ATTENTIVE VECTOR QUANTIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Discrete latent representations are widely used in generative and representation learning, but conventional vector quantization commonly relies on hard nearest-neighbor assignment with a straight-through estimator (STE), introducing a mismatch between forward computation and backward gradients. Learned vector quantizers can also exhibit highly non-uniform code usage or codebook collapse. We introduce Attentive Vector Quantization (AVQ), an STE-free differentiable quantization approach that replaces nearest-neighbor assignment during training with temperature-controlled cosine-similarity attention between latent tokens and a learned codebook. AVQ combines a soft-to-hard training curriculum, a compact codebook-usage regularizer that encourages confident local assignments and balanced global utilization, and hard-code decoder calibration to bridge differentiable training and discrete inference. We evaluate AVQ against conventional vector quantization (VQ), EMA-based vector quantization (VQ-EMA), and finite scalar quantization (FSQ) on COCO, Tiny ImageNet, and MiniPlaces. Across all three datasets, AVQ achieves complete codebook utilization with near-maximal hard-code perplexity and the lowest rFID among the evaluated single-stage quantizers. On COCO, across three model seeds, AVQ uses all 1024 codes with a perplexity of 1000.5 ± 3.5 and achieves an rFID of 41.5 ± 1.4, an 18.0% reduction relative to the best baseline mean. Similar improvements in rFID are observed on MiniPlaces and Tiny ImageNet, while AVQ maintains complete utilization and perplexity above 1000 on all three datasets. Across all single-stage AVQ ablations, complete codebook utilization persists under changes in codebook size, embedding dimension, normalization, learned code scaling, and spatial rate. We further extend AVQ to Attentive Residual Vector Quantization (ARVQ). Across all three datasets, ARVQ maintains near-maximal stage-wise perplexity above 1016 at every residual stage, whereas EMA-based residual vector quantization (RVQ-EMA) exhibits substantially lower and less consistent stage-wise perplexity. These results indicate that attentive quantization provides robust use of discrete capacity in both single-stage and residual quantization settings.

1 INTRODUCTION

Vector quantization maps continuous encoder features to a finite learned codebook, producing discrete latent representations that can be stored, transmitted, or modeled by downstream systems. Vector-quantized representations are widely used in image generation (van den Oord et al., 2017; Razavi et al., 2019; Esser et al., 2021) and have been adopted in speech, video, and multimodal modeling (Yan et al., 2021; Baevski et al., 2020; Zeghidour et al., 2021; Mousavi et al., 2025). However, conventional vector quantization relies on non-differentiable hard nearest-neighbor assignment and typically uses the straight-through estimator (STE), creating a mismatch between the discrete forward operation and the surrogate gradient used during backpropagation.

Differentiable soft assignment provides exact gradients through the quantizer, but introduces a different mismatch: the decoder is trained on convex combinations of codebook vectors while inference uses discrete codes. Sharpening the assignment distribution reduces this soft–hard gap by making soft assignments more discrete-like, but does not by itself ensure broad and balanced codebook usage. To address these challenges, we develop Attentive Vector Quantization (AVQ), an STE-free framework for learning discrete representations while explicitly managing the transition from differentiable training to hard inference. We further extend the formulation to residual quantization.

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.

The main contributions are:

- We introduce AVQ[1], an STE-free differentiable quantizer that replaces hard nearest-neighbor assignment during optimization with temperature-controlled codebook attention and uses a soft-to-hard curriculum with hard-code decoder calibration to bridge differentiable training and discrete inference.
- We introduce a compact utilization regularizer that encourages locally confident assignments while maintaining balanced global code usage, yielding high effective codebook capacity.
- We extend AVQ to residual quantization and evaluate AVQ/ARVQ on COCO, Tiny ImageNet, and MiniPlaces against VQ, VQ-EMA, FSQ, and RVQ-EMA using reconstruction, perceptual-quality, and codebook-usage metrics.

2 Related Work

Differentiable and STE-Free Quantization: Differentiable quantization methods replace or modify hard code assignment using convex-optimization-based procedures (Gautam et al., 2024), alternative gradient estimators, or reparameterization-based formulations (Vali et al., 2026). AVQ instead formulates code selection directly as temperature-controlled cosine attention between encoder features and a learned codebook, yielding differentiable soft quantization while a soft-to-hard curriculum progressively introduces discrete assignments without propagating surrogate gradients through hard selection.

Vector Quantization and Attention: Prior work has combined VQ with attention in different roles. Yang et al. (2023) quantize visual features before self-attention for quality-robust representation learning, while Rasul et al. (2024) quantize sequence representations within attention for efficient long-context forecasting. van den Dool et al. (2026) dynamically refine a hierarchical VQ codebook using attention mass to improve transformer efficiency. In contrast, AVQ uses attention as the quantization operator itself: encoder tokens directly attend to codebook vectors, and the resulting distribution defines the soft quantized representation.

Codebook Utilization and Collapse: Prior approaches use interventions such as dead-code replacement (Dhariwal et al., 2020; Kolesnikov et al., 2022), k-means initialization (Zeghidour et al., 2021), or factorized and ℓ_2-normalized codebooks (Yu et al., 2022). More recent methods modify the training objective or quantization space: Zhang et al. (2023) use prior-distribution regularization to encourage balanced code usage, while Lookup-Free Quantization (LFQ) (Yu et al., 2024) removes learned distance-based codebooks and represents discrete latents on a binary hypercube with entropy regularization. FSQ similarly avoids a learned vector codebook by quantizing low-dimensional latents using fixed scalar levels (Mentzer et al., 2024). AVQ instead retains a learnable vector codebook and combines a local entropy term for confident assignments with global balance regularization for distributed code usage. We characterize codebook usage using hard-code perplexity and codebook utilization alongside reconstruction quality.

3 Attentive Vector Quantization

Given an image $\mathbf{x} \in [-1, 1]^{3\times h\times w}$, an encoder Enc_θ produces a latent feature map $\mathbf{z} = \mathrm{Enc}_\theta(\mathbf{x}) \in \mathbb{R}^{D\times H\times W}$. We flatten the spatial dimensions into $N = HW$ latent tokens $\mathbf{Z} \in \mathbb{R}^{N\times D}$ and learn a codebook $\mathbf{E} \in \mathbb{R}^{K\times D}$ containing K embeddings of dimension D.

[1]The acronym AVQ has previously been used for adaptive vector quantization in classical coding (Shen et al., 2003) and, more recently, for adaptive codebook scaling in Transformer attention kernels (van den Dool et al., 2026). Throughout this work, AVQ refers specifically to Attentive Vector Quantization.

3.1 Differentiable Vector Quantization

We ℓ_2-normalize the latent tokens and codebook vectors, yielding $\tilde{\mathbf{Z}}$ and $\tilde{\mathbf{E}}$, and compute temperature-controlled cosine attention over the codebook:

$$\mathbf{P}_t = \mathrm{softmax}\left(\frac{\tilde{\mathbf{Z}}\tilde{\mathbf{E}}^\top}{\tau_t}\right), \qquad \mathbf{Q}^{\mathrm{soft}} = \mathbf{P}_t\tilde{\mathbf{E}}, \tag{1}$$

where the softmax is applied over the K codebook entries and τ_t is the temperature at training step t. This operation has the structure of cross-attention: normalized latent tokens act as queries and normalized codebook vectors act as both keys and values, without learned query, key, or value projections. Larger τ_t produces diffuse assignments, while decreasing τ_t progressively sharpens them. Because the assignment probabilities and weighted codebook representation are differentiable, reconstruction gradients propagate to both the encoder and codebook without an STE.

For hard quantization, each token is assigned to its maximum-probability codeword $k_n^* = \arg\max_k p_{nk}$, where $p_{nk} = [\mathbf{P}_t]_{nk}$; equivalently, this selects the maximum-cosine-similarity codeword. The resulting hard representation is $\mathbf{q}_n^{\mathrm{hard}} = \tilde{\mathbf{e}}_{k_n^*}$, producing the discrete sequence $(k_1^*, \ldots, k_N^*)$. Assuming a unique maximum, as $\tau_t \to 0^+$, the assignment distribution converges to a one-hot vector and $\mathbf{q}_n^{\mathrm{soft}} \to \mathbf{q}_n^{\mathrm{hard}}$, motivating progressive temperature annealing.

3.2 Soft-to-Hard Curriculum and Decoder Calibration

During training, the decoder receives a convex combination of the differentiable soft representation and detached hard codes:

$$\mathbf{q}_n^{\mathrm{train}} = \alpha_t\mathbf{q}_n^{\mathrm{soft}} + (1-\alpha_t)\,\mathrm{sg}\big(\mathbf{q}_n^{\mathrm{hard}}\big), \qquad \frac{\partial \mathbf{q}_n^{\mathrm{train}}}{\partial \mathbf{q}_n^{\mathrm{soft}}} = \alpha_t\mathbf{I}, \tag{2}$$

where $\mathrm{sg}(\cdot)$ denotes stop-gradient and $\alpha_t \in [0,1]$ controls the soft contribution. No gradient propagates through the discrete $\arg\max$ branch; AVQ therefore requires no straight-through estimator.

We initially set $\alpha_t = 1$ and cosine-anneal it to a positive endpoint $\alpha_{\min}$, progressively exposing the decoder to hard representations. In parallel, τ_t is cosine-annealed from $\tau_{\max}$ to $\tau_{\min}$ in log-temperature space, progressively sharpening assignments. Exact schedules and hyperparameters are provided in the appendix.

After this phase, we freeze the encoder and codebook and calibrate the decoder using hard quantized representations only, adapting it directly to the discrete representations used at inference and reducing the remaining soft–hard reconstruction gap.

3.3 Training Objectives

A useful quantizer should produce confident token-level assignments while using the codebook broadly. For a batch containing $M = BN$ tokens, let p_{mk} denote the soft assignment probability of token m to code k. We minimize the normalized local entropy

$$\mathcal{L}_{\mathrm{local}} = -\frac{1}{M\log K}\sum_{m=1}^{M}\sum_{k=1}^{K} p_{mk}\log(p_{mk}+\epsilon), \tag{3}$$

where $\epsilon > 0$ is a small constant for numerical stability. Minimizing this term encourages confident, low-entropy assignments.

To make the global usage statistic more representative of likely hard assignments while retaining differentiability, we sharpen each token distribution using an exponent $\gamma > 1$ and compute its batch-average distribution:

$$\tilde{p}_{mk} = \frac{p_{mk}^{\gamma}}{\sum_{j=1}^{K} p_{mj}^{\gamma}}, \qquad \bar{p}_k = \frac{1}{M}\sum_{m=1}^{M}\tilde{p}_{mk}, \qquad \mathcal{L}_{\mathrm{balance}} = \frac{1}{\log K}\sum_{k=1}^{K}\bar{p}_k\log(K\bar{p}_k). \tag{4}$$

This normalized KL divergence from the uniform distribution encourages broad codebook usage. The resulting codebook objective is $\mathcal{L}_{\mathrm{CB}} = \lambda_{\mathrm{local}}\mathcal{L}_{\mathrm{local}} + \lambda_{\mathrm{balance}}\mathcal{L}_{\mathrm{balance}}$.

Stacking the training tokens gives $\mathbf{Q}^{\text{train}}$, from which the decoder reconstructs $\hat{\mathbf{x}} = \text{Dec}_\phi(\mathbf{Q}^{\text{train}})$. We optimize mean squared reconstruction error together with the codebook objective and, when enabled, an auxiliary hinge adversarial loss:

$$\mathcal{L}_{\text{AE}} = \mathcal{L}_{\text{rec}} + \lambda_{\text{CB}}\mathcal{L}_{\text{CB}} + \lambda_{\text{adv}}\mathcal{L}_{\text{adv}}, \tag{5}$$

where the adversarial term is introduced only after an initial reconstruction-only phase. The discriminator is used only during training to provide an auxiliary adversarial signal for the reconstructed images and is not required at inference. Details of the discriminator objective and training schedule are provided in the appendix.

4 Attentive Residual Vector Quantization

To increase representational capacity, we extend AVQ to Attentive Residual Vector Quantization (ARVQ), where a sequence of S attentive quantizers progressively encodes residual information. Initializing $\mathbf{R}^{(1)} = \mathbf{Z}$, at each stage s the current residual is ℓ_2-normalized to $\tilde{\mathbf{R}}^{(s)}$ and attends to its own codebook $\mathbf{E}^{(s)} \in \mathbb{R}^{K\times D}$, normalized as $\tilde{\mathbf{E}}^{(s)}$. The stage output $\mathbf{Q}^{(s)}$ is computed using the same temperature scaling and convex soft–hard interpolation described in Section 3. The residual recursion and aggregate representation are

$$\mathbf{R}^{(s+1)} = \mathbf{R}^{(s)} - \text{sg}\Big(\mathbf{Q}^{(s)}\Big), \qquad \mathbf{Q} = \sum_{s=1}^{S} \mathbf{Q}^{(s)}. \tag{6}$$

Detaching $\mathbf{Q}^{(s)}$ prevents later-stage losses from propagating through the stage output into earlier-stage quantizer parameters. Each stage independently applies the attentive assignment and convex interpolation, so gradients propagate through its differentiable soft pathway without an STE. The codebook objective is applied independently at each stage and averaged as $\mathcal{L}_{\text{ARVQ}} = \frac{1}{S}\sum_{s=1}^{S}\mathcal{L}_{\text{CB}}^{(s)}$, encouraging confident assignments and broad code usage throughout the residual hierarchy. At inference, only hard assignments are retained, and the decoder receives $\mathbf{Q}^{\text{hard}} = \sum_{s=1}^{S}\mathbf{Q}_{\text{hard}}^{(s)}$.

5 Experiments

5.1 Experimental Setup

Datasets: We evaluate AVQ and ARVQ on Tiny ImageNet (Le & Yang, 2015), COCO 2017 (Lin et al., 2014), and MiniPlaces (Zhou et al., 2017). For Tiny ImageNet, we construct a deterministic class-stratified split of the official training set into 95,000 training and 5,000 validation images, reserving the official 10,000-image validation set for final evaluation. MiniPlaces similarly uses a 5% class-stratified holdout from its training set for checkpoint selection and the official validation split for final evaluation. For COCO 2017, we reserve 5,000 images from `train2017` using multilabel-stratified sampling over object categories and use `val2017` for final evaluation. Images are resized with bicubic interpolation to 64×64 for Tiny ImageNet and COCO and 128×128 for MiniPlaces, normalized to $[-1, 1]$, and augmented during training with random horizontal flipping.

Training and Seeds: Phase I uses Adam with learning rate 1×10^{-4} for the encoder–quantizer–decoder and 4×10^{-4} for the discriminator, with $\beta = (0.5, 0.9)$. Models are trained for 50 epochs and selected by internal-validation hard-code reconstruction MSE. For AVQ and ARVQ, only epochs after completion of the annealing schedule are eligible for Phase-I checkpoint selection; baseline methods are eligible throughout Phase I. Adversarial training is delayed until epoch 5, providing an initial reconstruction-only warm-up before discriminator gradients are introduced. Batch size is 64 for Tiny ImageNet and COCO at 64×64 resolution and 16 for MiniPlaces at 128×128, keeping the number of spatial pixels processed per batch constant across datasets. Phase II uses Adam with learning rate 1×10^{-4} and the same β values for up to 10 epochs with early-stopping patience of 3 epochs; optimization and decoder-calibration settings are summarized in Appendix Table 5. The data seed is fixed to 15 throughout. Main comparisons use model seeds $\{42, 259, 2027\}$ for AVQ, VQ, VQ-EMA, FSQ, ARVQ, and RVQ-EMA; ablations use seed 42 unless otherwise stated.

Hard-Code Decoder Calibration: Hard-code decoder calibration is part of the AVQ/ARVQ training pipeline and bridges the representation mismatch between differentiable Phase-I training and discrete inference. During Phase I, the AVQ/ARVQ decoder receives soft or mixed representations, whereas inference uses hard discrete codes. In Phase II, the encoder and quantizer are frozen and only the decoder is calibrated using hard representations. To ensure that improvements are attributable to the quantization mechanism rather than decoder adaptation, we apply the same decoder-only hard-code calibration protocol to all baseline methods. Full Phase-I checkpoint and Phase-II calibrated results are reported in the Appendix.

Evaluation Metrics: Reconstruction quality is measured using rFID, LPIPS, and MSE, where lower is better. Discrete representation quality is characterized by hard-code perplexity and codebook utilization; higher perplexity and utilization indicate broader effective use of the available codebook.

Baselines: For single-stage quantization, we compare AVQ against STE-based Vector Quantization (VQ), EMA-based Vector Quantization (VQ-EMA), and Finite Scalar Quantization (FSQ). For residual quantization, we compare the proposed ARVQ against Residual Vector Quantization with EMA codebook updates (RVQ-EMA); both use three residual stages with a separate $K = 1024$ codebook at each stage. Within each dataset, all compared methods use the same latent spatial resolution and matched encoder–decoder architecture; spatial resolution is changed only in the designated ablation experiments. Ablations evaluate codebook size K, embedding dimension D, spatial resolution, removal of ℓ_2 normalization, and learned code scaling.

5.2 Results

Single-Stage Quantization: Table 1 compares AVQ with VQ, VQ-EMA, and FSQ across all three datasets. AVQ achieves the lowest rFID on every dataset while maintaining complete codebook utilization and near-maximal hard-code perplexity. On COCO, AVQ attains a perplexity of 1000.5 ± 3.5 with all 1024 codes active and reduces rFID from the best baseline mean of 50.58 to 41.48, an 18.0% reduction. Similarly, AVQ attains perplexities of 1009.8 ± 3.1 on MiniPlaces and 1003.3 ± 5.3 on Tiny ImageNet, again with all 1024 codes active, while reducing rFID from the best baseline means of 21.40 to 18.23 and 31.69 to 28.24, corresponding to reductions of 14.8% and 10.9%, respectively. Conventional VQ exhibits substantial codebook collapse across all datasets, while VQ-EMA achieves substantially broader utilization but remains well below AVQ in hard-code perplexity. FSQ activates all available codes, yet its markedly lower perplexity indicates less balanced effective usage despite complete utilization. AVQ does not minimize pixel-level MSE and is not uniformly best in LPIPS, highlighting that pixel-level distortion, perceptual quality, and codebook usage need not improve together.

Single-Stage AVQ Ablations: Table 2 evaluates the effects of codebook size K, embedding dimension D, latent spatial resolution, removal of ℓ_2 normalization, and learned code scaling across all three datasets. AVQ maintains complete utilization and near-maximal perplexity relative to the available codebook size in every ablation setting. Removing ℓ_2 normalization degrades MSE, LPIPS, and rFID on all three datasets while preserving complete utilization, although its effect on perplexity is mixed. Reducing the codebook size from $K = 1024$ to $K = 512$ also degrades MSE, LPIPS, and rFID across all three datasets while still utilizing the full available codebook; for example, on MiniPlaces, rFID increases from 18.38 to 25.14, a 36.8% increase. Learned code scaling produces mixed effects across datasets and metrics, with no consistent improvement over the default configuration. Increasing the embedding dimension from $D = 64$ to $D = 128$ similarly produces mixed results, improving some metrics on COCO and Tiny ImageNet while degrading others and providing no consistent benefit on MiniPlaces. The most pronounced effect arises from reducing the latent spatial resolution, which substantially degrades MSE, LPIPS, and rFID on all three datasets even though perplexity remains close to the maximum available codebook size. This result shows that broad and balanced code usage alone is not sufficient to guarantee high reconstruction or perceptual quality.

Additional ARVQ ablations are reported in the Appendix, including learned code scaling on COCO, MiniPlaces, and Tiny ImageNet and reduced spatial resolution on MiniPlaces. The spatial-rate ablation is restricted to MiniPlaces because the default COCO and Tiny ImageNet ARVQ configurations already use a 16×16 latent grid.

Table 1: Single-stage quantization results across COCO 2017, MiniPlaces, and Tiny ImageNet. Results are mean ± standard deviation across model seeds. Lower MSE, LPIPS, and rFID are better; higher perplexity (PPL) and code utilization are better. VQ, VQ-EMA, and AVQ use $K = 1024$; FSQ uses 1000 implicit codes.

METRIC	METHOD	COCO	MINIPLACES	TINY-IMAGENET
MSE ↓	VQ	0.0141 ± 0.0001	0.0113 ± 0.0003	0.0268 ± 0.0006
	VQ-EMA	$\mathbf{0.0112 \pm 0.0009}$	$\mathbf{0.0066 \pm 0.0001}$	$\mathbf{0.0233 \pm 0.0001}$
	FSQ	0.0123 ± 0.0002	0.0072 ± 0.0001	0.0246 ± 0.0001
	AVQ (ours)	0.0185 ± 0.0008	0.0114 ± 0.0006	0.0303 ± 0.0003
LPIPS ↓	VQ	0.0979 ± 0.0014	0.1825 ± 0.0038	0.1564 ± 0.0014
	VQ-EMA	$\mathbf{0.0798 \pm 0.0127}$	$\mathbf{0.0973 \pm 0.0041}$	0.1274 ± 0.0007
	FSQ	0.0935 ± 0.0027	0.1027 ± 0.0014	0.1472 ± 0.0020
	AVQ (ours)	0.0866 ± 0.0019	0.1070 ± 0.0012	$\mathbf{0.1180 \pm 0.0017}$
rFID ↓	VQ	51.28 ± 0.68	36.17 ± 1.13	31.82 ± 0.47
	VQ-EMA	50.58 ± 5.56	21.40 ± 1.60	31.69 ± 0.49
	FSQ	56.22 ± 1.07	23.11 ± 0.65	37.44 ± 1.38
	AVQ (ours)	$\mathbf{41.48 \pm 1.45}$	$\mathbf{18.23 \pm 0.80}$	$\mathbf{28.24 \pm 0.28}$
PPL ↑	VQ	135.3 ± 8.2	52.1 ± 4.6	143.2 ± 18.6
	VQ-EMA	652.8 ± 124.6	717.2 ± 22.7	616.1 ± 27.0
	FSQ	671.0 ± 13.7	681.5 ± 5.3	658.1 ± 7.6
	AVQ (ours)	$\mathbf{1000.5 \pm 3.5}$	$\mathbf{1009.8 \pm 3.1}$	$\mathbf{1003.3 \pm 5.3}$
Utilization (%) ↑	VQ	16.4 ± 1.2	6.7 ± 0.6	17.8 ± 2.2
	VQ-EMA	83.3 ± 15.9	$\mathbf{100.0 \pm 0.0}$	84.5 ± 4.1
	FSQ	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$
	AVQ (ours)	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$

Table 2: AVQ ablations at model seed 42. The default configuration uses $K = 1024$, $D = 64$, ℓ_2 normalization, and no learned code scale. All configurations achieve 100% utilization of the available codebook. Lower MSE, LPIPS, and rFID are better. PPL measures the effective breadth of code usage and should be interpreted relative to the available codebook size; the $K = 512$ configuration has a maximum possible perplexity of 512, whereas the remaining configurations have a maximum of 1024. Spatial reduction denotes $16\times16 \to 8\times8$ for COCO and Tiny ImageNet and $32\times32 \to 16\times16$ for MiniPlaces. Bold indicates the best value within each dataset for MSE, LPIPS, and rFID, based on the full-precision experimental values prior to display rounding.

METRIC	CONFIGURATION	COCO	MINIPLACES	TINY-IMAGENET
MSE ↓	Default	0.01841	**0.01077**	0.03056
	$D = 128$	**0.01839**	0.01123	0.03051
	No normalization	0.02007	0.01207	0.03269
	Learned scale	0.01852	0.01128	**0.03019**
	$K = 512$	0.02002	0.01396	0.03408
	Reduced spatial rate	0.04103	0.02466	0.05112
LPIPS ↓	Default	**0.08611**	0.10750	**0.11783**
	$D = 128$	0.08684	0.10832	0.12381
	No normalization	0.09059	0.11184	0.13248
	Learned scale	0.08878	**0.10638**	0.11847
	$K = 512$	0.09670	0.13896	0.13823
	Reduced spatial rate	0.24592	0.28352	0.31493
rFID ↓	Default	40.832	**18.381**	28.266
	$D = 128$	**40.334**	20.171	27.422
	No normalization	42.665	20.537	32.113
	Learned scale	41.531	18.730	**27.418**
	$K = 512$	41.172	25.136	31.349
	Reduced spatial rate	81.921	51.552	68.951
PPL ↑	Default	998.9	1009.9	997.2
	$D = 128$	997.5	1004.8	1006.0
	No normalization	997.8	1011.4	992.5
	Learned scale	1001.6	1009.3	1004.1
	$K = 512$	506.1	495.5	502.6
	Reduced spatial rate	992.0	997.6	972.4

Residual Quantization: Table 3 compares three-stage ARVQ with RVQ-EMA. ARVQ maintains near-maximal perplexity at every residual stage across all three datasets, with stage-wise perplexity above 1016 throughout, compared with approximately 496–757 for RVQ-EMA. This separation persists through the later residual stages, showing that ARVQ continues to use the available discrete capacity broadly as quantization proceeds through the residual hierarchy. ARVQ also exhibits sub-

Table 3: Three-stage residual quantization results. Results are mean ± standard deviation across model seeds. Lower MSE, LPIPS, and rFID are better; higher stage-wise perplexity (PPL) is better. ARVQ and RVQ-EMA use three $K = 1024$ codebooks.

METRIC	METHOD	COCO	MINIPLACES	TINY-IMAGENET
MSE ↓	RVQ-EMA	$\mathbf{0.00309 \pm 0.00003}$	$\mathbf{0.00201 \pm 0.00016}$	$\mathbf{0.00628 \pm 0.00003}$
	ARVQ (ours)	0.00477 ± 0.00004	0.00333 ± 0.00061	0.00841 ± 0.00009
LPIPS ↓	RVQ-EMA	$\mathbf{0.01006 \pm 0.00021}$	$\mathbf{0.01418 \pm 0.00180}$	$\mathbf{0.02248 \pm 0.00060}$
	ARVQ (ours)	0.01833 ± 0.00039	0.02787 ± 0.00501	0.02686 ± 0.00034
rFID ↓	RVQ-EMA	$\mathbf{10.69 \pm 0.05}$	$\mathbf{3.56 \pm 0.30}$	$\mathbf{5.47 \pm 0.28}$
	ARVQ (ours)	12.34 ± 0.22	3.98 ± 0.83	5.61 ± 0.16
PPL_1 ↑	RVQ-EMA	756.9 ± 10.5	495.8 ± 284.0	658.8 ± 10.0
	ARVQ (ours)	$\mathbf{1019.7 \pm 0.6}$	$\mathbf{1016.3 \pm 5.1}$	$\mathbf{1017.8 \pm 0.8}$
PPL_2 ↑	RVQ-EMA	700.7 ± 8.2	668.0 ± 3.6	617.4 ± 13.8
	ARVQ (ours)	$\mathbf{1021.0 \pm 0.1}$	$\mathbf{1019.0 \pm 2.6}$	$\mathbf{1020.1 \pm 0.3}$
PPL_3 ↑	RVQ-EMA	708.6 ± 5.8	662.3 ± 6.0	608.5 ± 11.4
	ARVQ (ours)	$\mathbf{1021.6 \pm 0.1}$	$\mathbf{1020.6 \pm 1.5}$	$\mathbf{1021.5 \pm 0.3}$

stantially more stable stage-wise perplexity across seeds. For example, on MiniPlaces, first-stage perplexity is 1016.3 ± 5.1 for ARVQ compared with 495.8 ± 284.0 for RVQ-EMA, whose large standard deviation indicates substantial variability across seeds. RVQ-EMA achieves lower MSE, LPIPS, and rFID across all three datasets, showing that ARVQ's primary empirical advantage is higher and more stable stage-wise perplexity, reflecting broader and more balanced code usage, rather than lower reconstruction distortion.

The effect of Phase-II hard-code decoder calibration is reported in Appendix Table 10 using the internal validation split employed for checkpoint selection. Calibration reduces hard-code reconstruction error for all methods while leaving the encoder and quantizer fixed; final evaluation-set results are reported separately in Tables 1 and 3.

Qualitative Results: Figure 1 compares hard-code reconstructions from the single-stage and residual quantizers on a fixed set of validation images. VQ produces visibly blurred reconstructions and loses fine structure in several examples, such as the clock face and bicycle. VQ-EMA, FSQ, and AVQ recover more of the overall object structure, although the single-stage reconstructions remain somewhat smoothed. Both residual methods recover substantially finer structural detail than the single-stage quantizers, with RVQ-EMA often producing the sharpest reconstructions in the examples shown. These qualitative observations are consistent with the lower reconstruction and perceptual errors achieved by the residual models, while ARVQ's primary quantitative advantage remains its substantially higher and more stable stage-wise perplexity, reflecting broader and more balanced code usage. Additional reconstruction grids for all datasets are provided in the Appendix.

6 Discussion and Limitations

The experiments reveal a consistent distinction between codebook usage and reconstruction distortion. In the single-stage setting, AVQ achieves complete utilization, near-maximal perplexity, and the lowest rFID across all three datasets, but does not achieve the lowest MSE and is not uniformly best in LPIPS. Moreover, FSQ activates all available codes while exhibiting substantially lower perplexity than AVQ, indicating that full utilization does not necessarily imply balanced effective usage. The ablations reinforce this distinction: reducing the latent spatial resolution substantially degrades MSE, LPIPS, and rFID even while perplexity remains close to the maximum available codebook size. Thus, broad and balanced code usage is an important property of a discrete representation, but is not by itself sufficient for low reconstruction distortion or high perceptual quality.

The residual experiments show the same trade-off from a different perspective. ARVQ maintains near-maximal perplexity at all three residual stages and substantially more stable stage-wise perplexity across seeds than RVQ-EMA, whereas RVQ-EMA achieves lower MSE, LPIPS, and rFID. The primary empirical benefit of ARVQ in the present experiments is therefore its ability to maintain broad and balanced stage-wise code usage, as reflected by near-maximal perplexity, rather than to minimize reconstruction error. For both AVQ and ARVQ, Phase-II hard-code decoder calibration bridges differentiable soft-to-hard training and discrete hard-code inference by adapting the decoder

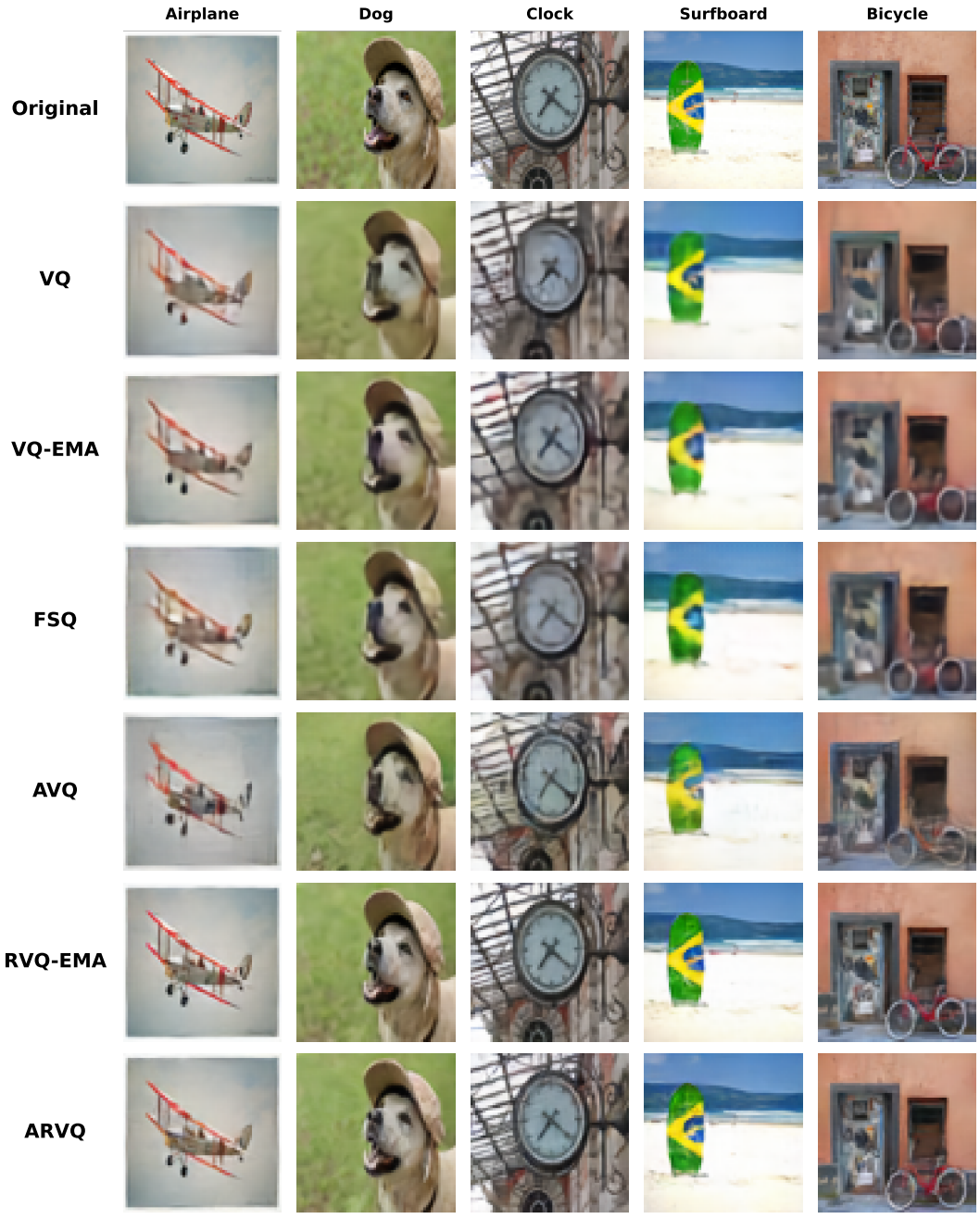

Figure 1: Qualitative reconstruction comparison on COCO. The same validation images are shown for all methods.

directly to the representations used at inference; the same calibration protocol is applied to the corresponding baselines for fair comparison, with detailed results reported in the Appendix. Together, these results highlight that code usage, decoder adaptation, and distortion are related but distinct aspects of quantizer behavior.

Several limitations suggest directions for future work. First, AVQ and ARVQ are evaluated using relatively compact convolutional autoencoders on three image datasets; their behavior with larger codebooks, higher-resolution models, audio, speech, video, multimodal tokenization, and downstream autoregressive or diffusion-based generative models remains to be studied. Second, we did not perform exhaustive benchmark-specific hyperparameter optimization for AVQ, ARVQ, or the baselines. The principal AVQ settings were established during preliminary MNIST and initial Tiny ImageNet experiments and then largely transferred across the reported benchmarks; further tuning of the temperature and soft-to-hard schedules, regularization weights, and corresponding baseline hyperparameters may improve reconstruction and perceptual performance. Third, AVQ computes attention over the full codebook, while ARVQ performs this operation at every residual stage; both also use a separate hard-code decoder-calibration phase, introducing additional computational and training overhead. Finally, because AVQ uses normalized code directions for attentive assignment, a natural extension is to more fully decouple angular code selection from representation magnitude. Building on the scalar code-scaling ablation, future work could explore richer direction–magnitude parameterizations inspired by DoRA (Liu et al., 2024), such as learnable codeword-specific magnitudes within each residual stage. Such a formulation may provide additional flexibility for representing residuals of different magnitudes while retaining broad and balanced code usage.

7 Conclusion

In this work, we introduced Attentive Vector Quantization (AVQ), an STE-free differentiable quantization framework that replaces hard nearest-neighbor assignment during training with cosine-similarity attention over a learned codebook. During Phase I, AVQ uses two complementary schedules: temperature annealing progressively sharpens the attention distribution, while a soft-to-hard weighting schedule gradually exposes the decoder to increasingly hard quantized representations while preserving a differentiable soft path for learning. Local-confidence and global-usage regularization further encourage increasingly discrete assignments while maintaining broad and balanced codebook utilization. In Phase II, the encoder and quantizer are frozen and the decoder is calibrated using hard codes, explicitly bridging differentiable training and discrete inference.

Across COCO, MiniPlaces, and Tiny ImageNet, AVQ achieves complete codebook utilization, near-maximal hard-code perplexity, and the lowest rFID among the evaluated single-stage quantizers (VQ, VQ-EMA, and FSQ). The ablations further show that complete utilization persists across changes in codebook size, embedding dimension, normalization, learned code scaling, and spatial rate. Notably, reducing the latent spatial resolution substantially degrades reconstruction and perceptual quality despite maintaining complete codebook utilization, showing that high utilization alone is not sufficient for high-quality reconstruction. We further extended the formulation to Attentive Residual Vector Quantization (ARVQ), which maintains near-maximal and stable stage-wise perplexity across all three residual stages, in contrast to the substantially lower and less consistent stage-wise perplexity observed with RVQ-EMA.

Taken together, these results show that broad and balanced use of discrete capacity can be achieved without a straight-through estimator in both single-stage and residual quantization settings. AVQ therefore provides a simple attentive formulation for discrete representation learning in which the transition from differentiable optimization to hard discrete inference is controlled explicitly through complementary temperature and soft-to-hard schedules followed by hard-code decoder calibration.

AI USE DISCLOSURE

The core conceptual framework of Attentive Vector Quantization (AVQ), including temperature-scaled differentiable latent–codebook attention as an alternative to straight-through estimation, was independently conceived by the author and initially developed through preliminary experiments on MNIST. Generative AI tools were subsequently used during research development, implementation, literature review, and manuscript preparation as described below.

Research Ideation and Experimental Design: Generative AI tools were used extensively as a research-development aid to discuss alternative modeling and training strategies, diagnose training behavior, and suggest experiments intended to improve stability and reconstruction quality. These discussions included normalization strategies, soft–hard scheduling, local and global codebook-usage objectives, learned code scaling, ablation design, and alternative ways of structuring the AVQ and ARVQ training procedures. Generative AI also suggested diagnostic analyses to identify potential sources of instability or poor reconstruction. In particular, after the author's initial temperature-annealed attentive quantization formulation had been developed, generative AI suggested introducing an explicit soft–hard interpolation weight and initially proposed a linear schedule for this transition. The author adopted the interpolation idea but instead proposed and implemented a cosine schedule consistent with the existing temperature-annealing strategy. The author also independently proposed the DoRA-inspired future direction of extending the scalar code-scaling formulation to richer direction–magnitude parameterizations, such as codeword-specific magnitudes within each residual stage. The author experimentally evaluated AVQ, ARVQ, and the baseline methods, and determined which suggested strategies and ablations were retained in the final study; the DoRA-inspired direction–magnitude formulation is proposed only as future work and was not evaluated in this study.

Coding and Implementation: The primary implementation was developed and executed by the author. Generative AI tools were used to review author-written functions and classes, identify bugs, suggest refactoring and efficiency improvements, generate portions of helper code, and assist with code for quantitative analysis, qualitative reconstructions, and several Appendix visualizations. All AI-assisted code was inspected, modified where necessary, and executed within the experimental workflow. All quantitative results reported in the paper were obtained from executed experiments and saved experimental outputs, and all qualitative reconstructions were generated from trained model checkpoints using the implemented inference and visualization pipeline rather than generated by AI.

Retrieval and Literature Review: Generative AI tools were used for initial literature discovery and for summarizing potentially relevant papers. The author independently expanded the literature search, including identifying relevant works that were not surfaced during AI-assisted retrieval, such as *AVQ-Attention*. AI-generated summaries were treated as preliminary aids; the author consulted the original papers and reviewed the relevant sections before incorporating related-work claims or citations into the manuscript.

Drafting and Writing Assistance: The author supplied technical content, experimental results, verbose drafts, and outline or bullet-point material, and generative AI tools were used to rewrite, condense, and reorganize this material, particularly to satisfy the conference page limit. Generative AI also assisted with manuscript scaffolding, LaTeX equations and tables, organization of sections, and language editing. The author reviewed and edited the resulting text for technical accuracy and consistency with the implementation and experimental results.

The author takes full responsibility for the methodology, implementation, reported results, citations, interpretations, and final content of this manuscript.

REPRODUCIBILITY STATEMENT

To support reproducibility, we provide detailed model architectures, training schedules, optimization settings, decoder-calibration procedures, and dataset configurations in the main paper and Appendix.

We additionally provide an anonymized reproducibility package with the OpenReview submission (`submission_35326_code.zip`). The package contains:

- `AVQ-ARVQ-Code-[35326].ipynb`: a self-contained implementation of AVQ and ARVQ for the Tiny ImageNet experiments at model seed 42, including the encoder, decoder, autoencoder, discriminator, quantizers, training utilities, evaluation utilities, and reconstruction-visualization code;
- `requirements.txt`: dependency specifications for running the notebook on external GPU environments;
- `README.md`: instructions for running the supplied experiments;
- `reference_results/`: saved outputs for the experiments reported in the paper, including per-experiment `result.json` files and `train.log` files across datasets, methods, model seeds, and completed ablations; and
- `LICENSE`: the Apache 2.0 license for the released code.

The notebook provides both a shortened quick-run configuration for verifying the complete training, calibration, evaluation, and visualization pipeline and the full training configuration used for the corresponding Tiny ImageNet experiment. AVQ and ARVQ can be selected using a single configuration flag. The quick-run configuration is intended for pipeline verification and is not intended to reproduce the reported full-training metrics. The full runs require approximately 7.5 hours for AVQ and 8.5 hours for ARVQ in the tested Kaggle GPU environment, while the quick configurations require approximately 3 and 3.5 hours, respectively.

The notebook is designed to run directly on Kaggle. It includes installation of the additional `torchmetrics[image]` evaluation dependency, so no separate installation from `requirements.txt` is required on Kaggle. Internet access is required when Tiny ImageNet is not already available because the notebook downloads the dataset automatically. For execution on other GPU platforms, the supplied `requirements.txt` can be used, and the dataset and experiment directory paths can be adjusted as described in the README.

The supplied reference results allow reviewers to inspect the numerical outputs and training behavior underlying the reported tables and figures and to compare reproduced Tiny ImageNet AVQ/ARVQ runs against the corresponding saved experiment outputs.

All supplementary code and experimental outputs are anonymized for double-blind review under submission number 35326. We intend to publicly release the complete repository after deanonymization.

REFERENCES

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

Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*, 2020.

Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

Tanmay Gautam, Reid Pryzant, Ziyi Yang, Chenguang Zhu, and Somayeh Sojoudi. Soft convex quantization: Revisiting vector quantization with convex optimization. In *International Conference on Learning Representations (ICLR)*, 2024.

Alexander Kolesnikov, André Susano Pinto, Lucas Beyer, Xiaohua Zhai, Jeremiah Harmsen, and Neil Houlsby. UViM: A unified modeling approach for vision with learned guiding codes. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 26295–26308, 2022.

Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. Stanford CS231n Course Project Report, 2015. URL `https://cs231n.stanford.edu/reports/2015/pdfs/yle_project.pdf`.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*, 2014.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-decomposed low-rank adaptation. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 32100–32121, 2024.

Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: VQ-VAE made simple. In *International Conference on Learning Representations (ICLR)*, 2024.

Pooneh Mousavi, Gallil Maimon, Adel Moumen, Darius Petermann, Jiatong Shi, Haibin Wu, Haici Yang, Anastasia Kuznetsova, Artem Ploujnikov, Ricard Marxer, Bhuvana Ramabhadran, Benjamin Elizalde, Loren Lugosch, Jinyu Li, Cem Subakan, Phil Woodland, Minje Kim, Hung yi Lee, Shinji Watanabe, Yossi Adi, and Mirco Ravanelli. Discrete audio tokens: More than a survey!, 2025.

Kashif Rasul, Andrew Bennett, Pablo Vicente, Umang Gupta, Hena Ghonia, Anderson Schneider, and Yuriy Nevmyvaka. VQ-TR: Vector quantized attention for time series forecasting. In *International Conference on Learning Representations*, 2024.

Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with VQ-VAE-2. In *Advances in Neural Information Processing Systems*, 2019.

Guobin Shen, Bing Zeng, and Ming L. Liou. Adaptive vector quantization with codebook updating based on locality and history. *IEEE Transactions on Image Processing*, 12(3):283–295, 2003. doi: 10.1109/TIP.2003.810915.

Mohammad Hassan Vali, Tom Bäckström, and Arno Solin. Diveq: Differentiable vector quantization using the reparameterization trick, 2026.

Winfried van den Dool, Patrick Forré, Amir Habibian, Yuki M. Asano, and Max Welling. AVQ-Attention: Adaptive vector-quantized attention. *arXiv preprint arXiv:2607.12789*, 2026. doi: 10.48550/arXiv.2607.12789.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, 2017.

Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. VideoGPT: Video generation using VQ-VAE and transformers. *arXiv preprint arXiv:2104.10157*, 2021.

Zhou Yang, Weisheng Dong, Xin Li, Mengluan Huang, Yulin Sun, and Guangming Shi. Vector quantization with self-attention for quality-independent representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24438–24448, 2023.

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 VQ-GAN. In *International Conference on Learning Representations (ICLR)*, 2022.

Lijun Yu, José Lezama, Nitesh B Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Vighnesh Birodkar, 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 *International Conference on Learning Representations (ICLR)*, 2024.

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

Jiahui Zhang, Fangneng Zhan, Christian Theobalt, and Shijian Lu. Regularized vector quantization for tokenized image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18467–18476, 2023.

Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):212–224, 2017.

This appendix provides additional formulation and implementation details for AVQ and ARVQ, including the exact annealing schedules, training and decoder-calibration procedures, model architectures, and experimental hyperparameters. We additionally report extended calibration and codebook-usage analyses, auxiliary ARVQ ablations, qualitative reconstructions, and visualizations of the learned AVQ codebook geometry.

A Additional Method and Implementation Details

A.1 Token-Level AVQ Formulation

For completeness, we give the token-level form of the matrix operation in Equation 1. Each encoder token and codebook vector is ℓ_2-normalized as

$$\tilde{\mathbf{z}}_n = \frac{\mathbf{z}_n}{\max(\|\mathbf{z}_n\|_2, \epsilon_{\text{norm}})}, \qquad \tilde{\mathbf{e}}_k = \frac{\mathbf{e}_k}{\max(\|\mathbf{e}_k\|_2, \epsilon_{\text{norm}})}, \tag{7}$$

where $\epsilon_{\text{norm}} = 10^{-6}$ is used for numerical stability. The cosine similarity and temperature-controlled assignment probability are

$$s_{nk} = \tilde{\mathbf{z}}_n^\top \tilde{\mathbf{e}}_k, \qquad p_{nk} = \frac{\exp(s_{nk}/\tau_t)}{\sum_{j=1}^{K} \exp(s_{nj}/\tau_t)}. \tag{8}$$

The corresponding differentiable soft representation is

$$\mathbf{q}_n^{\text{soft}} = \sum_{k=1}^{K} p_{nk} \tilde{\mathbf{e}}_k. \tag{9}$$

For hard quantization, $k_n^* = \arg\max_k p_{nk}$ and $\mathbf{q}_n^{\text{hard}} = \tilde{\mathbf{e}}_{k_n^*}$. Thus, in the default normalized formulation, normalized codebook vectors determine both the cosine-similarity assignments and the soft and hard quantized representations.

For the learned-code-scale ablation, a positive scalar $c = \text{softplus}(\rho) + 10^{-6}$ is applied to both representations, giving $\mathbf{q}_n^{\text{soft}} = c \sum_k p_{nk} \tilde{\mathbf{e}}_k$ and $\mathbf{q}_n^{\text{hard}} = c\tilde{\mathbf{e}}_{k_n^*}$. The default AVQ configuration does not use learned code scaling.

A.2 ARVQ Stage-Wise Formulation

Each ARVQ stage applies AVQ independently to the current residual using its own codebook. At stage s, the current residual is ℓ_2-normalized to $\tilde{\mathbf{R}}^{(s)}$, and the stage codebook is normalized to $\tilde{\mathbf{E}}^{(s)}$. The attentive assignment and corresponding soft representation are

$$\mathbf{P}^{(s)} = \text{softmax}\left(\frac{\tilde{\mathbf{R}}^{(s)} \tilde{\mathbf{E}}^{(s)\top}}{\tau_t}\right), \qquad \mathbf{Q}_{\text{soft}}^{(s)} = \mathbf{P}^{(s)} \tilde{\mathbf{E}}^{(s)}. \tag{10}$$

The hard representation $\mathbf{Q}_{\text{hard}}^{(s)}$ is obtained by selecting the maximum-probability normalized codebook vector independently for each residual token. During training, the stage representation is

$$\mathbf{Q}^{(s)} = \alpha_t \mathbf{Q}_{\text{soft}}^{(s)} + (1 - \alpha_t)\, \text{sg}\left(\mathbf{Q}_{\text{hard}}^{(s)}\right). \tag{11}$$

The residual passed to the next stage is

$$\mathbf{R}^{(s+1)} = \mathbf{R}^{(s)} - \text{sg}\left(\mathbf{Q}^{(s)}\right), \tag{12}$$

so later stages receive the remaining residual while gradients do not propagate through earlier-stage outputs. The aggregate representation is

$$\mathbf{Q} = \sum_{s=1}^{S} \mathbf{Q}^{(s)}. \tag{13}$$

Each stage has its own codebook objective $\mathcal{L}_{\mathrm{CB}}^{(s)}$, and the ARVQ codebook objective is averaged across stages:

$$\mathcal{L}_{\mathrm{ARVQ}} = \frac{1}{S}\sum_{s=1}^{S}\mathcal{L}_{\mathrm{CB}}^{(s)}. \tag{14}$$

At inference, ARVQ operates sequentially with hard assignments, and the decoder receives the sum of the hard stage outputs:

$$\mathbf{Q}^{\mathrm{hard}} = \sum_{s=1}^{S}\mathbf{Q}_{\mathrm{hard}}^{(s)}. \tag{15}$$

A.3 Soft-to-Hard and Temperature Schedules

The soft contribution α_t is held at 1 during an initial warm-up period, cosine-annealed to a positive endpoint $\alpha_{\min}$, and then held fixed. Let T_{warm} and T_{end} denote the beginning and end of the transition period. The schedule is

$$\alpha_t = \begin{cases} 1, & t < T_{\mathrm{warm}}, \\ \alpha_{\min} + \dfrac{1-\alpha_{\min}}{2}\left[1+\cos(\pi r_t)\right], & T_{\mathrm{warm}} \le t < T_{\mathrm{end}}, \\ \alpha_{\min}, & t \ge T_{\mathrm{end}}, \end{cases} \tag{16}$$

where

$$r_t = \frac{t - T_{\mathrm{warm}}}{T_{\mathrm{end}} - T_{\mathrm{warm}}} \in [0,1]. \tag{17}$$

The assignment temperature is annealed from $\tau_{\max}$ to $\tau_{\min}$ using cosine interpolation in log-temperature space. Defining

$$c(u) = \frac{1-\cos(\pi u)}{2}, \qquad u \in [0,1], \tag{18}$$

the temperature schedule is

$$\log \tau_t = (1 - c(u_t))\log \tau_{\max} + c(u_t)\log \tau_{\min}, \tag{19}$$

where u_t denotes normalized progress through the temperature-annealing interval. The temperature is held at $\tau_{\min}$ after annealing is complete.

A.4 Phase-I Training Objective and Procedure

When adversarial training is enabled, we use a hinge objective with discriminator D_ψ. The discriminator loss is

$$\mathcal{L}_D = \frac{1}{2}\left(\mathbb{E}_{\mathbf{x}}\left[\max(0, 1 - D_\psi(\mathbf{x}))\right] + \mathbb{E}_{\hat{\mathbf{x}}}\left[\max(0, 1 + D_\psi(\hat{\mathbf{x}}))\right]\right), \tag{20}$$

and the autoencoder-side adversarial loss is

$$\mathcal{L}_{\mathrm{adv}} = -\mathbb{E}_{\hat{\mathbf{x}}}\left[D_\psi(\hat{\mathbf{x}})\right]. \tag{21}$$

The discriminator provides an auxiliary training signal for reconstructed images and is not part of the inference model. The adversarial term is disabled during the initial reconstruction-only phase and activated later according to the training schedule.

Algorithm 1 summarizes a Phase-I AVQ training step. At each training step, τ_t and α_t are obtained from the schedules above. Gradients through the quantizer propagate exclusively through the differentiable soft pathway, while detached hard codes are progressively introduced to the decoder.

Algorithm 1 AVQ Phase-I Training Step

Require: Image batch $\mathbf{x}$, encoder Enc_θ, decoder Dec_ϕ, codebook $\mathbf{E}$, discriminator D_ψ

$\mathbf{Z} \leftarrow \mathrm{Enc}_\theta(\mathbf{x})$

$\tilde{\mathbf{Z}} \leftarrow \ell_2\text{-normalize}(\mathbf{Z}), \quad \tilde{\mathbf{E}} \leftarrow \ell_2\text{-normalize}(\mathbf{E})$

$\mathbf{P} \leftarrow \mathrm{softmax}\left(\tilde{\mathbf{Z}}\tilde{\mathbf{E}}^\top/\tau_t\right)$

$\mathbf{Q}^{\mathrm{soft}} \leftarrow \mathbf{P}\tilde{\mathbf{E}}$

$\mathbf{k}^* \leftarrow \arg\max_k \mathbf{P}$

$\mathbf{Q}^{\mathrm{hard}} \leftarrow \tilde{\mathbf{E}}[\mathbf{k}^*]$

$\mathbf{Q}^{\mathrm{train}} \leftarrow \alpha_t \mathbf{Q}^{\mathrm{soft}} + (1-\alpha_t)\,\mathrm{sg}(\mathbf{Q}^{\mathrm{hard}})$

$\hat{\mathbf{x}} \leftarrow \mathrm{Dec}_\phi(\mathbf{Q}^{\mathrm{train}})$

Compute $\mathcal{L}_{\mathrm{rec}}$ and $\mathcal{L}_{\mathrm{CB}}$

if adversarial training is active **then**

  Update D_ψ using $\mathcal{L}_D$ on $\mathbf{x}$ and $\mathrm{sg}(\hat{\mathbf{x}})$

  Compute $\mathcal{L}_{\mathrm{adv}}$

else

  $\mathcal{L}_{\mathrm{adv}} \leftarrow 0$

end if

$\mathcal{L}_{\mathrm{AE}} \leftarrow \mathcal{L}_{\mathrm{rec}} + \lambda_{\mathrm{CB}}\mathcal{L}_{\mathrm{CB}} + \lambda_{\mathrm{adv}}\mathcal{L}_{\mathrm{adv}}$

Update Enc_θ, Dec_ϕ, and $\mathbf{E}$ using $\mathcal{L}_{\mathrm{AE}}$

A.5 Phase-II Hard-Code Decoder Calibration

After Phase I, the encoder Enc_θ and codebook are frozen and the decoder Dec_ϕ is calibrated using hard quantized representations only. Calibration therefore updates only the decoder parameters and introduces no additional gradient path through the quantizer. This phase adapts the decoder directly to the discrete representations used at inference and reduces the remaining soft–hard reconstruction gap. The corresponding optimization settings and stopping criteria are summarized in Table 5.

A.6 Architectures and Training Hyperparameters

Table 4 summarizes the default AVQ and ARVQ training hyperparameters used across Tiny ImageNet, COCO 2017, and MiniPlaces unless otherwise specified. Phase I is run for 50 epochs. The assignment temperature is cosine-annealed in log-temperature space from $\tau_{\max} = 0.15$ to $\tau_{\min} = 0.02$ over the first 40 epochs. The soft contribution is initialized at $\alpha_{\mathrm{start}} = 1.0$, held fixed for the first 15 epochs, and cosine-annealed to $\alpha_{\min} = 0.2$ from epochs 15 to 40. Although these schedule boundaries are specified in epochs, both schedules are evaluated at every global training step.

The codebook objective is weighted by $\lambda_{\mathrm{CB}} = 0.01$, with $\lambda_{\mathrm{local}} = 0.1$ and $\lambda_{\mathrm{balance}} = 1.5$, and uses sharpening exponent $\gamma = 4$. Adversarial training begins at epoch 5 with $\lambda_{\mathrm{adv}} = 0.01$, following an initial reconstruction-only warm-up. Batch size is 64 for Tiny ImageNet and COCO at 64×64 resolution and 16 for MiniPlaces at 128×128, keeping the number of spatial pixels per batch constant. Phase-II decoder-calibration settings are summarized in Table 5.

Baseline Quantizer Configurations: The baseline quantizers use fixed method-specific configurations across the three reported datasets, summarized in Table 6. VQ and VQ-EMA use $K = 1024$ codes with embedding dimension $D = 64$ and commitment weight 0.25. VQ-EMA additionally uses EMA decay 0.99 and $\epsilon = 10^{-5}$. FSQ uses $D = 64$ with scalar levels $(8, 5, 5, 5)$, corresponding to 1000 possible discrete codes. RVQ-EMA uses three residual stages, each with $K = 1024$ and $D = 64$, commitment weight 0.25, EMA decay 0.99, and $\epsilon = 10^{-5}$. These method-specific settings are kept fixed across Tiny ImageNet, COCO 2017, and MiniPlaces.

Encoder and Decoder Architecture: All evaluated quantization methods use a matched convolutional encoder and decoder backbone within each dataset; only the quantization module is changed. The same architectural template is used across datasets, while the spatial dimensions depend on the input resolution. Tiny ImageNet and COCO use 64×64 inputs and produce a 16×16 latent grid,

Table 4: Default AVQ and ARVQ training hyperparameters used across Tiny ImageNet, COCO 2017, and MiniPlaces.

Hyperparameter	**Symbol**	**Value**
Phase I training epochs	–	50
Codebook-objective weight	λ_{CB}	0.01
Adversarial-loss weight	λ_{adv}	0.01
Adversarial start epoch	e_{adv}	5
Maximum temperature	$\tau_{\max}$	0.15
Minimum temperature	$\tau_{\min}$	0.02
Temperature annealing period (epochs)	e_{anneal}	40
Local entropy weight	λ_{local}	0.1
Global balance weight	$\lambda_{\mathrm{balance}}$	1.5
Sharpening exponent	γ	4.0
Initial soft weight	α_{start}	1.0
Minimum soft weight	$\alpha_{\min}$	0.2
Soft-weight warm-up (epochs)	e_{warm}	15
Soft-to-hard transition end (epoch)	e_{end}	40

Table 5: Optimization and decoder-calibration settings.

Setting	**Value**
Autoencoder optimizer	Adam
Autoencoder learning rate	1×10^{-4}
Discriminator learning rate	4×10^{-4}
Adam β	$(0.5, 0.9)$
Phase-II decoder learning rate	1×10^{-4}
Maximum calibration epochs	10
Calibration patience	3
Tiny ImageNet / COCO batch size	64
MiniPlaces batch size	16

Table 6: Quantizer-specific configurations used for the baseline methods.

Method	**Configuration**
VQ	$K = 1024,\ D = 64,\ \beta_{\mathrm{commit}} = 0.25$
VQ-EMA	$K = 1024,\ D = 64,\ \beta_{\mathrm{commit}} = 0.25,$ decay $= 0.99,\ \epsilon = 10^{-5}$
FSQ	$D = 64$, levels $= (8, 5, 5, 5)$; 1000 codes
RVQ-EMA	3 stages, $K = 1024,\ D = 64,\ \beta_{\mathrm{commit}} = 0.25,$ decay $= 0.99,\ \epsilon = 10^{-5}$

whereas MiniPlaces uses 128×128 inputs and produces a 32×32 latent grid. Tables 7 and 8 summarize this shared architectural template.

Table 7: Encoder architecture template used in the default experiments. For COCO and Tiny ImageNet, $H = 64$, yielding a 16×16 latent grid; for MiniPlaces, $H = 128$, yielding a 32×32 latent grid. The default embedding dimension is $D = 64$. Each residual block preserves spatial resolution and channel dimensionality and consists of GroupNorm(8), SiLU, 3×3 convolution, GroupNorm(8), SiLU, and a second 3×3 convolution with an identity skip connection.

LAYER	OPERATION	OUTPUT SHAPE
Input	RGB image	$3 \times H \times H$
1	Conv 3×3, $3 \to 64$, stride 1	$64 \times H \times H$
2	ResidualBlock(64)	$64 \times H \times H$
3	Conv 4×4, $64 \to 128$, stride 2	$128 \times H/2 \times H/2$
4	ResidualBlock(128)	$128 \times H/2 \times H/2$
5	Conv 4×4, $128 \to 256$, stride 2	$256 \times H/4 \times H/4$
6	ResidualBlock(256)	$256 \times H/4 \times H/4$
7	GroupNorm(8) + SiLU	$256 \times H/4 \times H/4$
8	Conv 1×1, $256 \to D$	$D \times H/4 \times H/4$

Table 8: Decoder architecture template used in the default experiments. For COCO and Tiny ImageNet, the decoder receives a $D \times 16 \times 16$ quantized latent representation and reconstructs a 64×64 image; for MiniPlaces, it receives a $D \times 32 \times 32$ latent representation and reconstructs a 128×128 image. The reconstructed RGB image is normalized to $[-1, 1]$.

LAYER	OPERATION	OUTPUT SHAPE
Input	Quantized latent	$D \times H/4 \times H/4$
1	Conv 3×3, $D \to 256$, stride 1	$256 \times H/4 \times H/4$
2	ResidualBlock(256)	$256 \times H/4 \times H/4$
3	Transposed Conv 4×4, $256 \to 128$, stride 2	$128 \times H/2 \times H/2$
4	ResidualBlock(128)	$128 \times H/2 \times H/2$
5	Transposed Conv 4×4, $128 \to 64$, stride 2	$64 \times H \times H$
6	ResidualBlock(64)	$64 \times H \times H$
7	GroupNorm(8) + SiLU	$64 \times H \times H$
8	Conv 3×3, $64 \to 3$	$3 \times H \times H$
9	Tanh	$3 \times H \times H$

A.7 Hyperparameter Development and Transfer

AVQ hyperparameters were initially developed using preliminary experiments on MNIST. During this stage, we explored a small number of settings for the temperature schedule, soft-to-hard weighting schedule, and codebook-usage regularization. For example, terminal temperatures of 0.01 and 0.02 were examined; a terminal temperature of 0.01 caused the soft assignments to approach hard, nearly one-hot assignments too rapidly during preliminary development, whereas 0.02 retained a smoother soft-to-hard transition and was therefore selected for subsequent experiments.

Table 9 compares the resulting preliminary MNIST configuration with the configuration used in the reported experiments. Most settings were transferred directly. The principal subsequent change was an increase in the minimum soft weight from 0.1 to 0.2 during the initial Tiny ImageNet experiments.

The resulting configuration was then used across COCO, MiniPlaces, and Tiny ImageNet rather than independently optimized for each benchmark. Dataset-specific choices such as image resolution, latent spatial resolution, and batch size were adjusted as described earlier. We did not perform exhaustive benchmark-specific hyperparameter searches for any of the evaluated quantizers. AVQ and ARVQ were evaluated with the fixed configuration described above, while VQ, VQ-EMA, FSQ, and RVQ-EMA were likewise evaluated using the fixed method-specific configurations in Table 6, rather than independently tuned for each dataset. The experiments therefore emphasize controlled within-dataset comparisons using matched encoder/decoder architectures and fixed method-specific quantizer configurations rather than the best attainable performance of each method after extensive hyperparameter optimization.

Table 9: Comparison of the principal AVQ hyperparameters used during preliminary MNIST development and in the reported experiments.

Hyperparameter	Symbol	MNIST development	Reported experiments
Phase I training epochs	–	50	50
Codebook-objective weight	λ_{CB}	0.01	0.01
Adversarial-loss weight	λ_{adv}	0.01	0.01
Adversarial start epoch	e_{adv}	5	5
Maximum temperature	$\tau_{\max}$	0.15	0.15
Minimum temperature	$\tau_{\min}$	0.02	0.02
Temperature annealing period	e_{anneal}	40	40
Local entropy weight	λ_{local}	0.1	0.1
Global balance weight	$\lambda_{\mathrm{balance}}$	1.5	1.5
Sharpening exponent	γ	4.0	4.0
Initial soft weight	α_{start}	1.0	1.0
Minimum soft weight	$\alpha_{\min}$	0.1	0.2
Soft-weight warm-up	e_{warm}	15	15
Soft-to-hard transition end	e_{end}	40	40

B Additional Experimental Analyses

B.1 Hard-Code Decoder Calibration

Table 10 reports hard-code reconstruction error at the Phase-I checkpoint and after Phase-II decoder calibration. Figures 2 and 3 visualize the same comparison for single-stage and residual quantization, respectively.

Table 10: Effect of hard-code decoder calibration across model seeds. Results are hard-code MSE on the internal validation split at the Phase-I checkpoint and after Phase-II decoder calibration, reported as mean ± standard deviation across three model seeds. During calibration, the encoder and quantizer are frozen and only the decoder is optimized. The same calibration protocol is applied to all methods. These values are used to measure the effect of calibration and are distinct from the final evaluation-set MSE reported in the main results. Lower MSE is better.

METHOD	CALIBRATION	COCO	MINIPLACES	TINY-IMAGENET
Single-stage quantization				
VQ	Phase-I Checkpoint	0.01768 ± 0.00015	0.01222 ± 0.00036	0.03138 ± 0.00018
	Phase-II Calibrated	0.01432 ± 0.00010	0.01124 ± 0.00033	0.02683 ± 0.00060
VQ-EMA	Phase-I Checkpoint	0.01282 ± 0.00024	0.00686 ± 0.00002	0.02457 ± 0.00023
	Phase-II Calibrated	0.01138 ± 0.00087	0.00655 ± 0.00012	0.02343 ± 0.00007
FSQ	Phase-I Checkpoint	0.01387 ± 0.00047	0.00765 ± 0.00014	0.02602 ± 0.00026
	Phase-II Calibrated	0.01256 ± 0.00024	0.00717 ± 0.00006	0.02469 ± 0.00007
AVQ (ours)	Phase-I Checkpoint	0.04095 ± 0.00250	0.01977 ± 0.00764	0.05143 ± 0.00262
	Phase-II Calibrated	0.01874 ± 0.00080	0.01134 ± 0.00061	0.03036 ± 0.00027
Residual quantization				
RVQ-EMA	Phase-I Checkpoint	0.00368 ± 0.00007	0.00226 ± 0.00012	0.00689 ± 0.00007
	Phase-II Calibrated	0.00316 ± 0.00003	0.00200 ± 0.00016	0.00631 ± 0.00003
ARVQ (ours)	Phase-I Checkpoint	0.00583 ± 0.00013	0.00401 ± 0.00085	0.00949 ± 0.00009
	Phase-II Calibrated	0.00486 ± 0.00004	0.00332 ± 0.00061	0.00843 ± 0.00009

B.2 Curriculum and Codebook-Usage Dynamics

Figures 4–7 show the Phase-I soft-to-hard curriculum, active-code utilization, normalized hard-code perplexity, and stage-wise residual perplexity across datasets.

B.3 Auxiliary ARVQ Ablations

Table 11 reports the completed ARVQ learned-scale and reduced-spatial-rate ablations at model seed 42.

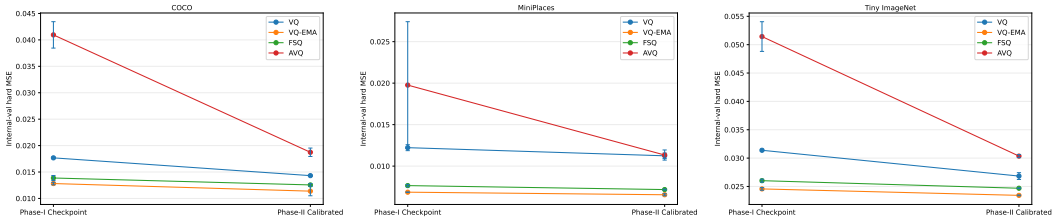

Figure 2: Effect of Phase-II hard-code decoder calibration for single-stage quantization on COCO, MiniPlaces, and Tiny ImageNet. Points show mean hard-code MSE on the internal validation split across model seeds, and error bars indicate standard deviation. "Phase-I checkpoint" denotes the selected Phase-I model evaluated with hard codes, while "Phase-II calibrated" denotes the decoder after hard-code calibration with the encoder and quantizer frozen. The same decoder-calibration protocol is applied to VQ, VQ-EMA, FSQ, and AVQ.

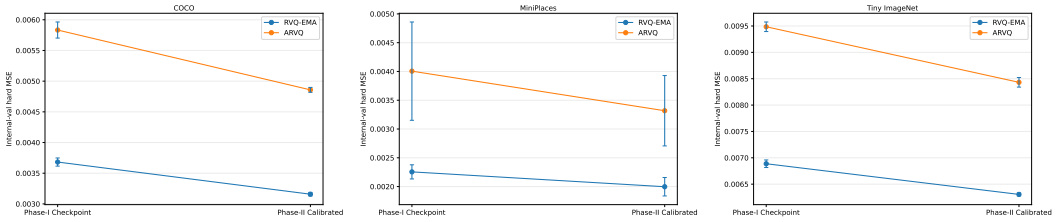

Figure 3: Effect of Phase-II hard-code decoder calibration for residual quantization on COCO, MiniPlaces, and Tiny ImageNet. Points show mean hard-code MSE on the internal validation split across model seeds, and error bars indicate standard deviation. "Phase-I checkpoint" denotes the selected Phase-I model evaluated with hard codes, while "Phase-II calibrated" denotes the decoder after hard-code calibration with the encoder and quantizer frozen. The same decoder-calibration protocol is applied to RVQ-EMA and ARVQ.

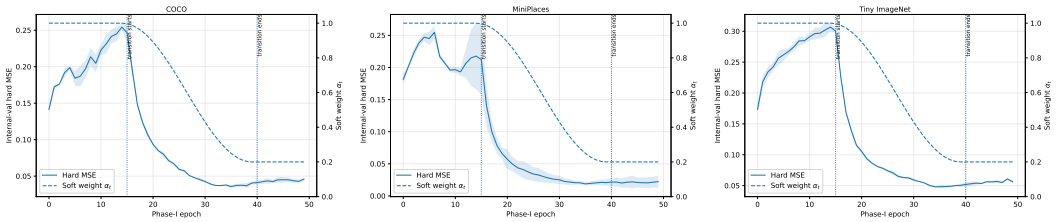

Figure 4: AVQ soft-to-hard curriculum dynamics across COCO, MiniPlaces, and Tiny ImageNet. Solid curves show mean validation hard-code MSE across model seeds, with shaded regions indicating standard deviation; dashed curves show the soft weight α_t. Vertical dotted lines mark the beginning and end of the soft-to-hard transition.

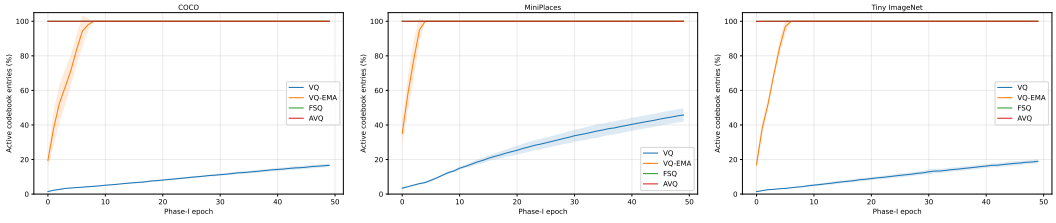

Figure 5: Codebook-utilization dynamics during Phase-I training on COCO, MiniPlaces, and Tiny ImageNet. Curves show the mean percentage of active codebook entries across model seeds, with shaded regions indicating standard deviation. AVQ maintains complete codebook utilization throughout training, while conventional VQ exhibits substantial codebook collapse. VQ-EMA rapidly expands utilization during early training, and FSQ uses all available implicit codes.

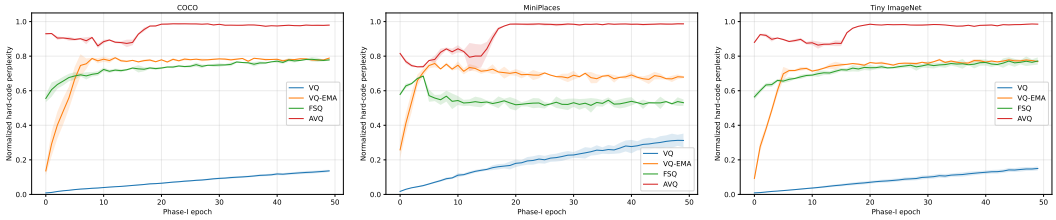

Figure 6: Codebook-usage dynamics during Phase-I training on COCO, MiniPlaces, and Tiny ImageNet. Curves show mean normalized hard-code perplexity across model seeds, with shaded regions indicating standard deviation. Perplexity is normalized by the available codebook size K, allowing comparison between VQ, VQ-EMA, AVQ ($K = 1024$) and FSQ (1000 implicit codes). AVQ approaches near-maximal normalized perplexity across all three datasets, while conventional VQ remains substantially lower.

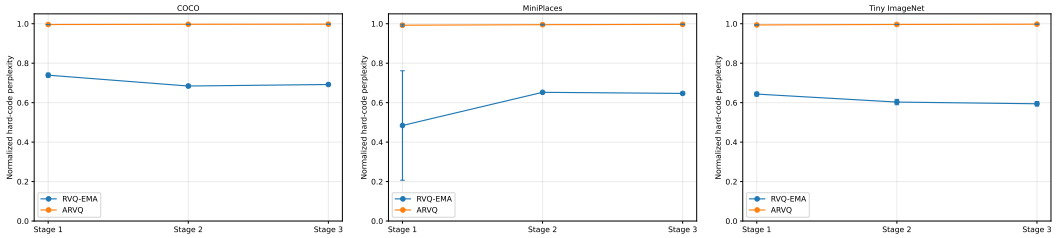

Figure 7: Stage-wise codebook usage for three-stage residual quantization on COCO, MiniPlaces, and Tiny ImageNet. Points show mean normalized hard-code perplexity across model seeds, with error bars indicating standard deviation. Perplexity is normalized by the stage codebook size $K = 1024$. ARVQ maintains near-maximal normalized perplexity across all residual stages and datasets, whereas RVQ-EMA exhibits substantially lower stage-wise perplexity and greater variability in some settings.

Table 11: Auxiliary ARVQ results at model seed 42. All experiments use three $K = 1024$ residual codebooks. "Learned scale" introduces a learned scalar code scale. The reduced spatial-rate experiment is evaluated on MiniPlaces using a 16×16 latent grid instead of the default 32×32. Lower MSE, LPIPS, and rFID are better; higher stage-wise perplexity (PPL) is better. Phase-I and Phase-II MSE denote internal-validation hard-code reconstruction error at the selected Phase-I checkpoint and after decoder calibration, respectively.

METRIC	CONFIGURATION	COCO	MINIPLACES	TINY-IMAGENET
Final evaluation MSE ↓	Learned scale	0.00482	0.00356	0.00841
	Reduced spatial rate	–	0.01310	–
LPIPS ↓	Learned scale	0.01799	0.03170	0.02672
	Reduced spatial rate	–	0.12988	–
rFID ↓	Learned scale	12.57	4.46	5.51
	Reduced spatial rate	–	20.85	–
PPL_1 ↑	Learned scale	1019.5	1015.7	1018.5
	Reduced spatial rate	–	1003.4	–
PPL_2 ↑	Learned scale	1021.3	1019.2	1020.8
	Reduced spatial rate	–	1010.0	–
PPL_3 ↑	Learned scale	1021.9	1020.5	1021.8
	Reduced spatial rate	–	1014.8	–
Phase-I MSE ↓	Learned scale	0.00584	0.00457	0.00953
	Reduced spatial rate	–	0.01766	–
Phase-II MSE ↓	Learned scale	0.00491	0.00355	0.00843
	Reduced spatial rate	–	0.01308	–

B.4 Additional Qualitative Reconstructions

Figures 8–10 provide additional hard-code reconstruction examples for COCO, MiniPlaces, and Tiny ImageNet using the same source images across methods.

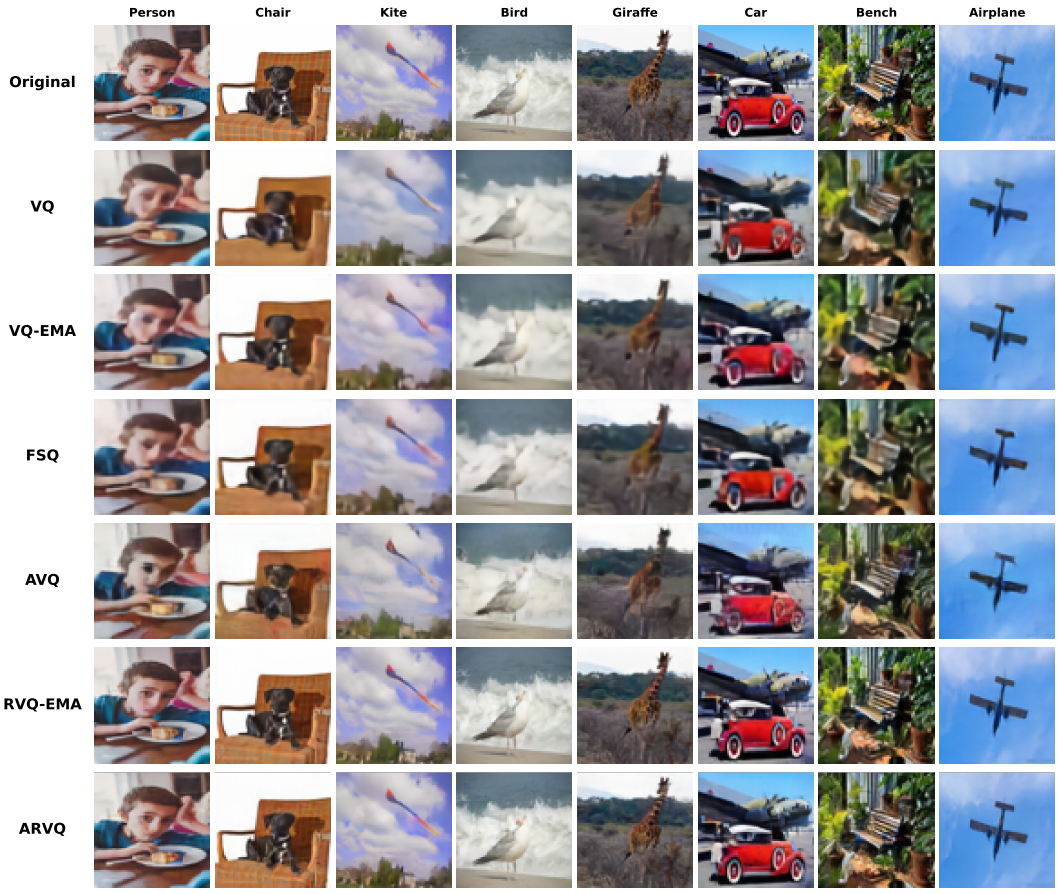

Figure 8: Additional qualitative reconstruction comparison on COCO. The same validation images are shown for all methods.

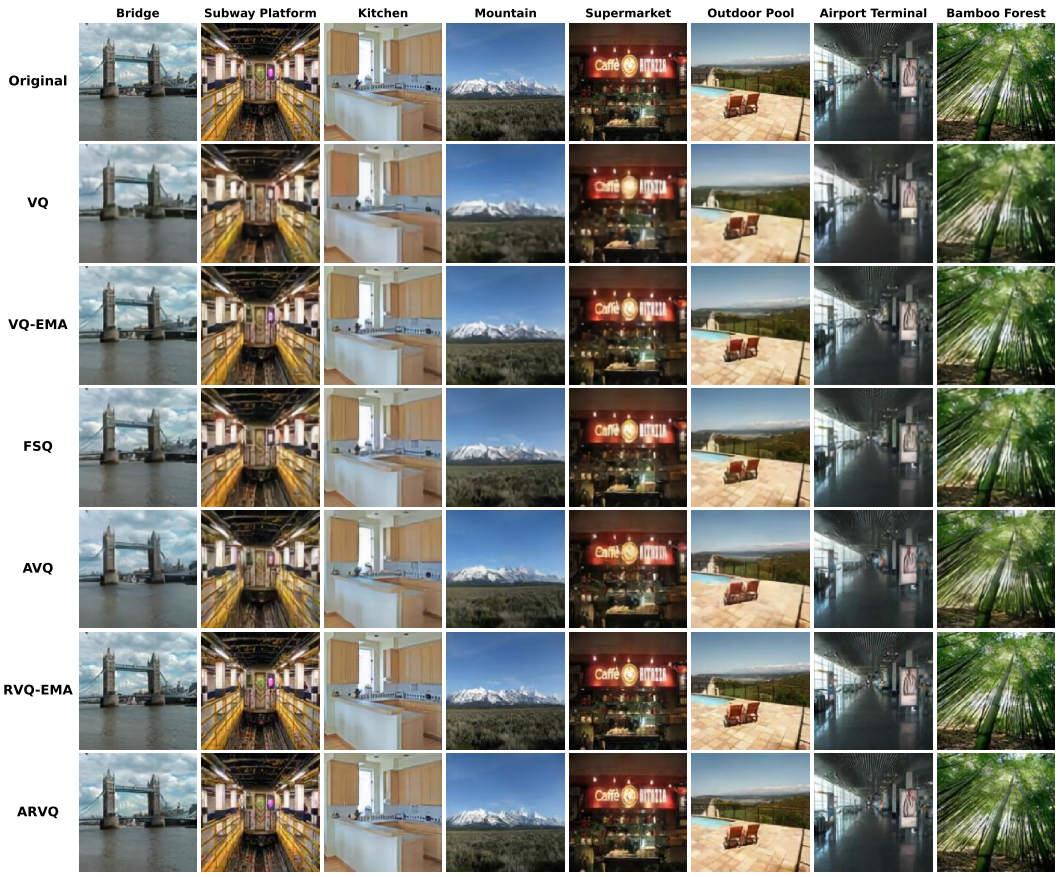

Figure 9: Qualitative reconstruction comparison on MiniPlaces. The same validation images are shown for all methods.

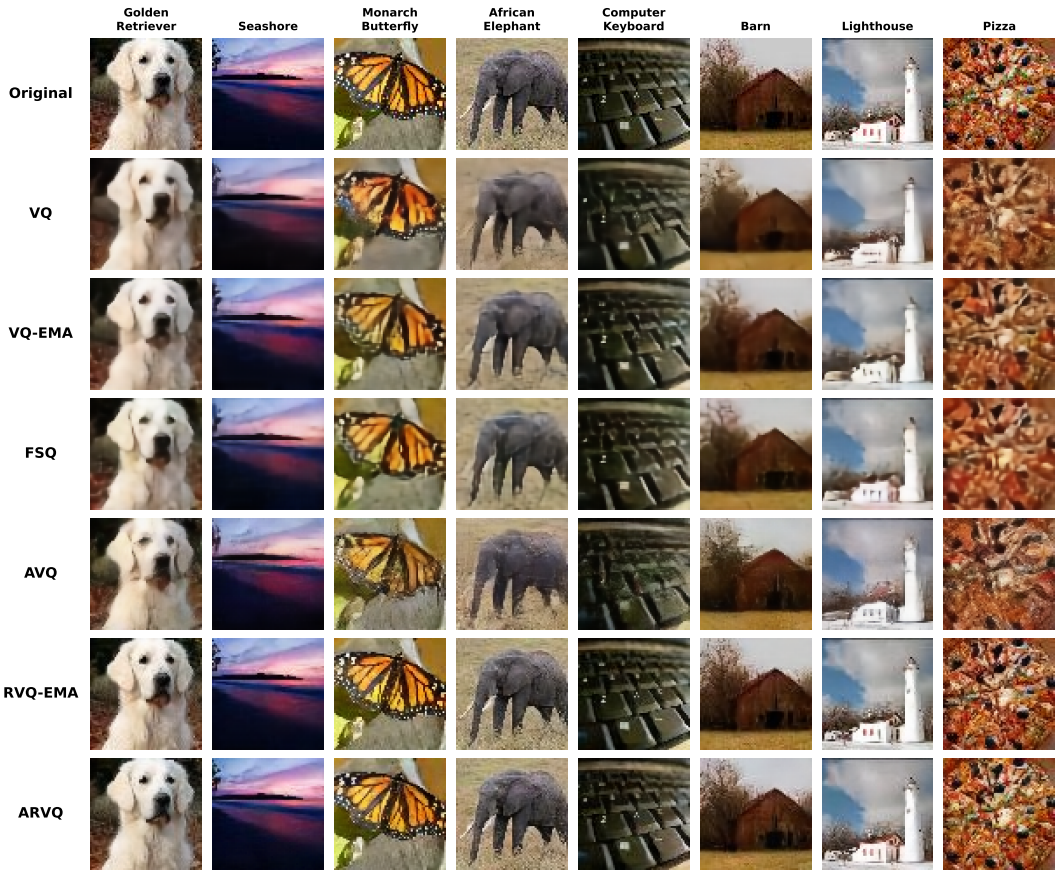

Figure 10: Qualitative reconstruction comparison on Tiny ImageNet. The same validation images are shown for all methods.

C CODEBOOK GEOMETRY VISUALIZATION

To qualitatively examine the geometry of the learned AVQ codebook, we construct a spatial visualization of the normalized code vectors and inspect the response of the trained decoder to the resulting organization. Because the codebook $\tilde{E} \in \mathbb{R}^{K \times D}$ has no inherent spatial ordering, we first embed its codewords into two dimensions and then assign them bijectively to a regular spatial lattice.

For the default codebook size $K = 1024$, we construct a 32×32 latent grid in which every spatial location contains exactly one unique normalized codeword. The ordering is obtained as follows:

1. **Pairwise cosine dissimilarity:** For normalized codewords $\tilde{e}_i$ and $\tilde{e}_j$, we compute
$$d(i, j) = 1 - \tilde{e}_i^\top \tilde{e}_j.$$
This produces a $K \times K$ pairwise dissimilarity matrix over the learned codebook.
2. **Two-dimensional embedding:** We apply multidimensional scaling (MDS) to the pairwise dissimilarity matrix to obtain a two-dimensional coordinate for each codeword. The resulting coordinates are independently min–max normalized to $[0, 1]^2$.
3. **Grid assignment:** We define a uniform 32×32 target lattice over $[0, 1]^2$. To obtain a one-to-one mapping between the 1024 embedded codewords and the 1024 lattice positions, we compute the squared Euclidean cost between every embedded coordinate and every grid point and solve the resulting linear-sum assignment problem using Hungarian matching.
4. **Decoding:** The resulting assignment defines a spatially ordered latent tensor $Q_{\text{grid}} \in \mathbb{R}^{D \times 32 \times 32}$ containing every normalized codeword exactly once. For MiniPlaces, whose default latent resolution is 32×32, this tensor is decoded directly. For COCO and Tiny ImageNet, whose default latent resolution is 16×16, the 32×32 lattice is partitioned into four non-overlapping 16×16 tiles. Each tile is decoded independently using the frozen decoder Dec_ϕ, and the four reconstructed outputs are assembled into a single visualization.

For visualization, each codeword is additionally assigned a color derived from its two-dimensional MDS coordinates, such that nearby points in the embedding receive similar colors. The resulting grid therefore provides a qualitative view of local organization in the learned codebook, while the decoded output illustrates how the decoder response varies across this imposed spatial ordering. The MDS orientation is arbitrary, so colors and spatial directions should only be interpreted within a dataset and are not directly comparable across datasets. Moreover, this visualization is diagnostic rather than a quantitative measure of codebook utilization or semantic organization; code usage is evaluated separately through perplexity and utilization metrics. We restrict this visualization to single-stage AVQ because individual ARVQ codebooks represent stage-dependent residual corrections; decoding an isolated residual-stage codebook would therefore not correspond to the standard ARVQ inference pathway.

Interpretation: Figure 11 visualizes the geometry of the seed-42 AVQ codebooks learned on COCO, MiniPlaces, and Tiny ImageNet. For each dataset, we first consider the normalized codebook $\tilde{E}$ with $K = 1024$ codewords. Pairwise cosine dissimilarities are embedded into two dimensions using multidimensional scaling (MDS), and the resulting coordinates are assigned one-to-one to a 32×32 lattice using Hungarian matching. The left panel for each dataset shows this MDS-ordered codebook, with colors derived from the two-dimensional coordinates so that nearby colors indicate local proximity within the learned codebook geometry. The right panel shows the response of the frozen decoder Dec_ϕ to the corresponding spatially ordered latent field. MiniPlaces is decoded directly at its native 32×32 latent resolution. For COCO and Tiny ImageNet, whose models use a native 16×16 latent grid, the 32×32 codebook lattice is partitioned into four 16×16 tiles; each tile is decoded independently and the resulting outputs are assembled into a single visualization. Because the orientation of an MDS embedding is arbitrary, colors and spatial directions should be interpreted only within each dataset rather than compared directly across datasets. These visualizations provide a qualitative view of local codebook organization and decoder response, complementing the quantitative utilization and perplexity analyses reported elsewhere in the appendix.

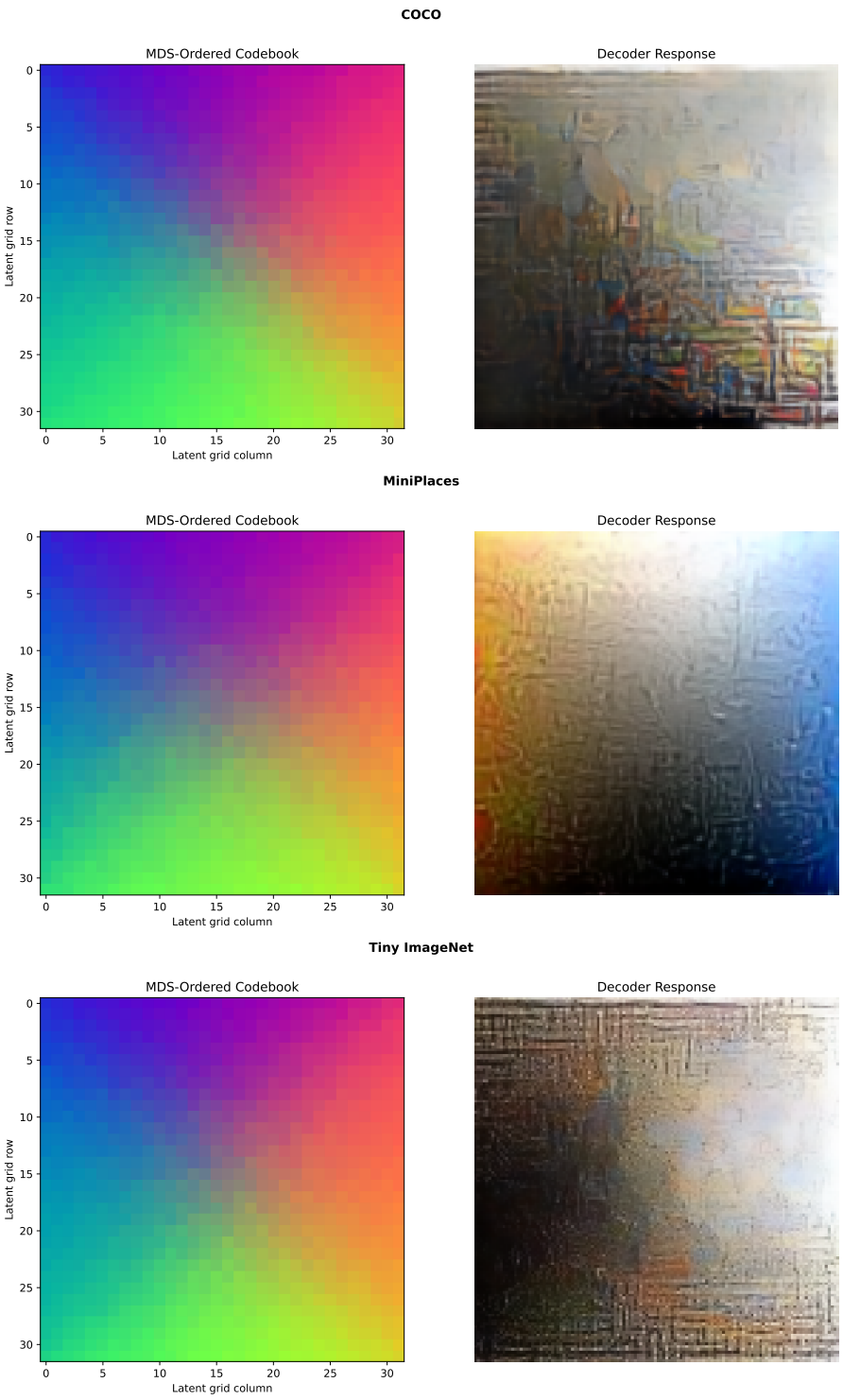

Figure 11: AVQ codebook geometry for the seed-42 models on COCO, MiniPlaces, and Tiny ImageNet. Left: MDS-ordered normalized codebook $\tilde{E}$. Right: response of Dec_ϕ to the corresponding spatially ordered codewords.