center-cropped to $H \times W \times C = 256 \times 256 \times 3$. The encoder first expanded the channel dimensionality of the input images from 3 to 256 by convolution, and then progressively downsampled them through a series of strided convolutions (see Table 4 for the spatial and channel sizes at each layer). Each downsampling layer was followed by a residual block (He et al., 2016). The decoder reconstructed the input images by upsampling the latent feature maps with a sequence of interpolations and residual blocks.

To improve memory efficiency, all but the input and output layers were implemented as depthwise separable convolutions (Chollet, 2017). All convolutional kernels had size 3×3.

Training employed the AdamW optimizer with $(\beta_1, \beta_2) = (0.9, 0.99)$ and a weight decay coefficient of 10^{-4}, except for Euclidean-based STE/rotational hard quantization, where weight decay was set to zero.

The learning rate was linearly warmed up from 0.0 to ρ_{max}, and subsequently annealed to $0.5\rho_{\text{max}}$ by cosine scheduling. For discrete autoencoding, the maximum learning rate ρ_{max} was set to 5×10^{-5} for Euclidean-based STE/rotational hard quantization, and to 10^{-4} for all other configurations (Fifty et al., 2025).

[7] Previous studies manually annealed the Gumbel-softmax temperature from $\tau = 0.5$ to 2.0, scaling it by 0.999995 at each iteration (Baevski et al., 2020b). This approach was also tested in the experiments here but did not yield improvements over the fixed temperature.

Table 4: Hyperparameters for discrete autoencoding/GAN.

	GAN (§4.2)	Autoencoding (§4.1)		
Feature Map Size	$16 \times 16 \times 32$	$16 \times 16 \times 32$	$64 \times 64 \times 3$	$64 \times 64 \times 64$
Codebook Size	1024	8196	8196	8196
Latent Channels	$128 \to 128 \to 64 \to 64 \to 32 \to 32$	$128 \to 64 \to 32 \to \{3, 64\}$		
Height & Width	$256 \to 128 \to 64 \to 32 \to 16 \to 16$	$256 \to 128 \to 64 \to 64$		
Batch Size	64	64	64	64
Training Epochs	60	25	20	20
Warmup Iterations	20,000	16,000	16,000	16,000

B.2 VQGAN

In the VQGAN experiment (§4.2), the models were optimized based on the following learning objective:

$$\mathcal{L}_{\text{main}} = \underbrace{|x - \mathfrak{F}(x)|_1}_{\text{Reconstruction}} + \overbrace{\sum_l \|\mathfrak{P}_l(x) - \mathfrak{P}_l \circ \mathfrak{F}(x)\|_2^2}^{\text{Perceptual}} \underbrace{-\lambda \mathfrak{G} \circ \mathfrak{F}(x)}_{\text{Adversarial}} \tag{14}$$

$\mathfrak{F}$ denotes the model network reconstructing original images x through discrete latent representations. $\mathfrak{P}$ is a perceptual-loss function implemented by the VGG-16 network pretrained on image classification and finetuned to predict image similarity judgements by human evaluators (Johnson et al., 2016; Zhang et al., 2018). $\mathfrak{G}$ is an adversarial discriminator network (PatchGAN implementation; Isola et al., 2017), which was trained with the hinge loss Equation 15.

$$\mathcal{L}_{\text{disc}} := \max\{0, 1 - \mathfrak{G}(x)\} + \max\{0, 1 + \mathfrak{G} \circ \mathfrak{F}(x)\} \tag{15}$$

The weight λ on the adversarial loss was set to zero during the initial 600k iterations, and to 0.1 for the rest of training. The maximum learning rate ρ_{max} for the main network was set to 10^{-4}, whereas that for the discriminator was set to 4.5×10^{-6}.

B.3 Evaluation Metrics

Fréchet Inception Distance (FID; Heusel et al., 2017) and Inception Score (IS; Salimans et al., 2016) were evaluated using PyTorch Ignite implementations (Fomin et al., 2020). The weights of their feature extractor network (Inception V3) conformed to the original implementation of FID.

C Computational Load

To evaluate the computational load of the proposed method, the wall-clock time and peak VRAM consumption per training iteration were measured during the discrete autoencoding task (Table 5). The models were run on four NVIDIA A100 GPUs (80GB VRAM per GPU) using `DistributedDataParallel` of PyTorch. The reported scores represent computational load per GPU on average over a single epoch of the ImageNet training split, with batch size set to 64.

This benchmark results suggest that the proposed regularization method ("KNN-CE") is not computationally expensive compared to existing methods in the (Gumbel-)Softmax family. The execution time is almost identical to the perplexity-based regularization ("PPL"), and is smaller than its combination with the hard Gumbel-Softmax sampling. Similarly, the VRAM consumption is identical across the smoothing approaches, indicating that the major source of memory load is in the softmax computation, not in the regularization.

D Alternative Implementations of the Intended Regularization

Alternative regularization strategies could also achieve the intended distribution of smoothed code-selection vectors ($\mathbf{p}$ or $\boldsymbol{\pi}$) around the simplex vertices. For example, one could align smoothed

Table 5: Wall-clock time and peak VRAM consumption per training iteration. The models were run on four NVIDIA A100 GPUs (80GB VRAM per GPU) using `DistributedDataParallel` of PyTorch. The reported scores represent computational load per GPU on average over a single epoch of the ImageNet training split, with batch size set to 64. The "Entire" wall-clock time includes the forward and backward passes through the encoder, quantizer, and decoder modules, while the 'Quant.' reports the wall-clock time consumed by the forward pass through the quantizer alone.

			Feature Map Size; Codebook Size								
			$16\times16\times32$; 1024			$64\times64\times3$; 8196			$64\times64\times64$; 8196		
			Wall-Clock Time			Wall-Clock Time			Wall-Clock Time		
Method		$K/4$	Entire (ms)	Quant. (ms)	VRAM (GB)	Entire (ms)	Quant. (ms)	VRAM (GB)	Entire (ms)	Quant. (ms)	VRAM (GB)
STE	STE Euclid	—	330.39	1.35	34.20	508.49	1.35	54.83	502.20	1.30	54.93
	SimVQ	—	330.18	1.31	34.20	504.28	1.36	54.83	502.25	1.37	54.92
Hard-Gumbel PPL		—	324.95	1.94	34.24	615.49	1.89	59.83	614.21	1.92	59.89
Softmax	No Regularization	—	323.48	1.40	34.24	561.05	1.44	59.83	557.22	1.38	59.87
	PPL	—	324.99	1.68	34.24	601.19	1.64	59.83	598.18	1.57	59.87
	KNN-CE	1	324.92	1.55	34.24	596.11	1.58	59.83	592.39	1.58	59.87
		8	325.60	1.50	34.24	595.86	1.64	59.83	592.85	1.54	59.87

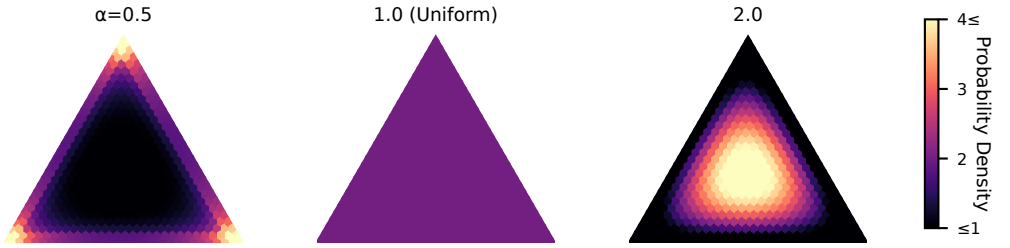


Figure 3: Dirichlet distributions on the simplex Δ^{3-1} with concentration parameters $\alpha_1 = \alpha_2 = \alpha_3 = \alpha$, where $\alpha \in \{0.5, 1.0, 2.0\}$.

code-selectors with a Dirichlet distribution with concentration parameters $\alpha_1 = \cdots = \alpha_K < 1.0$ (Figure 3). The probability density of such a distribution is highest at the vertices of the simplex, thus matching the ideal distribution of smoothed code-selectors. A possible formalization of this alignment is based on the Kullback-Leibler (KL) divergence between the Dirichlet prior ($\mathbb{P}$) and the distribution inferred from smoothed code-selectors ($\mathbb{Q}$).

$$\mathcal{D}_{\mathrm{KL}}\left(\mathbb{P} \mid \mathbb{Q}\right) := \int_{\Delta^{M-1}} \mathbb{P}(\mathbf{p}) \log \frac{\mathbb{P}(\mathbf{p})}{\mathbb{Q}(\mathbf{p})} d\mathbf{p} \tag{16}$$

However, Equation 16 is difficult to use directly as a regularization loss, due to the complexity of estimating $\mathbb{Q}$ from sample code-selectors, $\mathbf{p}$. Although one could constrain $\mathbb{Q}$ as another Dirichlet to make the KL divergence tractable, maximum likelihood estimation of its parameters requires iterative algorithms (e.g., the Newton-Raphson method; Ronning, 1989; Sklar, 2014; Wicker et al., 2008), complicating and slowing gradient-based optimization in deep learning frameworks. Moreover, this estimation involves digamma and trigamma functions (Sklar, 2014), whose derivatives can explode when the concentration parameters approach small values—as desired for tight smoothing—during the course of learning.

A more practical approach is to approximate Equation 16 itself in a tractable manner. For instance, Pérez-Cruz (2008) proposed a KNN-based estimation of the KL divergence that relies solely on *samples* from the two distributions, using the k-th nearest neighbor from $\mathbb{Q}$ for each sample from $\mathbb{P}$. The regularization method proposed in this study can thus be interpreted as minimizing this estimated KL divergence, with the samples from $\mathbb{P}$ constrained to onehot vectors.