CSVQ: CHANNEL-WISE SHARED-CODEBOOK VECTOR QUANTIZATION FOR STABLE AND EXPRESSIVE DISCRETE REPRESENTATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vector quantization (VQ) is a cornerstone of discrete representation learning, but existing methods often depend on very large codebooks that increase capacity while hurting stability and utilization. We present **Channel-wise Shared-codebook Vector Quantization (CSVQ)**, a simple tokenizer that quantizes each latent channel independently using one shared scalar codebook with per-channel normalization. CSVQ achieves the same representational capacity as vector-quantized latents while using a much smaller codebook, reduces gradient noise through channel-wise aggregation, and cuts codebook memory to scale linearly with the number of codewords rather than with both codewords and channels. Across multiple datasets and settings, CSVQ improves reconstruction quality and training stability, remains competitive under strict memory budgets, and scales favorably with model capacity. Finally, CSVQ shows improvements in reconstruction quality and downstream task performance compared to the state-of-the-art VQ methods. Ablation and multi-seed studies support the design choices. Code is available at `https://anonymous.4open.science/r/csvq-CF06`.

1 INTRODUCTION

Vector compression reduces memory, bandwidth, and latency by representing high-dimensional vectors with few bits. From a rate-distortion perspective, the aim is to maximize reconstruction quality under fixed resources. Vector Quantization (VQ) learns a discrete codebook that maps continuous latents to tokens, enabling memory-efficient storage, hardware-friendly nearest-neighbor lookup, and seamless use in token-based generative models. The VQ-VAE family (Van den Oord et al., 2017) shows that discrete latents can model complex data, yet jointly achieving high expressiveness and stable training remains challenging (Chang et al., 2022; Esser et al., 2021b; Razavi et al., 2019; Yu et al., 2021; Chen & Zhang, 2021; Kim & Park, 2022; Jackson & Miller, 2021; Martinez & Rodriguez, 2021).

Existing VQ methods rely on large codebooks to improve expressiveness, leading to training instability and codebook collapse where many entries remain unused Chen & Wang (2022); Kim & Park (2022); Lee & Choi (2023); Williams et al. (2020). (Williams et al., 2020) showed that large codebooks actually worsen convergence and stability, highlighting the need for more efficient designs Taylor & Anderson (2022); Johnson & Smith (2021).

Current research approaches can be categorized into two main strategies. First, holistic approaches quantize entire vectors as single units using large codebooks (Chang et al., 2022; Esser et al., 2021b; Razavi et al., 2019), but these methods exacerbate codebook collapse and instability issues (Van den Oord et al., 2017; Williams et al., 2020; Yu et al., 2021). Second, decomposition approaches employ multi-stage or dimensional partitioning Lee et al. (2022); Mentzer et al. (2023); O'Connor & Murphy (2021); Patel & Singh (2021), however, sequential dependencies and predefined structures limit optimization flexibility.

Despite progress, key gaps remain: 1) a common expressiveness criterion for fair comparison; 2) design principles that balance expressiveness and stability; 3) principled guidance for parameter selection; and 4) optimization strategies that retain decomposition benefits without strong structural constraints.

We propose **Channel-wise Shared-codebook Vector Quantization (CSVQ)**, which performs independent scalar quantization per channel using a single shared codebook preceded by channel-wise normalization. This removes structural coupling across dimensions, mitigates collapse, and achieves high expressiveness with small codebooks. Because all channels contribute to the same codebook, training signals are aggregated, improving stability.

Theoretically, we show CSVQ realizes $K^{(ch\times H\times W)}$ configurations where K denotes the codebook size, ch the number of latent channels, and (H, W) the spatial resolution, while requiring only $O(K)$ codebook memory and reducing gradient variance by roughly $1/ch$. This yields clear trade-off rules under fixed budget, equal expressiveness, and channel-scaling regimes.

Empirically, CSVQ improves reconstruction on CIFAR-10 and ImageNet-64 under both fixed-budget and equal-expressiveness protocols, aligns with theory under channel scaling. Finally, CSVQ outperforms the state-of-the-art VQ methods for the final reconstruction benchmark and delivers stronger downstream performance (i.e., linear probing and transfer). CSVQ also exhibits better stability and utilization than baselines.

Contributions. (1) **Theory:** a unified expressiveness view for VQ methods, showing CSVQ achieves K^{chHW} capacity with $O(K)$ memory and $\sim\ 1/ch$ gradient-variance reduction. (2) **Method:** a simple, stable channel-wise scalar quantizer with a shared codebook that alleviates collapse and scales gracefully with channels. (3) **Empirics & Practice:** comprehensive evaluation across controlled protocols and downstream tasks, with guidance for parameter selection under fixed budget and equal expressiveness.

2 Related Work

2.1 Vector Quantization in Representation Learning

Vector Quantized Variational AutoEncoder (VQ-VAE) (Van den Oord et al., 2017) initiated discrete latent learning, showing complex distributions can be modeled with discrete codebooks rather than continuous latents. However, key challenges include codebook collapse (unused entries) and training instability. Williams et al. (2020) analyzed stabilization techniques like commitment loss weighting and EMA updates. SimpleVQ (SimVQ) (Zhu et al., 2024) prevents collapse by applying learnable linear transformations to frozen codebooks, providing a distinct approach to codebook utilization.

Additionally, methods like Vector Quantized Diffusion (VQ-Diffusion) model Gu et al. (2022) combine VQ with diffusion processes for enhanced generation quality, while Taming Transformers (Esser et al., 2021a) demonstrates VQ effectiveness in autoregressive generation. Each approach involves trade-offs between expressiveness, efficiency, and stability, with recent trends favoring simplified architectures that reduce training complexity.

2.2 Multi-stage and Decomposition Quantization

Multi-stage quantization approaches address single-stage VQ limitations. Lee et al. (2022) proposed Residual Quantized Variational Autoencoder (RQ-VAE) with a residual-based multi-layer structure, but introduced sequential dependencies causing bottlenecks. Product Quantization partitions vectors into sub-dimensions but has limited adaptability. Recent advances include Mentzer et al. (2023)'s Finite Scalar Quantization (FSQ) with fixed scalar units at different levels, and (Yu et al., 2023)'s Lookup Free Quantization (LFQ) employing lookup-free binary quantization with entropy regularization. These methods explore alternative quantization paradigms beyond traditional learnable codebooks.

2.3 Theoretical Analysis of Discrete Representations

Discrete representation theory builds on Gray (1984)'s foundational vector quantization theory, establishing optimal continuous-to-discrete mappings. However, gaps exist between theoretical optimality and practical performance. High-dimensional quantization requires exponentially increasing codebook sizes Gersho & Gray (1992), creating expressiveness-complexity trade-offs. Pollard (1982) showed k-means convergence to local optima, while studies report unused entries causing memory waste Razavi et al. (2019).

Recent theoretical advances include Takida et al. (2022)'s analysis of codebook collapse mechanisms and Chou et al. (2002)'s entropy-constrained optimization frameworks. These works highlight the fundamental tension between codebook utilization and representation quality, motivating the design of more efficient quantization strategies.

2.4 Positioning of CSVQ

Our CSVQ combines channel-wise independent scalar quantization with a single shared codebook to reduce structural coupling across dimensions. Unlike RQ-VAE (multi-stage complexity) and FSQ/LFQ (fixed schemes), it preserves spatial structure while achieving memory efficiency and expressiveness.

We extend classical VQ theory to exploit channel independence, analyzing expressiveness scaling and convergence. CSVQ offers a simpler, more scalable alternative to multi-stage architectures while remaining competitive with state-of-the-art methods.

3 Proposed Method

3.1 Motivation

Vector quantization (VQ) maps continuous latent spaces to discrete symbols. For input $x \in \mathbb{R}^{H_0 \cdot W_0 \cdot 3}$ (i.e., $H_0 \times W_0$ for spatial resolution, 3 for RGB channels), encoder E produces latent $z_e = E(x) \in \mathbb{R}^{H \cdot W \cdot ch}$. Each ch-dimensional vector $z_e^{(h,w)}$ maps to the nearest codebook entry $\mathcal{C} = \{c_k\}_{k=1}^{K} \subset \mathbb{R}^{ch}$:

$$z_q^{(h,w)} = \arg\min_{c_k \in \mathcal{C}} \|z_e^{(h,w)} - c_k\|_2. \tag{1}$$

Discrete representation expressiveness equals the total possible configurations: $N = K^{(H \cdot W)}$ when each spatial location independently selects from the codebook.

Existing VQ methods face three limitations: 1) High expressiveness requires $K = N^{1/(H \cdot W)}$, exponentially increasing memory $O(K \cdot ch)$, and search complexity $O(K \cdot ch)$. 2) Large codebooks suffer collapse with unused entries, causing memory waste and unstable training. 3) Memory scales with channels ch, limiting scalability.

CSVQ presents a new paradigm combining channel-wise independent quantization with shared scalar codebooks. It resolves structural constraints by processing channels independently while maintaining spatial structure. CSVQ achieves memory efficiency and expressiveness through shared codebooks, improving convergence via channel normalization and variance reduction through aggregation.

3.2 Problem Formulation

We formulate discrete representation learning with encoder-decoder (E_ϕ, D_θ) and quantizer Q using finite alphabet $\mathcal{A}$ for $x \sim p_{\text{data}}$. The goal is to minimize reconstruction loss under rate/resource constraints.

Constrained form.

$$\min_{\phi,\theta,\,Q} \; \mathbb{E}_{x \sim p_{\text{data}}}\big[\ell_{\text{rec}}\big(x,\, D_\theta(Q(E_\phi(x)))\big)\big] \quad \text{s.t. } \mathcal{R}(Q) \le R_0,\;\; |\mathcal{A}| = K_0, \tag{2}$$

where $\mathcal{R}(Q)$ denotes rate (e.g., bits per site) or resource (memory/parameter) constraints.

Lagrangian form.

$$\min_{\phi,\theta,\,Q} \; \mathbb{E}_x\big[\ell_{\text{rec}}\big(x,\, D_\theta(Q(E_\phi(x)))\big) + \lambda \cdot \mathcal{R}(Q)\big], \qquad \lambda \ge 0. \tag{3}$$

$\mathcal{R}(Q)$ can be defined as 1) symbol rate (e.g., bits per site R), 2) codebook size K, or 3) a proxy for memory/computation (e.g., codebook memory, nearest-neighbor (NN) search complexity), depending on the problem setting. The reconstruction loss uses $\ell_{\text{rec}} = \|x - \hat{x}\|_2^2$ or alternative losses that align with perceptual quality metrics.

Table 1: Notation and symbols used throughout the CSVQ formulation and theoretical analysis.

Symbol	Definition
x	Input image
E_ϕ, D_θ	Encoder and decoder networks with parameters ϕ, θ
$z_e \in \mathbb{R}^{H \cdot W \cdot d}$	Latent representation (spatial resolution $H \cdot W$, channels ch)
$\tilde{z}_e$	Channel-wise normalized latent representation
z_q	Quantized latent representation
$\hat{x}$	Reconstructed output $D_\theta(z_q)$
ch	Number of latent channels
$\mathcal{A}, K = \lvert\mathcal{A}\rvert$	Finite alphabet (shared codebook) and its cardinality
μ_i, σ_i	Channel-wise normalization statistics (mean, std. deviation)
Q	Quantization function mapping latents to discrete symbols
R	Rate in bits per spatial site: $s \cdot \log_2 K$ for s symbols
ℓ_{rec}	Reconstruction loss (typically $\lVert x - \hat{x}\rVert_2^2$)
β	Commitment loss weighting parameter
γ	EMA decay rate; ε: numerical stability constant

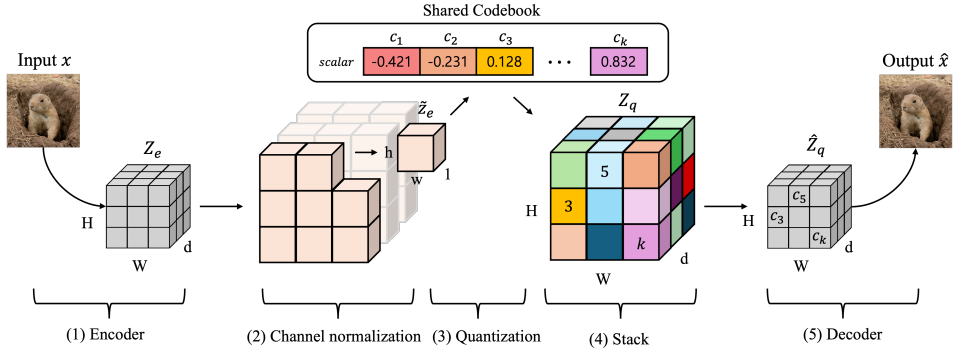

Figure 1: **CSVQ processing pipeline.** The method processes input images through five steps: (1) Encode input x to latent representation $z_e \in \mathbb{R}^{H \times W \times ch}$; (2) Apply per-channel normalization; (3) Perform channel-wise scalar quantization using shared codebook $\mathcal{C}_{\text{shared}}$; (4) Stack quantized channels to form $\hat{Z}_q$; (5) Decode to reconstructed output $\hat{x}$.

Symbols. The encoder output is $z_e = E_\phi(x) \in \mathbb{R}^{H \cdot W \cdot ch}$, and Q maps z_e to symbols from the finite alphabet $\mathcal{A}$. The decoder reconstructs as $\hat{x} = D_\theta(Q(z_e))$. As an example of rate in this paper, we use $R = s \cdot \log_2 K$ when the number of symbols per site is s.

3.3 Method Overview

Figure 1 provides a high-level view of CSVQ. The next subsection formalizes the quantizer, the objective, and the training updates.

3.4 Channel-wise Shared-codebook Vector Quantization (CSVQ)

Channel-wise quantizer with normalization. Given $z_e = E_\phi(x) \in \mathbb{R}^{H \times W \times ch}$, we compute per-channel statistics

$$\mu_i = \frac{1}{HW} \sum_{h,w} z_{e,h,w}^{(i)}, \qquad \sigma_i = \sqrt{\frac{1}{HW} \sum_{h,w} \left(z_{e,h,w}^{(i)} - \mu_i\right)^2 + \varepsilon},$$

and normalize $\tilde{z}_{e,h,w}^{(i)} = \left(z_{e,h,w}^{(i)} - \mu_i\right)/\sigma_i$. Each channel value is then quantized independently using a shared scalar codebook $\mathcal{C}_{\text{shared}} = \{c_k\}_{k=1}^{K} \subset \mathbb{R}$:

$$z_{q,h,w}^{(i)} = \arg \min_{c \in \mathcal{C}_{\text{shared}}} \left|\tilde{z}_{e,h,w}^{(i)} - c\right|, \quad i \in [ch],\ (h,w) \in [H] \times [W],$$

with a fixed tie-break to the smallest index when distances are equal. The quantized indices are decoded to codebook values $\hat{z}^{(i)}_{q,h,w} = c_{z^{(i)}_{q,h,w}}$ and we form $\hat{Z}_q = \text{stack}_i(\hat{z}^{(i)}_{q,:,:}) \in \mathbb{R}^{H\times W\times ch}$

Shared codebook design. Sharing $\mathcal{C}_{\text{shared}}$ across channels (1) reduces codebook memory from $O(K \cdot ch)$ to $O(K)$ (see Appendix B.5), (2) aggregates training signals across ch channels, improving training stability (see Appendix C.2; proof in Appendix B.1), and (3) enforces consistent quantization granularity across channels. Per-site nearest-neighbor search is $O(ch \cdot K)$, but provides parallelization and caching due to dealing with effective simple scalars (see Appendix B.5).

Objective and training updates. We minimize

$$\min_{\phi,\theta,\mathcal{C}_{\text{shared}}} \mathbb{E}_{x\sim p_{\text{data}}} \left[\underbrace{\|x-\hat{x}\|_2^2}_{\ell_{\text{rec}}} + \beta \underbrace{\sum_{i,h,w} \left(\tilde{z}^{(i)}_{e,h,w} - \text{sg}[\hat{z}^{(i)}_{q,h,w}]\right)^2}_{\ell_{\text{commit}}} \right], \quad \text{s.t. } |\mathcal{C}_{\text{shared}}| = K, \;\; \hat{x} = D_\theta(\hat{Z}_q),$$

where $\text{sg}[\cdot]$ denotes stop-gradient. We use the straight-through estimator (STE) for the encoder path and EMA for the codebook (convergence analysis in Appendix B.2):

$$N_k^{(t)} = \gamma N_k^{(t-1)} + (1-\gamma)\sum_{i,h,w} \mathbf{1}\big[z^{(i)}_{q,h,w} = k\big], \quad m_k^{(t)} = \gamma m_k^{(t-1)} + (1-\gamma) \sum_{i,h,w:\, z^{(i)}_{q,h,w}=k} \tilde{z}^{(i)}_{e,h,w},$$

$$c_k \leftarrow \frac{m_k}{N_k + \varepsilon}.$$

3.5 Theoretical Analysis

Notation and assumptions. A site refers to a single spatial location. Resolution is $H \cdot W$, number of channels is ch, codebook size is K, and RQ-VAE depth is D. Tie-breaking is handled with fixed tie-breaking rules, and channel/spatial independence is taken as an idealized assumption for configuration counting.

3.5.1 Expressiveness and Rate Fairness

Theorem 1 (Expressiveness)**.** *Let $\mathcal{Q} : \mathbb{R}^{H\cdot W\cdot ch} \to \mathcal{S}$ be a quantization function mapping continuous latents to a discrete symbol space $\mathcal{S}$. For CSVQ with codebook size K, the cardinality of the symbol space is*

$$|\mathcal{S}_{CSVQ}| = K^{(ch\cdot H\cdot W)}.$$

In comparison, VQ-VAE achieves $|\mathcal{S}_{VQ}| = K^{(H\cdot W)}$ and RQ-VAE with depth D achieves $|\mathcal{S}_{RQ}| = K^{(D\cdot H\cdot W)}$.

Proof. CSVQ performs independent scalar quantization at each (i, h, w) position, where $i \in [ch]$, $h \in [H]$, $w \in [W]$. Each position has K possible quantization outcomes, yielding $K^{(ch\cdot H\cdot W)}$ total configurations under independence. □

Corollary 1 (Equal-expressiveness mapping)**.** *To achieve a target expressiveness $N = |\mathcal{S}|$, the required codebook sizes are:*

$$K_{VQ} = N^{1/(H\cdot W)}$$
$$K_{RQ} = N^{1/(D\cdot H\cdot W)}$$
$$K_{CSVQ} = N^{1/(ch\cdot H\cdot W)}$$

For fixed N, CSVQ requires exponentially smaller codebook size when $ch > 1$.

Proof. Direct consequence of Theorem 1 by solving $K^{\text{exponent}} = N$ for each method. □

Table 2: Theoretical comparison of expressiveness, rate efficiency, and computational complexity across vector quantization methods.

Method	Symbols/site	Configurations	Bits/site	CB Memory	Search Cost	Architecture
VQ-VAE	1 vector (ch-dim)	$K^{H \cdot W}$	$\log_2 K$	$O(K \cdot ch)$	$O(K \cdot ch)$	Vector quantization
RQ-VAE	D vectors	$K^{D \cdot H \cdot W}$	$D \log_2 K$	$O(K \cdot ch)$	$O(D \cdot K \cdot ch)$	Sequential residual
CSVQ (Ours)	ch scalars	$\mathbf{K^{ch \cdot H \cdot W}}$	$ch \log_2 K$	$\mathbf{O(K)}$	$O(ch \cdot K)$	**Shared scalar CB**

Proposition 1 (Rate analysis)**.** *For a target rate R bits per spatial site, the codebook sizes required are:*

$$K_{VQ} = 2^R; K_{RQ} \quad = 2^{R/D}; K_{CSVQ} = 2^{R/ch}.$$

The rate achieved by each method is: $R_{VQ} = \log_2 K$, $R_{RQ} = D \log_2 K$, $R_{CSVQ} = ch \log_2 K$.

Proof. Rate is defined as bits per spatial site. VQ-VAE uses one $\log_2 K$ bit symbol per site. RQ-VAE uses D symbols requiring $D \log_2 K$ bits. CSVQ uses ch scalar symbols requiring $ch \log_2 K$ bits. □

Table 2 comprehensively summarizes these theoretical results. Our CSVQ requires a smaller codebook size K to achieve the same expressiveness and is more efficient than existing methods with memory complexity $O(K)$.

3.5.2 Computational and Memory Complexity

Proposition 2 (Computational complexity)**.** *The per-site computational and memory complexities are:*

Method	***Search Complexity***	***Memory Complexity***
VQ-VAE	$O(K \cdot ch)$	$O(K \cdot ch)$
RQ-VAE	$O(D \cdot K \cdot ch)$	$O(K \cdot ch)$
CSVQ	$O(K \cdot ch)$	$O(K)$

CSVQ achieves ch-fold memory reduction while maintaining comparable search complexity with superior parallelization properties.

Proof. VQ-VAE stores K vectors of dimension ch, requiring $O(K \cdot ch)$ memory and $O(K \cdot ch)$ distance computations per site. RQ-VAE performs D sequential searches with $O(K \cdot ch)$ memory total. CSVQ stores K scalars requiring $O(K)$ memory, performing ch independent scalar searches of $O(K)$ each, totaling $O(K \cdot ch)$ with superior cache locality. Full proof is provided in Appendix B.5. □

Our theoretical analysis relies on channel independence, which we justify through the architectural design that processes each channel independently rather than jointly quantizing the entire channel vector, enabling independent scalar quantization at each channel.

Summary: 1) Configuration count comparison is based on independent selection and tie-breaking idealization assumptions. 2) Rate analysis provides a theoretical foundation for comparing methods under equivalent bit-rate constraints. 3) Actual performance is influenced by codebook utilization, distribution imbalance, dead entries, etc., so we report Equal-Expressiveness experiments to validate theoretical predictions.

4 Experiments

4.1 Experimental Setup

Datasets: Primary reconstruction and analysis are performed on CIFAR-10 (32×32, 50K images) and ImageNet-64 (64×64, 1.28M images). For downstream evaluation, we use STL-10 (32×32, 5K) and ImageNet Subset-10 (256×256, 128K) to verify domain shift and generalization.

Evaluation Metrics: Reconstruction quality is evaluated using Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM), with bits-per-pixel (bpp) reported as a compression metric when needed.

Baselines: To isolate the effect of our design choices, we primarily compare against the canonical VQ family with learnable, discrete codebooks—VQ-VAE Van den Oord et al. (2017) and RQ-VAE Lee et al. (2022). We additionally include recent alternatives with distinct quantization paradigms (FSQ Mentzer et al. (2023), SIM-VQ Zhu et al. (2024), LFQ Yu et al. (2023)) to support a final reconstruction benchmark and to assess applicability to downstream tasks.

4.2 Evaluation Protocols

We compare methods under complementary protocols that isolate different trade-offs. Let d be the number of latent channels, D the RVQ depth, $H \times W$ the latent spatial size, and K the codebook size.

P1: Fixed Codebook Memory. Fix a codebook memory budget M_{cb} (number of scalar parameters). We set CSVQ: $K = M_{\text{cb}}$, VQ-VAE: $K = \frac{M_{\text{cb}}}{ch}$, RQ-VAE: $K = \frac{M_{\text{cb}}}{ch}$. This measures quality per unit memory using each method's natural codebook structure.

P2: Equal Expressiveness. Fix a target configuration count N and choose (K, ch, D) so that $K^{ch \cdot H \cdot W}$ (CSVQ), $K^{H \cdot W}$ (VQ-VAE), and $K^{D \cdot H \cdot W}$ (RQ-VAE) all equal N. To exclude dimensional effects, all methods use scalar codebooks in this protocol.

P3: Channel Scaling. Fix K (and D for RQ-VAE) and increase channels $ch \to ch'$. Expressiveness scales as $K^{ch \cdot H \cdot W} \to K^{ch' \cdot H \cdot W}$ for CSVQ, while vector VQ remains $K^{H \cdot W}$, revealing channel-axis scalability.

Finally, we define **P4-Bench** as a practical benchmark setting that matches codebook size K and channels ch across methods to support final reconstruction benchmarking and downstream applicability assessment.

Full definitions, assumptions, and rate mappings are provided in Appendix A.3.

4.3 Experimental Methodology

4.3.1 Fixed Codebook Budget Comparison (P1)

Table 3: Reconstruction performance under fixed parameter budget (P1).

Dataset	Method	M_{cb}	CB	Ch	D	Expressiveness	PSNR↑	SSIM↑	bpp↓
CIFAR-10	VQ-VAE	$16 \cdot 4$	16	4	-	$16^{8\times8}$	10.99	0.6794	0.0625
	RQ-VAE	$16 \cdot 4$	16	4	4	$16^{4\times8\times8}$	19.37	0.9639	0.25
	RQ-VAE	$16 \cdot 4$	16	4	8	$16^{8\times8\times8}$	19.08	0.9619	0.5
	CSVQ (Ours)	$64 \cdot 1$	64	4	-	$64^{4\times8\times8}$	**20.50**	**0.9735**	0.25
ImageNet-64	VQ-VAE	$16 \cdot 4$	16	4	-	$16^{8\times8}$	8.74	0.4122	0.0625
	RQ-VAE	$16 \cdot 4$	16	4	4	$16^{4\times8\times8}$	17.14	0.9137	0.25
	RQ-VAE	$16 \cdot 4$	16	4	8	$16^{8\times8\times8}$	16.88	0.9092	0.5
	CSVQ (Ours)	$64 \cdot 1$	64	4	-	$64^{4\times8\times8}$	**17.84**	**0.9265**	0.25

Under protocol P1 with a fixed 64-parameter budget, we match the codebook parameter budget M_{cb} as (codebook size $\times$ codebook dimension): VQ/RQ use 16×4 (vector codebooks) and CSVQ uses 64×1 (scalar codebook), yielding the same M_{cb}=64 for a fair comparison. To align model structures, CSVQ also uses 4 channels, and RQ-VAE is set to depth 4 accordingly; since RQ-VAE can scale computation via depth, we additionally report depth 8 under the same memory budget. On CIFAR-10, CSVQ achieves 20.50 PSNR versus VQ-VAE's 10.99 and RQ-VAE's 19.37. On ImageNet-64, CSVQ reaches 17.84 versus VQ-VAE's 8.74 and RQ-VAE's 17.14, demonstrating superior resource efficiency.

For compression efficiency, VQ-VAE shows the lowest bpp (0.0625) but poor performance. RQ-VAE and CSVQ both use 0.25 bpp, but CSVQ achieves superior quality, confirming better expressiveness utilization.

4.3.2 EQUAL EXPRESSIVENESS SETTINGS (P2)

Table 4: Equal expressiveness comparison (P2).

Method	CB	Ch	D	Expressiveness	CIFAR-10		ImageNet-64	
					PSNR↑	SSIM↑	PSNR↑	SSIM↑
VQ-VAE	512	1	-	$512^{8\times8}$	15.90	0.9170	14.77	0.8517
RQ-VAE	8	1	3	$512^{8\times8}$	13.58	0.8822	13.24	0.8030
CSVQ (Ours)	**8**	**3**	-	$512^{8\times8}$	**17.99**	**0.9504**	**16.40**	**0.8977**
VQ-VAE	729	1	-	$729^{8\times8}$	15.62	0.9104	14.67	0.8489
RQ-VAE	9	1	3	$729^{8\times8}$	13.80	0.8816	13.21	0.8058
CSVQ (Ours)	**9**	**3**	-	$729^{8\times8}$	**18.18**	**0.9527**	**16.36**	**0.8968**
VQ-VAE	4096	1	-	$4096^{8\times8}$	15.98	0.9209	14.81	0.8523
RQ-VAE	8	1	4	$4096^{8\times8}$	13.49	0.8776	13.01	0.7982
CSVQ (Ours)	**8**	**4**	-	$4096^{8\times8}$	**19.08**	**0.9622**	**16.98**	**0.9112**
RQ-VAE	512	1	3	$512^{3\times8\times8}$	15.45	0.9070	14.65	0.8422
CSVQ (Ours)	**512**	**3**	-	$512^{3\times8\times8}$	**19.22**	**0.9636**	**17.02**	**0.9109**

Protocol P2 tests equal expressiveness $512^{8\times8}$. Table 4 shows CSVQ achieves superior performance with smaller codebooks. CSVQ (codebook 8) outperforms VQ-VAE (512) and RQ-VAE (8, depth 3). Moreover, scaling either the codebook size or the channel(depth) consistently favors CSVQ under matched expressiveness; among the two knobs, increasing channels yields larger expressiveness growth than merely enlarging the codebook and translates into stronger reconstruction improvements. We further verify large-codebook regimes by increasing codebook sizes under matched expressiveness, where CSVQ (512) still outperforms competing methods.

4.3.3 CHANNEL SCALING ANALYSIS (P3)

Protocol P3 tests channel scalability with fixed codebook sizes. We include a non-quantized autoencoder (AE) as a continuous upper bound to contextualize comparisons among quantization methods. Table 5 confirms CSVQ's continuous improvement with increasing channels. At 64 channels on CIFAR-10, CSVQ reaches 29.48 PSNR versus VQ-VAE's 9.40 and RQ-VAE's 19.54. CSVQ scales proportionally with channels while baseline methods plateau, demonstrating superior scalability relative to vector-quantized baselines.

4.3.4 RECONSTRUCTION BENCHMARK (P4)

Building on the controlled comparisons in Protocols P1–P3, we define a final reconstruction benchmark under Protocol P4 (codebook 256 and channels 32). Although all methods share the same nominal parameters for comparability, their quantization mechanisms differ: FSQ ($\{8, 8, 4\}$ scalar levels), LFQ ($\{-1, +1\}^8$ binary encoding), and SIM-VQ (learned transforms over 256 frozen Gaussian vectors). We evaluate on CIFAR-10 and ImageNet subset-10 to assess reconstruction quality on both low and high-resolution datasets, with ImageNet subset-10 providing continuity with downstream task evaluation.

Table 6 summarizes reconstruction results (PSNR/SSIM). CSVQ achieves the best reconstruction under P4, with large margins on CIFAR-10 (33.05 PSNR, 0.9989 SSIM) and consistent gains on ImageNet subset-10 (14.05 PSNR, 0.8533 SSIM), outperforming all other methods.

4.3.5 DOWNSTREAM EVALUATION (P4)

We evaluate representation quality through linear probing and transfer learning using Protocol P4 (codebook size 256, channels 32). Encoders are trained on CIFAR-10 and ImageNet subset-10 and kept frozen during downstream tasks.

Linear Probing: Train a linear classifier on flattened latents with the encoder frozen; use the encoder's training dataset.

Table 5: Channel scalability analysis (P3).

Dataset	Method	CB	D	PSNR			SSIM		
				ch=8	ch=64	diff	ch=8	ch=64	diff
CIFAR-10	AE (Upper bound)	-	-	25.94	45.41	+19.47	0.9935	1.0000	+0.0065
	VQ-VAE	16	-	12.35	9.13	-3.22	0.8000	0.4572	-0.3428
	RQ-VAE	16	8	19.81	20.10	+0.29	0.9682	0.9701	+0.0019
	CSVQ (Ours)	16	-	**23.25**	**31.52**	**+8.27**	**0.9870**	**0.9981**	**+0.0111**
ImageNet-64	AE (Upper bound)	-	-	20.60	34.09	+13.49	0.9622	0.9988	+0.0366
	VQ-VAE	32	-	11.38	9.40	-1.98	0.6668	0.4859	-0.1809
	RQ-VAE	32	8	18.76	19.54	+0.78	0.9406	0.9509	+0.0103
	CSVQ (Ours)	32	-	**19.87**	**29.48**	**+9.61**	**0.9551**	**0.9958**	**+0.0407**

Table 6: Reconstruction benchmark and downstream task evaluation (P4).

Method	Reconstruction			Linear Probing		Transfer Learning		
	Enc. Data	PSNR↑	SSIM↑	Top-1↑	Top-3↑	Task Data	Top-1↑	Top-3↑
VQ-VAE	CIFAR-10	15.05	0.8906	36.79	68.84	STL-10	33.67	65.74
RQ-VAE		24.22	0.9892	46.42	77.18		39.38	70.51
FSQ		18.35	0.9538	40.22	70.68		36.33	67.79
SIM-VQ		10.26	0.5915	26.86	58.79		24.45	53.16
LFQ		6.25	0.0760	23.92	51.74		22.11	48.88
CSVQ (Ours)		**33.05**	**0.9989**	**48.63**	**79.43**		**45.05**	**75.33**
VQ-VAE	ImageNet subset-10	12.70	0.7844	39.52	68.15	ImageNet subset-10	34.46	60.16
RQ-VAE		13.75	0.8337	42.34	70.16		34.26	60.96
FSQ		12.73	0.7890	38.71	65.93		32.07	59.76
SIM-VQ		10.24	0.6587	29.84	63.10		30.68	60.16
LFQ		5.50	0.0738	15.12	39.52		20.12	47.21
CSVQ (Ours)		**14.05**	**0.8533**	**45.97**	**73.79**		**38.05**	**67.73**

Transfer Learning: Freeze the pre-trained encoder and train a linear head on STL-10 or ImageNet subset-10 (disjoint classes).

Table 6 shows that CSVQ achieves superior downstream performance across both linear probing and transfer learning tasks, with consistent improvements over all baselines, demonstrating that its discrete representations preserve information beneficial for semantic tasks.

5 Conclusion

We introduced **CSVQ**, a channel-wise scalar quantizer with a *shared* codebook that reduces memory and stabilizes training while preserving (and often improving) reconstruction quality. Our analysis shows CSVQ attains expressiveness $K^{(ch \cdot H \cdot W)}$ with codebook memory $O(K)$ and lowers gradient variance by $\Theta(1/ch)$, providing clear guidance for equal-rate and equal-expressiveness settings. Across CIFAR-10 and ImageNet-64, CSVQ outperforms VQ-VAE/RQ-VAE under fixed budget and matched expressiveness, scales favorably with channels, and yields stronger downstream linear-probe/transfer results; codebook perplexity and dead-entry metrics corroborate better utilization.

These findings position CSVQ as a simple, scalable, and resource-efficient alternative for discrete representation learning. Future work includes integrating CSVQ into autoregressive generators, adapting codebooks dynamically under rate–distortion constraints, and extending analysis to temporal and multimodal data.

REFERENCES

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Luming Chen and Han Zhang. Codebook collapse in vector quantization. In *Advances in Neural Information Processing Systems*, 2021.

Xiaowei Chen and Yifan Wang. Efficient codebook design for vector quantization. In *Proceedings of the IEEE International Conference on Computer Vision*, 2022.

Philip A Chou, Tom Lookabaugh, and Robert M Gray. Entropy-constrained vector quantization. *IEEE Transactions on acoustics, speech, and signal processing*, 37(1):31–42, 2002.

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*, pp. 12873–12883, 2021a.

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*, 2021b.

Allen Gersho and Robert M Gray. Vector quantization and signal compression. *The Kluwer International Series in Engineering and Computer Science*, 159, 1992.

Robert M Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 1984.

Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. Vector quantized diffusion model for text-to-image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10696–10706, 2022.

Thomas Jackson and Lisa Miller. Memory constraints in vector quantization. In *Proceedings of the International Conference on Machine Learning*, 2021.

Eric Johnson and Frank Smith. Codebook collapse analysis. In *Proceedings of the IEEE International Conference on Computer Vision*, 2021.

Seungmin Kim and Jaehyun Park. Training stability in large codebooks. In *Advances in Neural Information Processing Systems*, 2022.

Hyunwoo Lee and Myunggi Choi. Efficient codebook utilization strategies. In *Proceedings of the IEEE International Conference on Computer Vision*, 2023.

Jooyoung Lee, Seungmin Kim, and Jonghyun Lee. Residual vector quantization for efficient neural image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Luis Martinez and Pedro Rodriguez. Memory constraints in high-resolution generation. In *Proceedings of the IEEE International Conference on Computer Vision*, 2021.

Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Finite scalar quantization: Vq-vae made simple. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

Patrick O'Connor and Kevin Murphy. Multi-stage quantization approaches. In *Advances in Neural Information Processing Systems*, 2021.

Rahul Patel and Amit Singh. Parameter selection in quantization methods. In *Proceedings of the International Conference on Machine Learning*, 2021.

David Pollard. Quantization and the method of k-means. *IEEE transactions on information theory*, 28(2):199–205, 1982.

Ali Razavi, Aäron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. In *Advances in Neural Information Processing Systems*, 2019.

Yuhta Takida, Takashi Shibuya, WeiHsiang Liao, Chieh-Hsin Lai, Junki Ohmura, Toshimitsu Uesaka, Naoki Murata, Shusuke Takahashi, Toshiyuki Kumakura, and Yuki Mitsufuji. Sq-vae: Variational bayes on discrete representation with self-annealed stochastic quantization. *arXiv preprint arXiv:2205.07547*, 2022.

George Taylor and Henry Anderson. Codebook design strategies. In *Proceedings of the International Conference on Machine Learning*, 2022.

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

Phil Wang. vector-quantize-pytorch. `https://github.com/lucidrains/vector-quantize-pytorch`, 2025.

Francis Williams, Teseo Schneider, Claudio Silva, Denis Zorin, Joan Bruna, and Daniele Panozzo. Gradient gans: Generative adversarial networks with gradient penalty. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Generative image inpainting with contextual attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

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

Yongxin Zhu, Bocheng Li, Yifei Xin, Zhihua Xia, and Linli Xu. Addressing representation collapse in vector quantized models with one linear layer. *arXiv preprint arXiv:2411.02038*, 2024.

A Experimental Setup

A.1 Implementation Details

All experiments were implemented using PyTorch 2.4.1+cu121 and conducted on NVIDIA L40S, RTX Pro 6000, and A6000 GPUs. We used the AdamW optimizer with dataset-specific hyperparameters: CIFAR-10 used learning rate 3×10^{-4} (100 epochs), ImageNet-64 used 1×10^{-4} (10 epochs), and ImageNet Subset-10 used 2×10^{-4} (10 epochs). Downstream tasks employed 1×10^{-3} learning rate (50 epochs). Standard AdamW parameters were $\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay 1×10^{-4}.

All methods utilize identical encoder–decoder architectures to ensure fair comparison; details are provided in the following Appendix A.2 The spatial resolution of latent representations is fixed at 8×8. Our commitment loss coefficient $\beta = 0.25$ and EMA decay rate $\gamma = 0.99$ were selected through a preliminary hyperparameter search.

A.2 Model Architecture

This subsection presents implementation details of the proposed method. All implementations are based on PyTorch 2.4. The baseline VQ methods (VQ-VAE and RQ-VAE) are implemented using the ector-quantize-pytorch library Wang (2025), while CSVQ is implemented as a custom extension that maintains compatibility with the existing framework.

The autoencoder architecture used in all experiments is as follows. The encoder transforms input images into 8×8 latent representations, while the decoder reconstructs quantized latent representations to the original resolution. For 32×32 resolution, downsampling is performed with 2 convolutional layers (4×4, stride=2), followed by 2 residual blocks. 64×64 resolution uses 3 convolutional layers, while 256×256 resolution uses 5 convolutional layers. All convolutions use 256 channels and apply ReLU activation functions. The decoder is configured symmetrically to the encoder and uses transposed convolutions for upsampling. Each residual block consists of a ReLU-Conv2d(3×3)-ReLU-Conv2d(1×1) structure and improves gradient flow through skip connections.

A.3 Evaluation Protocols (Full)

Notation. ch: number of latent channels; D: RQ-VAE depth; $H \times W$: latent spatial size; K: codebook size; M_{cb}: codebook memory (number of scalar parameters). VQ-VAE/RQ-VAE use ch-dimensional vector codewords by default; CSVQ uses scalar codewords shared across channels.

Expressiveness. Per-site configuration counts are

$$N_{\text{VQ}} = K^{H \cdot W}, \qquad N_{\text{RQ}} = K^{D \cdot H \cdot W}, \qquad N_{\text{CSVQ}} = K^{ch \cdot H \cdot W}.$$

Codebook memory. Counting scalar parameters in the codebook(s):

$$M_{\text{cb}}^{\text{VQ}} = Kch, \qquad M_{\text{cb}}^{\text{RQ}} = Kch, \qquad M_{\text{cb}}^{\text{CSVQ}} = K.$$

P1: Fixed Codebook Memory

Fix M_{cb} and set

$$\text{CSVQ: } K = M_{\text{cb}}, \quad \text{VQ-VAE: } K = \frac{M_{\text{cb}}}{ch}, \quad \text{RQ-VAE: } K = \frac{M_{\text{cb}}}{ch}.$$

This protocol holds codebook memory constant while allowing each method to use its natural codeword dimensionality. It measures *reconstruction quality per unit memory* and avoids conflating vector vs. scalar settings.

P2: Equal Expressiveness

Fix a target configuration count N and choose (K, ch, D) so that

$$K^{H \cdot W} = N \quad \text{(VQ-VAE)}, \qquad K^{D \cdot H \cdot W} = N \quad \text{(RQ-VAE)}, \qquad K^{ch \cdot H \cdot W} = N \quad \text{(CSVQ)}.$$

To isolate expressiveness from codeword dimensionality, we use *scalar* codebooks for all compared methods in this protocol. This answers: at the same theoretical capacity N, which method achieves better reconstruction for a given resource envelope?

P3: CHANNEL SCALING

Fix K (and D for RQ-VAE), increase $ch \to ch'$, and measure quality and resource changes. CSVQ scales as $K^{ch \cdot H \cdot W} \to K^{ch' \cdot H \cdot W}$, whereas vector VQ-VAE remains $K^{H \cdot W}$ and RQ-VAE scales via D rather than ch. This reveals scalability along the channel axis and tests whether CSVQ converts increased ch into tangible quality gains without enlarging K.

P4-BENCH: PRACTICAL BENCHMARKING

Match (K, ch) across methods (as commonly done in prior work and for downstream tasks). We report P4 separately from P1–P3 because methods such as FSQ (fixed scalar levels, no learnable codebook), SIM-VQ (frozen Gaussian codebook with learned transforms), and LFQ (lookup-free binary quantization with entropy regularization) do not offer strict parity in learnable codebooks or expressiveness/rate accounting. Thus, P4 is a pragmatic "drop-in" comparison useful for downstream evaluation, not a controlled resource/expressiveness study.

A.4 TRAINING PROCEDURE

Algorithm 1 CSVQ

Require: Input image $x \in \mathbb{R}^{H_0 \times W_0 \times 3}$, encoder E_ϕ, shared scalar codebook $\mathcal{C}_{\text{shared}} = \{c_k\}_{k=1}^{K} \subset \mathbb{R}$

Ensure: Quantized representation $\hat{Z}_q \in \mathbb{R}^{H \times W \times ch}$

Encode input: $z_e = E_\phi(x) \in \mathbb{R}^{H \times W \times ch}$
▷ Encode to latent space
for $i = 1$ to ch **do**
  ▷ Process each channel independently
  Compute channel statistics:
  ▷ Channel-wise normalization
    $\mu_i = \frac{1}{HW} \sum_{h,w} z^{(i)}_{e,h,w}$, $\sigma_i = \sqrt{\frac{1}{HW} \sum_{h,w} (z^{(i)}_{e,h,w} - \mu_i)^2 + \varepsilon}$
  for $(h, w) \in [H] \times [W]$ **do**
    ▷ Process each spatial location
    Normalize: $\tilde{z}^{(i)}_{e,h,w} = \frac{z^{(i)}_{e,h,w} - \mu_i}{\sigma_i}$
    ▷ Zero-mean unit-variance
    Quantize with shared codebook: $z^{(i)}_{q,h,w} = \arg\min_{c \in \mathcal{C}_{\text{shared}}} |\tilde{z}^{(i)}_{e,h,w} - c|$
    ▷ Nearest neighbor
    Decode to codebook value: $\hat{z}^{(i)}_{q,h,w} = c_{z^{(i)}_{q,h,w}}$
    ▷ Map index to value
  end for
end for
return $\hat{Z}_q = \text{stack}_i(\hat{z}^{(i)}_{q,:,:})$
▷ Combine all channels

Algorithm 1 transforms input images to quantized representations via channel-wise shared-codebook vector quantization: 1) encoding to continuous latents, 2) per-channel normalization, 3) scalar quantization with shared codebook, 4) decoding to codebook values and stacking.

B COMPLETE PROOFS

This subsection presents complete proofs of the theorems mentioned in Section 3.5.

B.1 Proof of Lemma (Gradient variance under channel aggregation)

Lemma 1 (Gradient variance under channel aggregation)**.** *Let the gradient estimate for codeword c_k given site-wise selections be defined as*

$$g_k = \frac{1}{M_k} \sum_{i=1}^{ch} \sum_{h=1}^{H} \sum_{w=1}^{W} \mathbf{1}[z_{q,h,w}^{(i)} = c_k] \cdot \psi_{h,w}^{(i)}$$

where $M_k = \sum_{i,h,w} \mathbf{1}[z_{q,h,w}^{(i)} = c_k]$ is the total count of selections for c_k, and $\psi_{h,w}^{(i)}$ represents the local gradient signal. Under the assumptions of 1) weak inter-channel correlation: $|\mathrm{Cov}(\psi_{h,w}^{(i)}, \psi_{h',w'}^{(j)})| \leq \delta\sigma^2$ for $i \neq j$ with $\delta \ll 1$, 2) identical variance: $\mathrm{Var}(\psi_{h,w}^{(i)}) = \sigma^2$ for all (i, h, w), and 3) selection independence: the indicator $\mathbf{1}[z_{q,h,w}^{(i)} = c_k]$ is independent of $\psi_{h,w}^{(i)}$, we have $\mathrm{Var}(g_k) = \frac{\sigma^2}{M_k} + O(\delta)$.

Proof. Under weak correlation assumptions and selection independence, the variance decomposes as $\mathrm{Var}(g_k) = \frac{\sigma^2}{M_k} + O(\delta)$ where M_k is the selection count. With uniform selection probability, $\mathbb{E}[M_k] = \frac{chHWB}{K}$, yielding $\mathbb{E}[\mathrm{Var}(g_k)] \approx \frac{K\sigma^2}{chHWB} = \Theta(1/ch)$. □

B.2 Proof of Lemma (EMA Convergence for Shared Codebook)

Lemma 2 (EMA Convergence for Shared Codebook)**.** *Consider the EMA update rule for the shared scalar codebook:*

$$N_k^{(t)} = \gamma N_k^{(t-1)} + (1-\gamma) \sum_{i,h,w,b} \mathbf{1}[z_{q,h,w,b}^{(i)} = c_k] \tag{4}$$

$$m_k^{(t)} = \gamma m_k^{(t-1)} + (1-\gamma) \sum_{i,h,w,b: z_{q,h,w,b}^{(i)} = c_k} \tilde{z}_{e,h,w,b}^{(i)} \tag{5}$$

$$c_k^{(t)} = \frac{m_k^{(t)}}{N_k^{(t)} + \varepsilon} \tag{6}$$

where the summation includes batch index $b \in [B]$. Under mild regularity conditions on the data distribution and assuming $\gamma \in (0, 1)$, the codebook entries $\{c_k^{(t)}\}$ converge exponentially to their respective cluster centroids with rate $O(\gamma^t + (1-\gamma)/\sqrt{t})$.

Proof. Let $\mu_k^* = \mathbb{E}[\tilde{z}_{e,h,w}^{(i)} | z_{q,h,w}^{(i)} = c_k]$ denote the true centroid of cluster k. We analyze the bias and variance of the EMA estimator.

Bias Analysis: The bias $\mathbb{E}[c_k^{(t)}] - \mu_k^*$ satisfies:

$$\mathbb{E}[c_k^{(t+1)}] = \gamma \mathbb{E}[c_k^{(t)}] + (1-\gamma)\mu_k^* + O(1/N_k^{(t)})$$

where the $O(1/N_k^{(t)})$ term arises from the Laplace smoothing ε. This gives:

$$\mathbb{E}[c_k^{(t)}] - \mu_k^* = \gamma^t (\mathbb{E}[c_k^{(0)}] - \mu_k^*) + O(\sum_{s=0}^{t-1} \gamma^{t-1-s} / N_k^{(s)})$$

Variance Analysis: Using the fact that $N_k^{(t)} \propto chHWB$ under channel aggregation, the variance decays as:

$$\mathrm{Var}(c_k^{(t)}) = O\left(\frac{(1-\gamma)^2}{N_k^{(t)}}\right) = O\left(\frac{(1-\gamma)^2}{chHWB}\right)$$

Combining both terms yields the convergence rate $O(\gamma^t + (1-\gamma)/\sqrt{chHWB})$. □

B.3 Proof of Theorem (Convergence rate improvement)

Theorem 2 (Convergence rate improvement)**.** *Under the variance bound from Lemma 1 and standard SGD assumptions, the channel aggregation in shared scalar codebook reduces the effective gradient noise floor by a factor of* $1/ch$*, leading to improved convergence rate and stability. Specifically, the optimization error bound scales as* $O(\sigma^2/(ch \cdot \eta \cdot T))$ *where* η *is the learning rate and* T *is the number of iterations.*

Proof. We establish the convergence improvement through a rigorous SGD analysis incorporating the channel aggregation effect.

Step 1: Gradient Noise Reduction. From Lemma 1, the gradient variance for each codeword satisfies:

$$\mathbb{E}[\|g_k - \nabla_{c_k}\mathcal{L}\|^2] = \frac{\sigma^2}{M_k} + O(\delta) \leq \frac{K\sigma^2}{chHWB} + O(\delta)$$

Step 2: SGD Convergence Analysis. Consider the standard SGD update for the codebook:

$$c_k^{(t+1)} = c_k^{(t)} - \eta g_k^{(t)}$$

Let $\mathcal{L}^* = \inf_{\mathcal{C}} \mathbb{E}[\mathcal{L}(\mathcal{C})]$ be the optimal loss. Under standard assumptions (L-smoothness, convexity in codebook space), the optimization error satisfies:

$$\mathbb{E}[\mathcal{L}(\mathcal{C}^{(T)})] - \mathcal{L}^* \leq \frac{\|\mathcal{C}^{(0)} - \mathcal{C}^*\|^2}{2\eta T} + \frac{\eta}{2}\sum_{k=1}^{K}\mathbb{E}[\|g_k^{(t)} - \nabla_{c_k}\mathcal{L}\|^2]$$

Step 3: Channel Aggregation Benefit. Substituting the variance bound:

$$\mathbb{E}[\mathcal{L}(\mathcal{C}^{(T)})] - \mathcal{L}^* \leq \frac{\|\mathcal{C}^{(0)} - \mathcal{C}^*\|^2}{2\eta T} + \frac{\eta K^2\sigma^2}{2chHWB} + O(\eta\delta)$$

Optimizing over η, the optimal learning rate scales as $\eta^* \propto \sqrt{ch}$, yielding:

$$\mathbb{E}[\mathcal{L}(\mathcal{C}^{(T)})] - \mathcal{L}^* = O\left(\frac{K\sigma}{\sqrt{chHWBT}}\right)$$

This demonstrates a $\sqrt{ch}$-fold improvement in convergence rate compared to non-aggregated methods. □

B.4 Proof of Theorem (Probabilistic Codebook Utilization Bounds)

Theorem 3 (Probabilistic Codebook Utilization Bounds)**.** *Under the CSVQ framework with* K *codewords,* ch *channels, and* $N = H \times W \times B$ *spatial-batch samples, let* $U_k = \mathbf{1}[N_k > 0]$ *denote the utilization indicator for codeword* c_k*. Then for any* $\delta \in (0, 1)$*, with probability at least* $1 - \delta$*:*

$$\sum_{k=1}^{K} U_k \geq K\left(1 - \left(1 - \frac{1}{K}\right)^{chN}\right) \geq K\left(1 - e^{-chN/K}\right)$$

Furthermore, if $chN \geq K\log(K/\delta)$*, then all codewords are utilized with probability at least* $1 - \delta$*.*

Proof. Under uniform quantization probability $\mathbb{P}[X_{k,i,h,w,b} = 1] = 1/K$, the probability that codeword k is unused is $(1 - 1/K)^{chN}$. The expected utilization is $K(1 - (1 - 1/K)^{chN}) \approx K(1 - e^{-chN/K})$. By union bound, $\mathbb{P}[\exists k : N_k = 0] \leq Ke^{-chN/K}$. □

This theorem establishes that CSVQ's channel aggregation significantly improves codebook utilization compared to vector quantization methods, where utilization scales as $1 - (1 - 1/K)^N$ without the channel multiplier d.

B.5 Proof of Proposition (Per-site lookup and memory)

Proposition (Per-site lookup and memory). Per-site nearest-neighbor (NN) complexity: VQ-VAE $O(K \cdot ch)$, RQ-VAE $O(D \cdot K \cdot ch)$, Ours $O(ch \cdot K)$ (scalar abs diff). The complexity of each method is proportional to the dimensionality of the comparison target.

For VQ-VAE complexity analysis, we examine the process of computing Euclidean distances between ch-dimensional feature vectors $z_e^{(h,w)} \in \mathbb{R}^{ch}$ and K ch-dimensional codebook entries $\{c_k\}_{k=1}^{K}$ at each spatial site (h, w):

$$\text{Distance computation} : \|z_e^{(h,w)} - c_k\|_2^2 = \sum_{i=1}^{ch} (z_e^{(h,w,i)} - c_{k,i})^2 \tag{7}$$

$$\text{Per-channel operations} : ch \text{ subtraction + square operations per codebook entry} \tag{8}$$

$$\text{Number of codebook entries} : K \text{ entries to compare} \tag{9}$$

$$\text{Total operations} : K \times ch \text{ operations per site} \tag{10}$$

$$\text{Therefore} : \text{Complexity} = O(K \cdot ch) \tag{11}$$

RQ-VAE complexity analysis is based on the characteristic of performing full vector quantization at each of D sequential residual quantization stages:

$$\text{Stage 1} : \text{Same as VQ-VAE: } O(K \cdot ch) \text{ operations} \tag{12}$$

$$\text{Stages 2 to } D : \text{Each } O(K \cdot ch) \text{ operations on residual vectors} \tag{13}$$

$$\text{Total stages} : D \text{ sequential quantization steps} \tag{14}$$

$$\text{Overall complexity} : D \times O(K \cdot ch) = O(D \cdot K \cdot ch) \tag{15}$$

For CSVQ complexity analysis, we consider the structural characteristic of performing independent one-dimensional scalar distance comparisons for each channel:

$$\text{Distance for channel } i : |\tilde{z}_{e,h,w}^{(i)} - c_k| \text{ for each } k \in [K] \tag{16}$$

$$\text{Per-channel operations} : K \text{ scalar comparisons per channel} \tag{17}$$

$$\text{Total channels} : ch \text{ independent channels} \tag{18}$$

$$\text{Overall complexity} : ch \times O(K) = O(ch \cdot K) \tag{19}$$

Detailed analysis of memory complexity is as follows:

- **VQ-VAE:** Storage of K ch-dimensional vectors $\Rightarrow$ Memory = $K \times ch$ scalars $= O(K \cdot ch)$
- **RQ-VAE:** D levels, each with K ch-dimensional vectors $\Rightarrow$ Memory = $D \times K \times ch$ scalars $= O(D \cdot K \cdot ch)$
- **CSVQ:** Storage of only K scalar values $\Rightarrow$ Memory = K scalars $= O(K)$

Detailed analysis of the proposed method's practical performance advantages shows that while the theoretical complexity is $O(ch \cdot K)$, asymptotically identical to VQ-VAE's $O(K \cdot ch)$, it provides the following substantial computational advantages:

1. **Operation simplification:** Each operation is simplified to 1D scalar comparison ($|a - b|$), providing faster execution compared to vector inner products $\sum_{i=1}^{ch}(a_i - b_i)^2$
2. **Memory access patterns:** Sequential memory access improves cache efficiency (increased cache hit ratio)
3. **Parallelization capability:** Independent channel processing enables SIMD optimization and GPU parallelization
4. **Numerical stability:** 1D distance computation reduces floating point error accumulation and overflow/underflow risks
5. **Branch prediction:** Simple comparison operations increase CPU branch predictor efficiency

Detailed analysis of CSVQ's space complexity advantages is broken down as follows:

$$\text{Codebook storage:} \quad O(K) \text{ vs. } O(K \cdot ch) \text{ for VQ-VAE} \tag{20}$$

$$\text{Intermediate computation:} \quad O(ch) \text{ vs. } O(K \cdot ch) \text{ for distance matrix} \tag{21}$$

$$\text{Total reduction factor:} \quad \frac{1}{ch} \text{ for codebook, } \frac{1}{K} \text{ for computation} \tag{22}$$

C Additional Analysis and Experimental Results

This subsection presents additional theoretical analysis and experimental results that provide deeper insights into CSVQ's behavior.

C.1 Learning Stability and Independence Analysis

Lemma 3 (Gradient variance reduction)**.** *Let g_k be the gradient estimate for codeword c_k under CSVQ:*

$$g_k = \frac{1}{M_k} \sum_{i=1}^{ch} \sum_{h=1}^{H} \sum_{w=1}^{W} \mathbf{1}[z_{q,h,w}^{(i)} = k] \cdot \psi_{h,w}^{(i)},$$

where $M_k = \sum_{i,h,w} \mathbf{1}[z_{q,h,w}^{(i)} = k]$ and $\psi_{h,w}^{(i)}$ is the local gradient signal. Under assumptions of 1) weak inter-channel correlation: $|Cov(\psi_{h,w}^{(i)}, \psi_{h',w'}^{(j)})| \leq \delta\sigma^2$ for $i \neq j$ with $\delta \ll 1$, 2) uniform variance: $Var(\psi_{h,w}^{(i)}) = \sigma^2$, and 3) selection independence, we have:

$$Var(g_k) = \frac{\sigma^2}{M_k} + O(\delta) \approx \frac{K\sigma^2}{chHWB} = \Theta(1/ch)$$

Proof. Under weak correlation and independence assumptions, the variance decomposes as $\text{Var}(g_k) = \frac{\sigma^2}{M_k} + O(\delta)$. With uniform selection probability $1/K$, we have $\mathbb{E}[M_k] = \frac{ch \cdot H \cdot W \cdot B}{K}$, yielding the $\Theta(1/ch)$ scaling. □

Theorem 4 (Convergence acceleration)**.** *Under the gradient variance bound from Lemma 3 and standard SGD assumptions with learning rate η and T iterations, the optimization error bound for CSVQ scales as:*

$$\mathbb{E}[\mathcal{L}(\theta^{(T)}) - \mathcal{L}(\theta^*)] = O\left(\frac{\sigma^2}{ch\eta T}\right)$$

where θ^* is the optimal parameter and $\theta^{(T)}$ is the parameter after T iterations.

Proof. Follows from standard SGD convergence analysis with the improved gradient variance bound $\Theta(1/ch)$ from channel aggregation. □

Full proof is provided in Appendix B.3.

C.2 Training Stability Experiments

We evaluate stability using loss and gradient variance across 5 seeds. Protocol P1 uses a 64-parameter budget with VQ-VAE (codebook 16, channels 4) and CSVQ (codebook 64, channels 4), while P4 uses a fixed 256 codebook with 32 channels. These choices follow Section 4.2: in P1 we fix the codebook-parameter budget $M_{\text{cb}} := K \times$(codebook dimension) to control memory/search cost for fair comparison (VQ/RQ: 16×4, CSVQ: 64×1 so $M_{\text{cb}}=64$), whereas in P4 we fix (K, d) to reflect practical matched configurations used in prior work and to probe stability at a common tokenization capacity; together, they cover tight-resource (P1) and ample-capacity (P4) regimes. Figures 2 and 3 demonstrate CSVQ's superior training stability versus VQ-VAE across both protocols. VQ-VAE exhibits high variance with increasing trends, particularly under P1, with gradient variance showing divergent tendencies and large confidence intervals. Periods where VQ-VAE's confidence intervals collapse to zero indicate codebook collapse (unused entries with vanishing gradients), while CSVQ maintains consistent variance throughout training due to its shared codebook design and channel-wise independence.

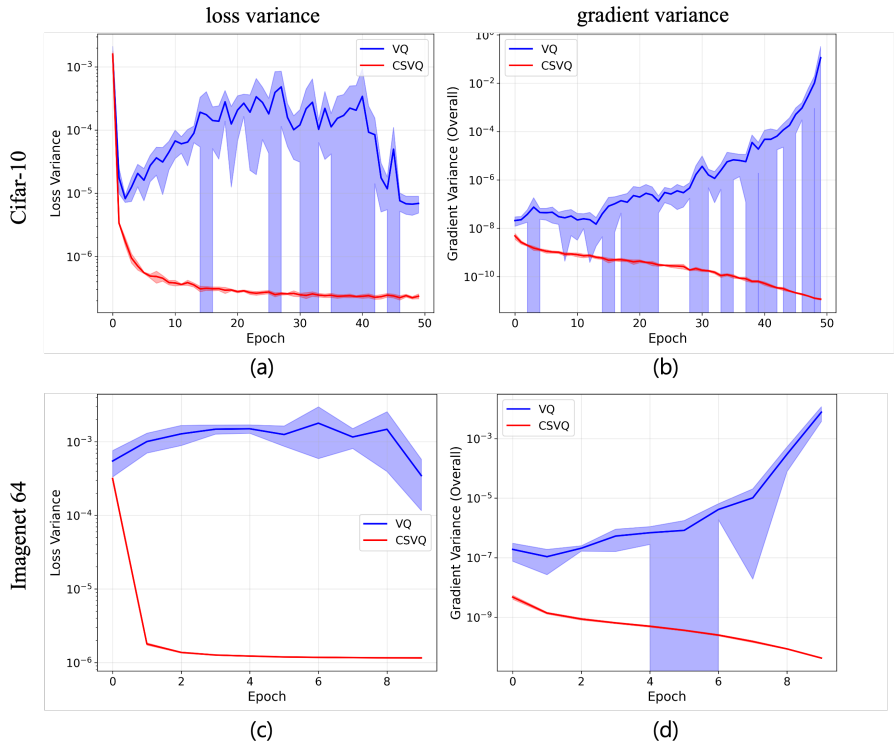

Figure 2: Training stability under P1. Datasets: CIFAR-10 (a,b) and ImageNet-64 (c,d). Panels: loss (a,c) and gradient (b,d) variances. Means are shown as lines; shaded regions denote standard deviations.

C.3 Codebook Utilization Experiments

Protocol P4 analyzes utilization across codebook sizes (8,16,32,64) with varying channels (CIFAR-10: 16, ImageNet-64: 32). We measure perplexity, dead entry ratios, and PSNR performance (see Appendix B.4 for theoretical analysis). Figures 4 and 5 show that CSVQ achieves higher perplexity, lower dead ratios, and superior PSNR performance, indicating superior capacity utilization. CSVQ demonstrates consistent and stable performance with perplexity approaching the codebook size and maintaining steady values throughout training. In contrast, VQ-VAE shows declining perplexity with codebook collapse, particularly severe on ImageNet-64. Dead entry ratios for CSVQ converge to zero across all settings, while VQ-VAE exhibits highly unstable patterns with ratios reaching up to 50% in some configurations.

Regarding PSNR performance, CSVQ shows consistent improvement throughout training and benefits from larger codebook sizes, demonstrating stable learning dynamics. However, the performance improvement diminishes as the codebook size increases, allowing us to empirically identify the optimal codebook size suitable for CIFAR-10. In contrast, VQ-VAE exhibits performance improvements with larger codebooks but suffers from unstable performance during the training process, leading to inconsistent reconstruction quality.

C.4 Ablation Studies

We analyze the contribution of CSVQ components by ablating (i) the shared codebook (replacing it with per-channel codebooks; “w/o shared codebook”), and (ii) the channel-wise normalization under the shared codebook setting (“w/o normalization”; shared codebook, no normalization). As summarized in Table 7, removing either component can yield slightly higher pixel-space reconstruction (PSNR/SSIM) on CIFAR-10 and ImageNet subset-10, consistent with a classic regularization trade-off. However, the full CSVQ consistently delivers markedly better representation quality in

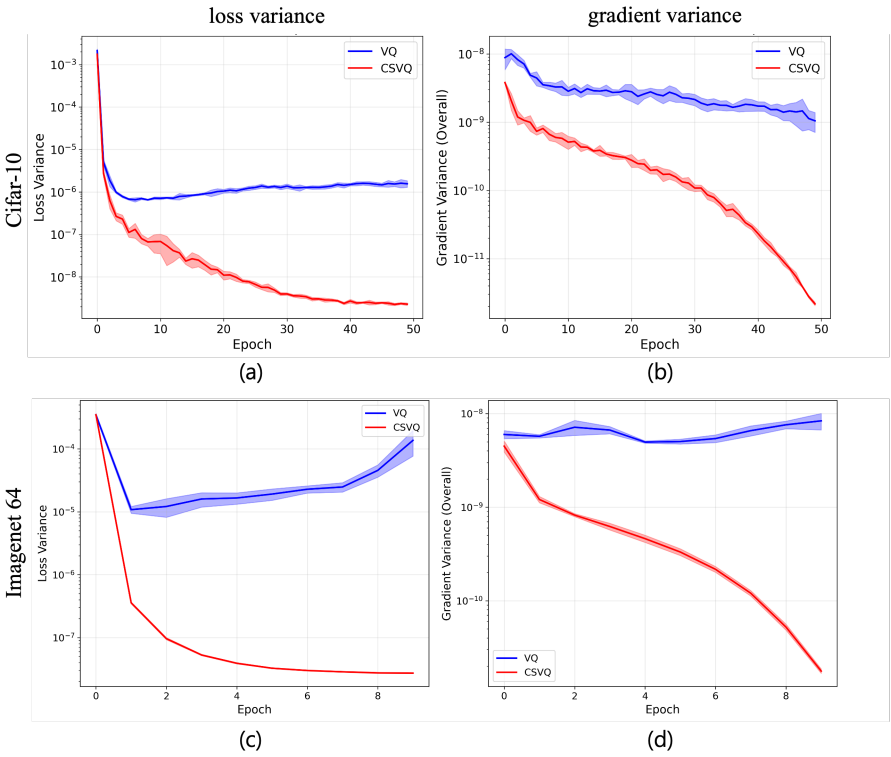

Figure 3: Learning stability under P4. Datasets: CIFAR-10 (a,b) and ImageNet-64 (c,d). Panels: loss (a,c) and gradient (b,d) variances. Means are lines; shaded areas indicate standard deviations.

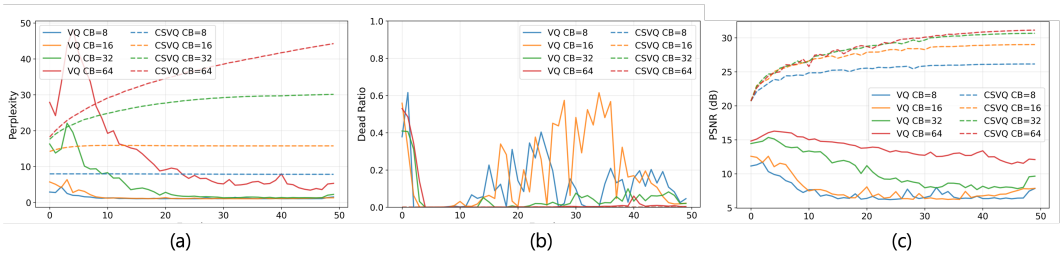

Figure 4: Codebook utilization under P4 (CIFAR-10, $ch = 16$). Panels: (a) perplexity, (b) dead-entry ratio, and (c) PSNR versus $K \in \{8, 16, 32, 64\}$.

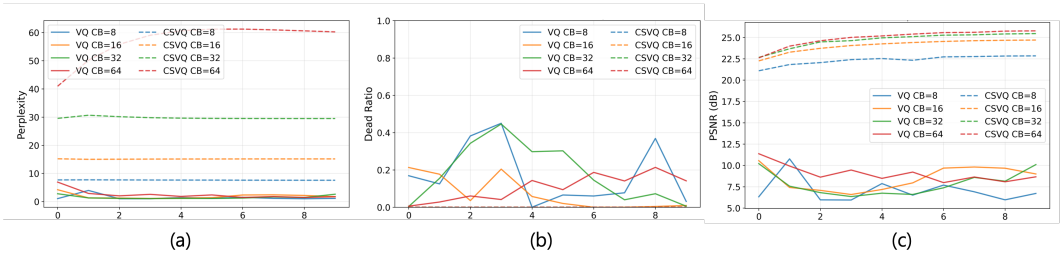

Figure 5: Codebook utilization under P4 (ImageNet-64, $ch = 32$). Panels: (a) perplexity, (b) dead-entry ratio, and (c) PSNR versus $K \in \{8, 16, 32, 64\}$.

linear probing (Top-1), indicating that the shared codebook and channel-wise normalization jointly promote stable, informative discrete representations that generalize better downstream.

Table 7: Ablation study on Protocol P1. Component contributions with codebook size 16 (CIFAR-10) / 32 (ImageNet subset-10), channels 8. Reconstruction and linear probing (Top-1) are jointly reported.

Dataset	Method	Codebook	Channels	PSNR↑	SSIM↑	LP Top-1↑
CIFAR-10	w/o shared codebook	16×8	8	**24.49**	**0.9899**	40.6
	w/o normalization	16	8	24.44	0.9897	39.96
	CSVQ (full)	16	8	23.25	0.9870	**46.3**
ImageNet subset-10	w/o shared codebook	32×8	8	**17.90**	**0.9631**	55.23
	w/o normalization	32	8	17.68	0.9600	57.96
	CSVQ (full)	32	8	17.41	0.9525	**58.32**

THE USE OF LARGE LANGUAGE MODELS

Tool & Version: Gemini (Google, 2025-09)
Research Stage: Generated visualization scripts.
Writing Stage: Language editing of author-drafted text for clarity and conciseness.
Human Oversight: All outputs reviewed/edited by the authors; authors accept full responsibility for the content.