

DIVEQ: DIFFERENTIABLE VECTOR QUANTIZATION USING THE REPARAMETERIZATION TRICK

Mohammad Hassan Vali¹, Tom Bäckström² & Arno Solin¹

¹Department of Computer Science, Aalto University, Finland

²Department of Information and Communications Engineering, Aalto University, Finland

{mohammad.vali, tom.backstrom, arno.solin}@aalto.fi

ABSTRACT

Vector quantization is common in deep models, yet its hard assignments block gradients and hinder end-to-end training. We propose DiVeQ, which treats quantization as adding an error vector that mimics the quantization distortion, keeping the forward pass hard while letting gradients flow. We also present a space-filling variant (SF-DiVeQ) that assigns to a curve constructed by the lines connecting codewords, resulting in less quantization error and full codebook usage. Both methods train end-to-end without requiring auxiliary losses or temperature schedules. On VQ-VAE compression and VQGAN generation across various data sets, they improve reconstruction and sample quality over alternative quantization approaches.

1 INTRODUCTION

Vector quantization (VQ, Gersho & Gray, 2012) is a classical compression technique for discretizing continuous data distributions into a finite set of representative vectors, resulting in a *codebook*. Each data sample is assigned to its nearest codebook vector (*codeword*), enabling compact discrete representations. Within deep learning, VQ was popularized by the vector-quantized variational autoencoder (VQ-VAE, van den Oord et al., 2017), where discretization of latent representations yields substantial improvements in compression and generative quality. Since then, VQ has become a central component in architectures for images (Yu et al., 2021), video (Yan et al., 2021), and speech (Kumar et al., 2023), becoming a fundamental module in the tool-stack for deep neural networks (DNNs).

Mapping a continuous data sample to its closest codeword is mathematically non-differentiable. Hence, when using VQ in the computational graph of a DNN, the gradients will not pass through the VQ layer in the backward pass. For instance, in a VQ-VAE, the encoder parameters will not receive any gradients from the VQ layer. This issue is known as the *gradient collapse* problem. There exists a variety of solutions to this problem, each with its own challenges and drawbacks (see Table 1).

In this paper, we revisit differentiable vector quantization through the lens of the reparameterization trick (Kingma & Welling, 2013). We propose *Differentiable Vector Quantization* (DiVeQ), a method that models quantization as the addition of a simulated quantization error vector (see Fig. 1). The direction of this vector is aligned with the nearest codeword, while its magnitude equals the input–codeword distance, thereby preserving hard assignments in the forward pass while enabling meaningful gradient flow. Unlike prior approaches such as NSVQ (Vali & Bäckström, 2022), which

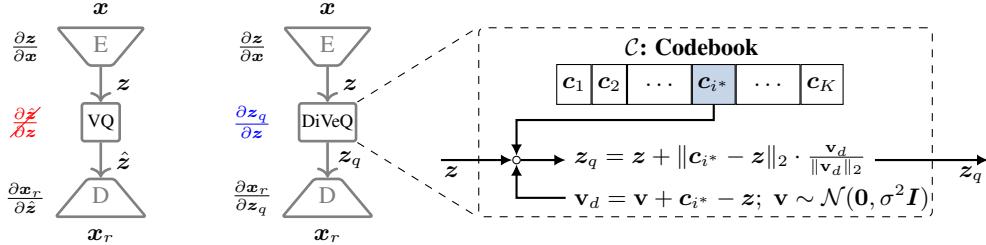


Figure 1: We replace the **non-differentiable** VQ operation ($\hat{z} = \mathbf{c}_{i^*} = \arg \min_{\mathbf{c}_j} \|\mathbf{z} - \mathbf{c}_j\|_2$) on the left with **differentiable** vector quantization (DiVeQ) on the right that lets the gradients flow.

Table 1: Summary of training properties for different VQ optimization methods in Sec. 2.

	STE	EMA	RT	ST-GS	NSVQ	DiVeQ	SF-DiVeQ
No auxiliary loss terms	✗	✗	✗	✗	✓	✓	✓
No hyperparameter tuning	✗	✗	✗	✗	✓	✓	✓
Unbiased codebook gradients	✗	N/A	✓	✗	✓	✓	✓
No train–test mismatch	✓	✓	✓	✗	✗	✓	✓
End-to-end training	✗	✗	✗	✓	✓	✓	✓
Precise nearest-codeword assignment	✓	✓	✓	✓	✗	✓	✓
Not prone to codebook misalignment	✗	✗	✗	✗	✗	✗	✓
Avoiding codebook collapse	✗	✗	✗	✗	✗	✗	✓

approximate the quantization error with limited directional fidelity, DiVeQ ensures that the differentiable surrogate remains geometrically consistent with the underlying nearest-neighbor operation.

Building on this foundation, we further introduce *Space-Filling DiVeQ* (SF-DiVeQ), which extends quantization from discrete codewords to continuous line segments connecting neighboring codewords. This construction simultaneously mitigates quantization error and alleviates codebook under-utilization without requiring replacement heuristics. By quantizing along structured manifolds within the codebook, SF-DiVeQ achieves full utilization and improves representational efficiency.

We evaluate DiVeQ and SF-DiVeQ on VQ-VAE (van den Oord et al., 2017) for image compression and VQGAN (Esser et al., 2021) for image generation across CELEBA-HQ, FFHQ, AFHQ, and LSUN Bedroom and Church. Our methods consistently improve reconstruction fidelity and maintain sample quality compared to existing quantization strategies. Importantly, DiVeQ and SF-DiVeQ act as *drop-in replacements* for standard VQ layers, requiring only minimal changes to model code.

The contributions of this paper are as follows:

- We propose DiVeQ, a differentiable vector quantization technique that enables end-to-end training with hard forward assignments, avoiding auxiliary losses and hyperparameter tuning.
- We further propose SF-DiVeQ, a space-filling variant, which quantizes along codeword connections, ensuring reduced quantization error and full codebook utilization without any auxiliary losses or codebook reinitialization. Contrary to all other methods, SF-DiVeQ avoids misalignment of latent and codebook representations.
- For VQ optimization methods prone to *codebook collapse*, we present an improved codebook replacement algorithm, achieving faster and more stable utilization than prior methods (see Sec. B.1). Also, we show that DiVeQ and SF-DiVeQ are advantageously applicable to other VQ variants like Residual VQ with superior performance to the other techniques (see Sec. C.7).

2 BACKGROUND AND RELATED WORK

Vector quantization has been extensively studied in both signal processing and modern deep learning, and a wide range of strategies have been developed to overcome its non-differentiability. Since the core challenge addressed in this paper is precisely the integration of VQ into end-to-end trainable neural architectures, we provide a structured review of existing solutions. This detailed discussion is necessary because, although many methods yield partial remedies, they differ substantially in terms of optimization objectives, gradient fidelity, codebook utilization, and computational overhead. Our goal in this section is therefore twofold: (*i*) to establish a precise baseline understanding of the mechanisms underlying each class of methods, and (*ii*) to highlight their limitations in relation to the desiderata of differentiable quantization.

Vector quantization (VQ, Gersho & Gray, 2012) clusters a continuous distribution to a limited set of codewords, by mapping a latent vector $z \in \mathbb{R}^D$ to the nearest codeword $c_{i^*} \in \mathbb{R}^D$ of a codebook $\mathcal{C} = \{c_1, \dots, c_K\}$, where $i^* = \arg \min_j \|z - c_j\|_2$. This mapping introduces a distortion $\xi_Q = z - c_{i^*}$, known as quantization error. Since the nearest-neighbor assignment is non-differentiable, VQ blocks gradients during backpropagation, leading to the well-known *gradient collapse* problem (Mentzer et al., 2023). To explain existing techniques, we consider the

case of training VQ codebook in a vector-quantized variational autoencoder (VQ-VAE, van den Oord et al., 2017). In VQ-VAE (Fig. 1), the input \mathbf{x} is fed to the encoder E to obtain the continuous latent variable \mathbf{z} , which is then discretized by VQ to $\hat{\mathbf{z}}$. The discrete latent variable $\hat{\mathbf{z}}$ is fed to the decoder D to reconstruct the input \mathbf{x}_r :

$$\mathbf{z} = E(\mathbf{x}) \longrightarrow \hat{\mathbf{z}} = \text{VQ}(\mathbf{z}) ; \hat{\mathbf{z}} = \mathbf{c}_{i^*} = \arg \min_{\mathbf{c}_j} \|\mathbf{z} - \mathbf{c}_j\|_2 \longrightarrow \mathbf{x}_r = D(\hat{\mathbf{z}}). \quad (1)$$

Since $\arg \min$ is not differentiable ($\frac{\partial \hat{\mathbf{z}}}{\partial \mathbf{z}}$ does not exist), VQ does not pass any gradients from \mathbf{x}_r to \mathbf{x} .

Straight-Through Estimator (STE, Bengio et al., 2013) deals with this issue by copying the gradients through a non-differentiable VQ function, while assuming $\frac{\partial \hat{\mathbf{z}}}{\partial \mathbf{z}} = 1$, such that it defines the final quantized latent as $\mathbf{z}_q = \mathbf{z} + sg[\hat{\mathbf{z}} - \mathbf{z}]$. The loss function that STE uses is

$$\text{Loss} = \text{MSE}(\mathbf{x}, \mathbf{x}_r) + \alpha \cdot \|sg[\mathbf{z}] - \mathbf{c}_{i^*}\|_2^2 + \beta \cdot \|\mathbf{z} - sg[\mathbf{c}_{i^*}]\|_2^2, \quad (2)$$

where $sg[\cdot]$ refers to the stop gradient operator and MSE is the mean squared error. The first term is the reconstruction loss that optimizes the encoder and decoder parameters by copying the gradients intact over VQ in backpropagation. The second term is the codebook loss that optimizes the selected codewords (*i.e.*, \mathbf{c}_{i^*}), and the third term is the commitment loss that optimizes the encoder parameters to make the encoder output \mathbf{z} close to \mathbf{c}_{i^*} .

Exponential Moving Averages (EMA, van den Oord et al., 2017) also copies the gradients intact through VQ via $\mathbf{z}_q = \mathbf{z} + sg[\hat{\mathbf{z}} - \mathbf{z}]$. However, EMA updates the codebook vectors as a function of moving averages of latent variables \mathbf{z} . EMA keeps the reconstruction and commitment losses, and skips the codebook loss in Eq. (2) to update the selected codewords as

$$\mathbf{c}_{i^*}^{(t)} := \frac{\mathbf{g}_{i^*}^{(t)}}{h_{i^*}^{(t)}} \text{ s.t. } h_{i^*}^{(t)} := \gamma \cdot h_{i^*}^{(t-1)} + (1-\gamma) \cdot n_{i^*}^{(t)} \quad \text{and} \quad \mathbf{g}_{i^*}^{(t)} := \gamma \cdot \mathbf{g}_{i^*}^{(t-1)} + (1-\gamma) \cdot \sum_{j=1}^{n_{i^*}^{(t)}} \mathbf{z}_{i^*,j}^{(t)}, \quad (3)$$

where at training iteration t , $\{\mathbf{z}_{i^*,1}, \mathbf{z}_{i^*,2}, \dots, \mathbf{z}_{i^*,n_{i^*}}\}$ is the set of n_{i^*} latent variables that are closest to the selected codeword \mathbf{c}_{i^*} , and $\gamma \in (0, 1)$ is the decay rate that controls how much past observations influence the updates.

Rotation Trick (RT, Fifty et al., 2025) passes gradients through VQ by transforming the latent \mathbf{z} to its closest codeword \mathbf{c}_{i^*} with a combination of rotation and rescaling such that it defines

$$\mathbf{z}_q = sg[\rho \mathbf{R} \mathbf{z}] \quad \text{s.t.} \quad \rho = \frac{\|\mathbf{c}_{i^*}\|_2}{\|\mathbf{z}\|_2} \quad \text{and} \quad \mathbf{R} = (\mathbf{I} - 2\mathbf{r}\mathbf{r}^\top + 2\bar{\mathbf{c}}_{i^*}\bar{\mathbf{c}}_{i^*}^\top), \quad (4)$$

where $\bar{\mathbf{z}} = \frac{\mathbf{z}}{\|\mathbf{z}\|_2}$, $\bar{\mathbf{c}}_{i^*} = \frac{\mathbf{c}_{i^*}}{\|\mathbf{c}_{i^*}\|_2}$, and $\mathbf{r} = \frac{\bar{\mathbf{z}} + \bar{\mathbf{c}}_{i^*}}{\|\bar{\mathbf{z}} + \bar{\mathbf{c}}_{i^*}\|_2}$. RT uses the same loss as in Eq. (2), while the codebook can be updated via gradients using codebook loss or via EMA.

Gumbel-Softmax (GS, Jang et al., 2016) samples differentiable variables \mathbf{y}_i from a continuous distribution that approximates a discrete K -class categorical distribution over K codewords

$$\mathbf{y}_i = \text{softmax}((\log(\pi_i) + \mathbf{g}_i)/\tau) = \frac{\exp((\log(\pi_i) + \mathbf{g}_i)/\tau)}{\sum_{j=1}^K \exp((\log(\pi_j) + \mathbf{g}_j)/\tau)}, \quad i \in \{1, 2, \dots, K\}, \quad (5)$$

where $\{\pi_1, \pi_2, \dots, \pi_K\}$ are class probabilities of the categorical distribution and $\{\mathbf{g}_1, \mathbf{g}_2, \dots, \mathbf{g}_K\}$ are i.i.d. samples drawn from Gumbel(0, 1) distribution. The scalar τ is the softmax temperature. When τ approaches 0, the Gumbel-Softmax distribution converges to a discrete categorical distribution. To train a VQ-VAE, the loss function that Gumbel-Softmax uses is

$$\text{Loss} = \text{MSE}(\mathbf{x}, \mathbf{x}_r) + \varphi \cdot D_{\text{KL}}[Q(\mathbf{z} | \mathbf{x}) \| P(\mathbf{z})], \quad (6)$$

where D_{KL} is the Kullback–Leibler divergence that pushes the encoder’s posterior distribution $Q(\mathbf{z} | \mathbf{x})$ to be close to the uniform prior distribution of $P(\mathbf{z}) = \frac{1}{K}$. The D_{KL} term encourages all K codewords to be used for quantization. As stated in Jang et al. (2016), for VQ tasks it is better to use the Straight-Through version of Gumbel-Softmax (ST-GS), in which \mathbf{y}_i is discretized using $\arg \max$ in the forward pass, and gradients are computed from the continuous \mathbf{y}_i in Eq. (5).

Noise Substitution in Vector Quantization (NSVQ, Vali & Bäckström, 2022) simulates the VQ distortion by adding a noise to the latent vector to mimic the original quantization error ξ_Q :

$$\mathbf{z}_q = \mathbf{z} + \xi_Q = \mathbf{z} + \|\mathbf{z} - \hat{\mathbf{z}}\|_2 \cdot \frac{\mathbf{v}}{\|\mathbf{v}\|_2}; \quad \hat{\mathbf{z}} = \mathbf{c}_{i^*} = \arg \min_{\mathbf{c}_j} \|\mathbf{z} - \mathbf{c}_j\|_2 \quad \text{and} \quad \mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \quad (7)$$

In this way, z_q is a differentiable function of the latent z and the selected codeword c_{i^*} . As shown in Fig. 2, as NSVQ samples the noise v from $\mathcal{N}(\mathbf{0}, \mathbf{I})$, the quantized latent z_q can be mapped to any random direction on the surface of a hypersphere with the radius of quantization error that equals $\|z - c_{i^*}\|_2$. Fig. 2 shows the probability that z_q incurs higher quantization errors than the actual error (of $\|z - c_{i^*}\|_2$) that equals $\theta_2/360^\circ$. For any radius, $\theta_2 = 240^\circ$. Hence, NSVQ incurs a higher quantization error than the true error with the probability of $240^\circ/360^\circ \approx 0.67$. Furthermore, due to high randomness in the direction of z_q , it is challenging for the codebook to converge to its optimum location. This randomness also causes a train–test mismatch, which leads to poor reconstructions, because NSVQ applies different mappings for the same z during training and testing.

Issues with the State-of-the-Art All of the methods mentioned above have their own drawbacks, which are highlighted in Table 1. To optimize the VQ codebook, they (i) change the hyperplane of optimization by adding auxiliary loss terms (Eqs. (2) and (6)) to the main objective loss (STE, EMA, RT, ST-GS), (ii) need additional hyperparameter (*i.e.*, $\alpha, \beta, \gamma, \varphi, \tau$) tuning (STE, EMA, RT, ST-GS), (iii) do not train codebooks end-to-end (STE, EMA, RT), (iv) cause biased gradients via mismatch between forward and backward passes (STE, ST-GS), (v) cause train–test mismatch as they apply different quantization for training and testing (ST-GS, NSVQ), (vi) suffer from *codebook collapse* (STE, EMA, RT, ST-GS, NSVQ), and (vii) are prone to misaligned latent and codebook representations (STE, EMA, RT, ST-GS, NSVQ) (see Fig. 4 and Sec. C.6).

Other Related Work There exist many approaches that still use STE to pass gradients through the VQ layer (Chang et al., 2022; Rombach et al., 2022; Zhu et al., 2023; Huang et al., 2023; Dong et al., 2023; Lee et al., 2022; Huh et al., 2023), each with small modifications to improve something specific. For instance, some methods use STE while resolving *codebook collapse* with different codebook replacement and reinitialization strategies (Kolesnikov et al., 2022; Łaćucki et al., 2020; Zheng & Vedaldi, 2023; Dhariwal et al., 2020; Zeghidour et al., 2021), or by using different distance metrics than Euclidean (Yu et al., 2021; Goswami et al., 2024). Others use stochastic sampling (Maddison et al., 2017; Takida et al., 2022; Chen et al., 2024) while increasing codebook utilization by additional loss terms (Zhang et al., 2023; Yu et al., 2023), or soft quantization to backpropagate gradients through the VQ layer (Gautam et al., 2023; Agustsson et al., 2017). Additionally, some recent methods use NSVQ (Gómez et al., 2023; Walsh et al., 2024; Lee et al., 2024; Wang et al., 2025; Ye et al., 2025; Zhu et al., 2025), and RT (Kim et al., 2025; Xue et al., 2025; Bae & Shin, 2025) techniques. In rather unique lines of work, Finite Scalar Quantization (FSQ, Mentzer et al., 2023), Random Projection Quantizer (RPQ, Chiu et al., 2022), Binary Spherical Quantization (BSQ, Zhao et al., 2024), and Lookup-Free Quantization (LFQ, Yu et al., 2023) constrain the codebook to follow a predefined geometric structure and do not train the codebook together with the model.

3 METHODS

We address the problem of making vector quantization differentiable while preserving hard assignments in the forward pass. Our goal is to construct a differentiable surrogate $z_q(z, \mathcal{C})$ that (i) coincides with the hard nearest-neighbor assignment in the small-variance limit, (ii) yields stable and geometrically faithful gradients for both encoder and codebook, and (iii) avoids auxiliary losses, hyperparameter tuning, or train–test mismatches. We first introduce *DiVeQ* (Sec. 3.1), which reparameterizes the quantization error to align its direction with the nearest codeword. We then extend this construction to *SF-DiVeQ* (Sec. 3.2), which quantizes along line segments between neighboring codewords to reduce error and promote codebook utilization.

3.1 DiVeQ: DIFFERENTIABLE VQ VIA DIRECTIONAL REPARAMETERIZATION

DiVeQ models quantization as adding a simulated error vector whose *magnitude* equals the input–codeword distance and whose *direction* is aligned with the nearest codeword. Similar to NSVQ

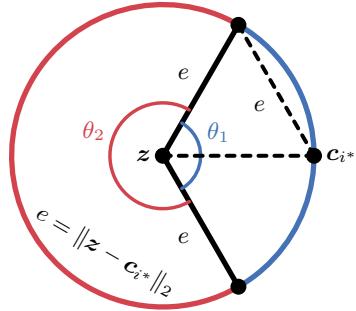


Figure 2: **Illustration of NSVQ quantization.** Input z is mapped to a random point on the circle. The mapping overshoots the true quantization error with probability $\theta_2/360^\circ \approx 0.67$, leading to a higher distortion than the nearest-codeword assignment.

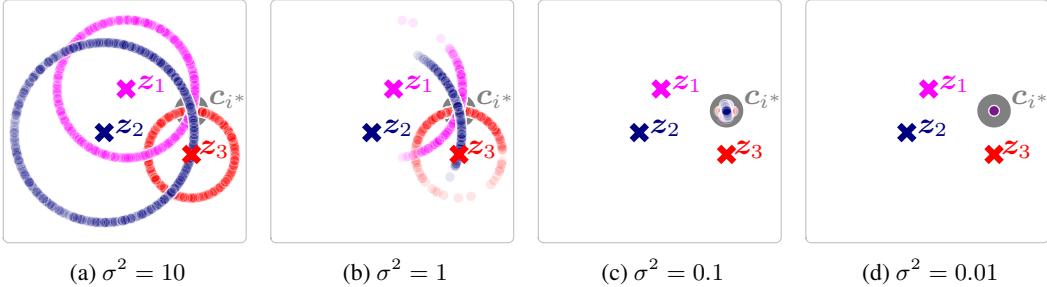


Figure 3: Impact of σ^2 in DiVeQ quantization accuracy. Each panel shows mappings of input z_i to its closest codeword c_{i*} using our proposed DiVeQ (Eq. (8)) when sampling 1000 random directional vectors v_d from $\mathcal{N}(0, \sigma^2 I)$. DiVeQ quantization accuracy increases when $\sigma^2 \rightarrow 0$ (see Sec. C.5).

(see Sec. 2), to obtain a differentiable quantized input z_q , DiVeQ approximates VQ as the addition of the quantization error ξ_Q to the input z , i.e., $z_q = z + \xi_Q$. However, in difference to NSVQ, DiVeQ models ξ_Q such that z_q points to the closest codeword c_{i*} precisely and thus, z_q results in the original quantization error of $\|z - c_{i*}\|_2$ (see Fig. 3). To this end, by getting intuition from the reparameterization trick (Kingma & Welling, 2013), we define a directional noise $v_d = v + \vec{d}$ such that $\vec{d} = c_{i*} - z$ and $v \sim \mathcal{N}(0, \sigma^2 I)$. Then,

$$z_q = z + \xi_Q = z + \|\vec{d}\|_2 \cdot sg \left[\frac{v_d}{\|v_d\|_2} \right] = z + \|c_{i*} - z\|_2 \cdot sg \left[\frac{v_d}{\|v_d\|_2} \right]; \quad c_{i*} = \arg \min_{c_j} \|z - c_j\|_2, \quad (8)$$

where $sg[\cdot]$ is the stop gradient operator. Here, by changing the variance σ^2 from ∞ to zero (Fig. 3a to Fig. 3d), the quantization accuracy will increase, as z_q starts from pointing to a hypersphere (as in NSVQ) and approaches pointing exactly to the selected codeword c_{i*} . The limits give:

$$\begin{aligned} \lim_{\sigma^2 \rightarrow \infty} v_d &= \lim_{\sigma^2 \rightarrow \infty} [\mathcal{N}(0, \sigma^2 I) + \vec{d}] = \mathcal{N}(c_{i*} - z, \infty) \Rightarrow z_q = z + \|c_{i*} - z\|_2 \cdot sg[s], \\ \lim_{\sigma^2 \rightarrow 0} v_d &= \lim_{\sigma^2 \rightarrow 0} [\mathcal{N}(0, \sigma^2 I) + \vec{d}] \approx \vec{d} = c_{i*} - z \Rightarrow z_q = z + \|c_{i*} - z\|_2 \cdot sg \left[\frac{c_{i*} - z}{\|c_{i*} - z\|_2} \right] = c_{i*}, \end{aligned} \quad (9)$$

where s is a random variable almost uniformly distributed on the surface of the D -dim unit hypersphere of $\mathbb{S} = \{\forall s \in \mathbb{R}^D : \|s\|_2 = 1\}$. When using Eq. (8) during training, z_q is a differentiable function of z and c_{i*} , and thus it can be used in end-to-end training of VQ together with other trainable modules in a neural network. Hence, the gradients can be calculated as

$$\frac{\partial z_q}{\partial z} = \mathbf{1} + a \cdot \frac{z - c_{i*}}{\|c_{i*} - z\|_2} \quad \text{and} \quad \frac{\partial z_q}{\partial c_{i*}} = a \cdot \frac{c_{i*} - z}{\|c_{i*} - z\|_2} \quad \text{s.t.} \quad a = sg \left[\frac{c_{i*} - z}{\|c_{i*} - z\|_2} \right]. \quad (10)$$

3.2 SF-DiVeQ: SPACE-FILLING DIFFERENTIABLE VECTOR QUANTIZATION

Space-Filling Vector Quantization (SFVQ, Vali & Bäckström, 2023) is a modification of VQ that maps an input to a piece-wise continuous curve (see Fig. 1 in Vali & Bäckström, 2023). Contrary to VQ that quantizes the input exclusively on the codewords, SFVQ is a curve that gets turned and twisted inside a D -dim distribution and quantizes the input on a continuous curve whose corner points are the codewords of the SFVQ base codebook \mathcal{C} . SFVQ adopts a dithering trick to generate the curve. For each training iteration by having an input $z \in \mathbb{R}^D$ and a base codebook \mathcal{C} with K codewords, SFVQ generates a dithered codebook $\mathcal{C}^d = \{c_1^d, \dots, c_{K-1}^d\}$ with $K-1$ codewords by sampling at random places on the line connecting two subsequent codewords of c_j and c_{j+1} . The input z is then quantized to the closest codeword from the dithered codebook \mathcal{C}^d as

$$\hat{z} = \arg \min_{c_j^d} \|z - c_j^d\|_2 = c_{i*}^d = (1 - \lambda_{i*})c_{i*} + \lambda_{i*}c_{i*+1}, \quad j, i* \in \{1, \dots, K-1\}, \quad (11)$$

where c_{i*}^d is the closest dithered codeword that is generated by the interpolation of two subsequent codewords of c_{i*} and c_{i*+1} from the base codebook \mathcal{C} . λ_j is the interpolation factor for the codewords c_j and c_{j+1} that is sampled from uniform distribution of $U(0, 1)$. Generating new dithered

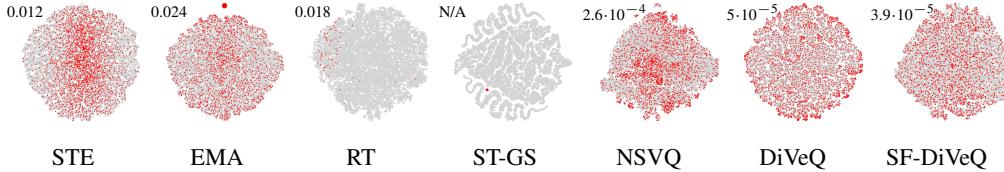


Figure 4: **Codebook misalignment:** t-SNE plots of the learned codebook \mathcal{C}_z (*red crosses*) and latent \mathcal{P}_z (*gray points*) representations for different VQ methods in VQ-VAE compression. The figure shows the misalignment between \mathcal{C}_z and \mathcal{P}_z (discussed in Sec. 4) for different methods. The plots refer to the cases highlighted in Fig. 22. The numbers report distortion per bit \downarrow (see Sec. C.6).

codebooks by random interpolations (using λ_j) on the line connecting c_j and c_{j+1} over and over for different training batches will establish a continuous topology between subsequent codewords of \mathcal{C} and locates SFVQ codewords such that the lines connecting subsequent codewords to be inside the distribution space. The reason is that the points on the SFVQ curve are representatives of dithered codewords c_j^d that should be valid quantization points for the input data.

Proposed SF-DiVeQ When training VQ, if a codeword is not selected for quantization, it will not receive any gradients and, as a result, it will not get updated. This problem is known as *codebook collapse* (Mentzer et al., 2023), in which a subset of codebook vectors remains inactive and will not get updated during training. To resolve this issue, NSVQ proposed a codebook replacement procedure such that after a specific number of training batches, inactive codewords will be replaced by a perturbation of the active ones. To maximize the codebook vectors’ usage during training, it is better and safer always to apply codebook replacement regardless of the VQ optimization technique (*e.g.*, STE, EMA, RT, ST-GS, NSVQ, or DiVeQ). According to Huh et al. (2023), codebook replacement techniques work well and do not degrade the model performance. That is why we adopt codebook replacement in this paper. However, codebook replacement is a heuristic method that adds to the complications of VQ training. By leveraging intuition from SFVQ, we propose SF-DiVeQ, a differentiable VQ technique that eliminates the need for codebook replacement. SF-DiVeQ quantizes the input z to a random location on the line connecting two subsequent codewords as

$$z_q = z + \|c_{i^*} - z\|_2 \cdot sg \left[\frac{(1 - \lambda_{i^*})\mathbf{v}_{d_{i^*}}}{\|\mathbf{v}_{d_{i^*}}\|_2} \right] + \|c_{i^*+1} - z\|_2 \cdot sg \left[\frac{\lambda_{i^*} \mathbf{v}_{d_{i^*+1}}}{\|\mathbf{v}_{d_{i^*+1}}\|_2} \right]; i^* \in \{1, \dots, K-1\}, \quad (12)$$

where c_{i^*} and c_{i^*+1} are the two codewords that their interpolation is the closest quantization point to the input z , and $\mathbf{v}_{d_{i^*}} = \mathbf{v} + (c_{i^*} - \mathbf{v})$ and $\mathbf{v}_{d_{i^*+1}} = \mathbf{v} + (c_{i^*+1} - \mathbf{v})$ such that $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$. λ_{i^*} is the interpolation factor that is sampled from the uniform distribution of $U(0, 1)$ for each training batch. As discussed earlier, since during training SF-DiVeQ quantizes z on the lines connecting subsequent codewords, these lines will be pulled inside the distribution space as they should be valid quantization points. As a result of this property, SF-DiVeQ prevents misalignment of codebook and latent representations (see Fig. 4), and it does not need any heuristic codebook replacement during training. Furthermore, as SF-DiVeQ quantizes z on a curve (rather than exclusively on the codewords), it has many more degrees of freedom for quantization than ordinary VQ methods. Therefore, it potentially results in smaller quantization errors than the other methods. Apart from DiVeQ and SF-DiVeQ, we also propose two variants of them, which are discussed in Sec. B.2.

4 EXPERIMENTS

We compare the performance of our proposed DiVeQ and SF-DiVeQ with other approaches of STE, EMA, RT, ST-GS, and NSVQ in two different applications of image compression and image generation. For the compression task, we use the VQ-VAE of van den Oord et al. (2017) with minor modifications to have a basic compression model that can clearly reflect the performance difference between various VQ methods, and for image generation, we use VQGAN (Esser et al., 2021). Experimental setup and relevant results are provided in the following. For more details on the implementation, model architectures, and hyperparameters, see Sec. A.

Loss Functions In VQ-VAE compression, to enhance the quality of reconstructions for high resolution 256×256 images, in addition to the MSE reconstruction loss in Eqs. (2) and (6), we add LPIPS

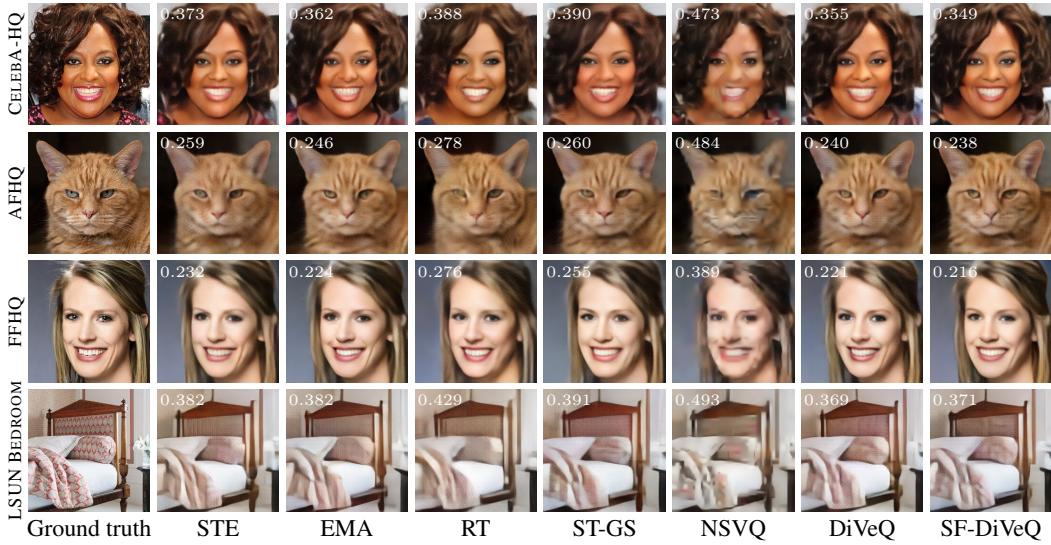


Figure 5: **DiVeQ and SF-DiVeQ improve image reconstruction.** Qualitative comparison of reconstructed images in VQ-VAE compression task for different VQ optimization methods with VQ bitrate of 11 (or codebook size of $2^{11} = 2048$). We report LPIPS \downarrow values in the left-hand corners.

(VGG-16) as the perceptual loss with a weighting coefficient of one. For VQGAN, we use the same loss functions as in [Esser et al. \(2021\)](#) for the generator, discriminator, and transformer models.

Evaluation Metrics During inference, the trained models with different VQ optimization techniques are used to reconstruct the test set images for VQ-VAE compression and to generate new images for VQGAN. For inference, we find the closest codeword with hard VQ using $\arg \min$. Structural Similarity Index Measure (SSIM), Peak Signal to Noise Ratio (PSNR) (both from *scikit-image* library), and Learned Perceptual Image Patch Similarity (LPIPS, [Zhang et al., 2018](#)) are used to evaluate the quality of the reconstructions for image compression, and Fréchet Inception Distance (FID) score ([Parmar et al., 2022](#)) is used to assess the quality of VQGAN generations.

Data Sets In both image compression and generation tasks, we do the experiments over AFHQ ([Choi et al., 2020](#)), CELEBA-HQ ([Karras et al., 2018](#)), FFHQ ([Karras et al., 2019](#)), LSUN Bedroom, and LSUN Church ([Yu et al., 2016](#)) data sets with resolution of 256×256 . The datasets contain 15803, 30k, 70k, 70k, 70k images, respectively. For image compression, all data sets are divided into 80/20% train-test splits. For image generation, we use the train sets for training except for AFHQ and CELEBA-HQ, in which we use the full data sets, as we need more data for the task. To compute the FID score, we generate the same number of images used for training.

Codebook Replacement and Initialization In the NSVQ paper ([Vali & Bäckström, 2022](#)), the NSVQ method is compared with STE and EMA, such that the codebook replacement is only used for NSVQ. To have a fair comparison of different VQ training methods, our proposed codebook replacement ([Sec. B.1](#)) is used for all methods (except SF-DiVeQ, which does not require it). Moreover, in all experiments, codebook initialization is the same for all methods except for SF-DiVeQ, in which the models are trained without VQ for two epochs, then VQ starts, such that the codebook is initialized by the mean of latent vectors of the last 10–50 training batches.

VQ-VAE Compression Results After training VQ-VAE with different VQ training methods, the trained codebook is used to reconstruct the test set images. [Fig. 5](#) shows ground truth images from different data sets and their corresponding reconstructions using learned codebooks from different VQ techniques. Apart from qualitative comparison, all test set images (from different data sets) are reconstructed, and then the SSIM, PSNR, and LPIPS metrics are computed for all of these reconstructions. [Fig. 6](#) shows the quantitative comparison of all VQ methods over different bitrates for the AFHQ data set. The reported value for each metric is the average of that metric over all test set reconstructions and over three different individual runs. In the figure, our DiVeQ and SF-DiVeQ outperform other VQ optimization methods for different bitrates and for all three metrics. Our proposed methods yield higher SSIM and PSNR values, and lower LPIPS with a big margin

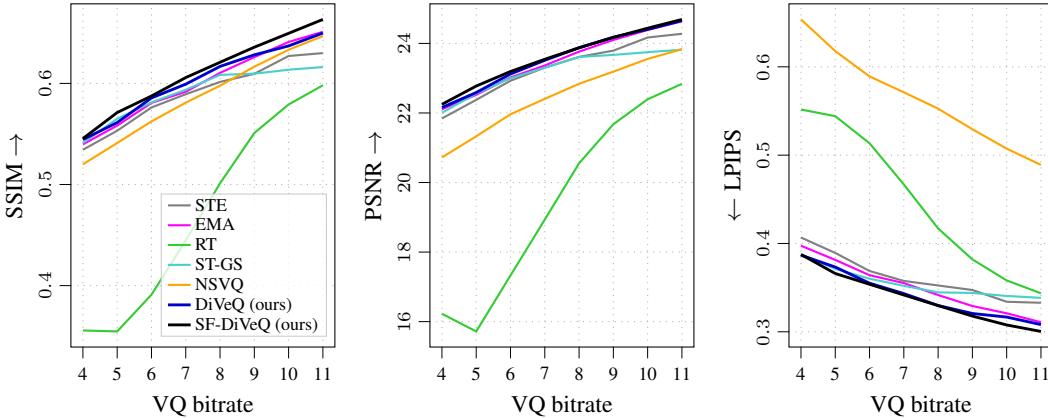


Figure 6: Consistent improvement in image reconstruction. Quantitative comparison of reconstructed images from AFHQ test set for different VQ optimization methods over different bitrates. Each curve is the average of the metric over all test set reconstructions and over three different individual runs. We include results for FFHQ, CELEBA-HQ, and LSUN in Figs. 12 to 15 in Sec. C.1.

to RT and NSVQ approaches, and with a smaller margin to STE, EMA, and ST-GS. In addition, the performance gap of our methods to STE and ST-GS grows with the increase in VQ bitrate, while this gap decreases compared to the RT method. The last takeaway from the figure is that the RT performs poorly for low VQ bitrates (or small codebook sizes). Sec. C.1 provides similar quantitative comparisons for other data sets.

VQGAN Generation Results After training the VQGAN’s generator and transformer models for each data set, we compute the FID between the original train set and generated samples. Table 2 presents the obtained FID values for different VQ optimization methods for CELEBA-HQ data set. Table 6 in Sec. C.2 provides similar FID comparisons for other data sets. We trained the generator (*i.e.*, VQ-VAE) with small batch size and learning rate to ensure convergence for all compared methods, while this is less likely to highlight the robustness of our approach. The evaluations reveal that RT, ST-GS, and NSVQ reduce FID with the increase in bitrate, and (similar to STE and EMA) our methods maintain the generation quality over different bitrates such that they achieve lower FIDs than the other methods for low bitrates (*i.e.*, 8 and 9 bit) where the generation task becomes more challenging. Furthermore, by pairing up the FIDs for RT method with the results in VQ-VAE compression task, we can claim that RT performs poor codebook learning for low bitrates in both compression and generation tasks. Note that the FID metric reflects distributional similarity, not necessarily the reconstruction quality. Hence, the FID value should not necessarily improve with the increase in codebook size K , because (*i*) the transformer might struggle with large K where the entropy of token indices is high, and (*ii*) sampling is inherently a random process that might end up in different FID values when sampling several times from a fixed pretrained transformer. Fig. 7 shows qualitative comparison of VQGAN generations for different VQ methods (see Sec. C.9 for more qualitative comparisons).

Misalignment of Codebook and Latent Representations When training a neural network (*e.g.*, VQ-VAE), the latent representation \mathcal{P}_z of the network layers is constantly shifting. For high learning rates and large batch sizes, the update shift in \mathcal{P}_z can be large, causing a misalignment between \mathcal{P}_z and codebook representation \mathcal{C}_z . Misalignment means that \mathcal{C}_z does not fit well with \mathcal{P}_z , and this phenomenon is also called *internal codebook covariate shift* (Huh et al., 2023).

Codebook replacement mitigates this issue, but it does not avoid it. Fig. 4 shows the learned \mathcal{P}_z and \mathcal{C}_z representations for the VQ-VAEs trained in our image compression task using different VQ methods. According to the figure, different types of misalignment happen for all methods (except

Table 2: DiVeQ and SF-DiVeQ robustify training for challenging low bitrates. FID \downarrow scores for different VQ optimization methods over different bitrates for the CELEBA-HQ data set. See Table 6 in the Sec. C.2 for more results.

	VQ bitrate			
	8	9	10	12
STE	6.64	5.57	5.28	6.69
EMA	6.86	6.30	6.32	6.24
RT	9.32	7.55	6.40	5.44
ST-GS	8.47	6.81	5.48	5.47
NSVQ	81.5	70.4	59.2	48.9
DiVeQ (ours)	5.90	6.69	6.32	7.69
SF-DiVeQ (ours)	6.24	5.21	5.57	6.00

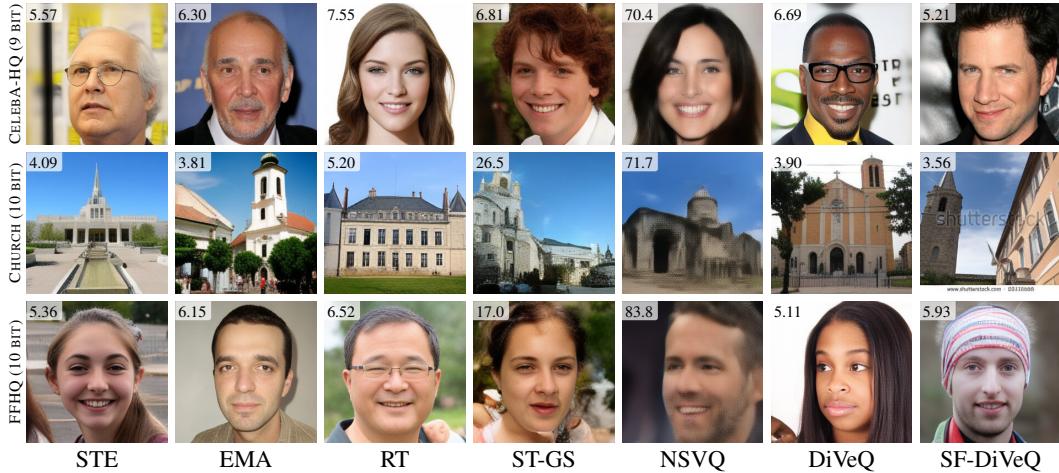


Figure 7: **Generation task.** Qualitative comparison of random generated images in the VQGAN generation task for different VQ optimization methods. We report $\text{FID} \downarrow$ values in the left-hand corners. More generations are provided in Figs. 26 to 30 in Sec. C.9.

for SF-DiVeQ), even if they use codebook replacement. For STE, RT, and NSVQ, the codewords are located inside \mathcal{P}_z , but they are not homogeneously scattered. In ST-GS, \mathcal{C}_z is completely out of \mathcal{P}_z , whereas in EMA, a large portion of codewords are outside \mathcal{P}_z . A much milder misalignment happens for our proposed DiVeQ, as a small portion of codewords are scattered densely within \mathcal{P}_z , particularly in the corners. However, SF-DiVeQ scatters the codewords homogeneously within \mathcal{P}_z . In our experiments, we do not notice any sign of misalignment for SF-DiVeQ. We hypothesize the reason is the training strategy of SF-DiVeQ, which pulls codewords inside \mathcal{P}_z without requiring any heuristic codebook replacement.

According to Shannon’s rate-distortion theory (Shannon, 1959), the rate-distortion function $D(R)$ is a continuous, strictly decreasing, convex function. Therefore, in a lossy compression task, if the VQ bitrate increases, then the distortion D must decrease, and as a result, the quantitative metrics improve (assuming the metrics fully reflect the amount of distortion). After plotting the quantitative results of trained VQ-VAEs with different learning rates and batch sizes (see the ablation studies on batch size and learning rate in Secs. C.3 and C.4), we spot that the rate-distortion theory does not hold in some experiments, such that the increase in VQ bitrate does not result in improvement of quantitative metrics. Hence, we hypothesize that the reason could be the misalignment of \mathcal{P}_z and \mathcal{C}_z . In Fig. 22, we highlighted those cases that are potentially prone to misalignment with red circles. t-SNE plots in Fig. 4 are for these cases, which confirms our hypothesis that misalignment is the main culprit. In this paper, a misalignment is considered the case in which the rate-distortion theory does not hold, even if all of the codewords are inside the latent representation \mathcal{P}_z (like STE in Fig. 4).

5 DISCUSSION AND CONCLUSION

In this paper, we introduced DiVeQ and SF-DiVeQ, two new differentiable vector quantization methods that can be trained end-to-end in a neural network. We provided a comprehensive overview of the limitations of existing approaches (Sec. 2) that motivated our approach. In Sec. 3.1, we showed how DiVeQ uses a directional reparameterization of the quantization error to preserve hard assignments in forward pass while allowing gradients to flow. We further extended this idea to SF-DiVeQ (Sec. 3.2), which quantizes along codeword connections (*i.e.*, space-filling) to reduce error and ensure full codebook utilization. In Sec. 4, we demonstrated that both methods outperform existing quantization strategies in VQ-VAE and VQGAN models across several benchmarks. Also, they are advantageously applicable to other VQ variants like Residual VQ (experiments in Sec. C.7).

Importantly, DiVeQ and SF-DiVeQ act as **drop-in replacements** for standard VQ layers and require no auxiliary losses, temperature schedules, hyperparameter tuning, or special heuristics. We advocate their use as choices for differentiable quantization in deep generative models.

REFERENCES

- Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc Van Gool. Soft-to-hard vector quantization for end-to-end learning compressible representations. In *Advances in Neural Information Processing Systems 31 (NeurIPS)*, pp. 1141–1151. Curran Associates, Inc., 2017.
- Jungwoo Bae and Jitae Shin. Robust training framework via multi-stage feature rectification. In *2025 International Technical Conference on Circuits/Systems, Computers, and Communications (ITC-CSCC)*, pp. 1–5. IEEE, 2025.
- Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.
- 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 (CVPR)*, pp. 11315–11325, 2022.
- Hang Chen, Sankepally Sainath Reddy, Ziwei Chen, and Dianbo Liu. Balance of number of embedding and their dimensions in vector quantization. *arXiv preprint arXiv:2407.04939*, 2024.
- Yongjian Chen, Tao Guan, and Cheng Wang. Approximate nearest neighbor search by residual vector quantization. *Sensors*, 10(12):11259–11273, 2010.
- Chung-Cheng Chiu, James Qin, Yu Zhang, Jiahui Yu, and Yonghui Wu. Self-supervised learning with random-projection quantizer for speech recognition. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 3915–3924. PMLR, 2022.
- Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. StarGAN v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8188–8197, 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.
- Xiaoyi Dong, Jianmin Bao, Ting Zhang, Dongdong Chen, Weiming Zhang, Lu Yuan, Dong Chen, Fang Wen, Nenghai Yu, and Baining Guo. PeCo: Perceptual codebook for bert pre-training of vision transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 552–560, 2023.
- Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12873–12883, 2021.
- Christopher Fifty, Ronald Guenther Junkins, Dennis Duan, Aniketh Iyengar, Jerry Weihong Liu, Ehsan Amid, Sebastian Thrun, and Christopher Re. Restructuring vector quantization with the rotation trick. In *International Conference on Learning Representations (ICLR)*, 2025.
- Tanmay Gautam, Reid Pryzant, Ziyi Yang, Chenguang Zhu, and Somayeh Sojoudi. Soft convex quantization: Revisiting vector quantization with convex optimization. *arXiv preprint arXiv:2310.03004*, 2023.
- Allen Gersho and Robert M Gray. *Vector Quantization and Signal Compression*, volume 159. Springer Science & Business Media, 2012.
- Esteban Gómez, Mohammad Hassan Vali, and Tom Bäckström. Low-complexity real-time neural network for blind bandwidth extension of wideband speech. In *Proceedings of the European Signal Processing Conference (EUSIPCO)*, pp. 31–35. IEEE, 2023.
- Nabarun Goswami, Yusuke Mukuta, and Tatsuya Harada. HyperVQ: MLR-based vector quantization in hyperbolic space. *arXiv preprint arXiv:2403.13015*, 2024.

-
- Mengqi Huang, Zhendong Mao, Zhuowei Chen, and Yongdong Zhang. Towards accurate image coding: Improved autoregressive image generation with dynamic vector quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22596–22605, 2023.
- Minyoung Huh, Brian Cheung, Pulkit Agrawal, and Phillip Isola. Straightening out the straight-through estimator: Overcoming optimization challenges in vector quantized networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 14096–14113. PMLR, 2023.
- Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with Gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.
- Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations (ICLR)*, 2018.
- Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4401–4410, 2019.
- Nam-Gyu Kim, Deok-Hyeon Cho, Seung-Bin Kim, and Seong-Whan Lee. Spotlight-TTS: Spot-lighting the style via voiced-aware style extraction and style direction adjustment for expressive text-to-speech. *arXiv preprint arXiv:2505.20868*, 2025.
- Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- 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 35 (NeurIPS)*, pp. 26295–26308. Curran Associates, Inc., 2022.
- Rithesh Kumar, Prem Seetharaman, Alejandro Luebs, Ishaan Kumar, and Kundan Kumar. High-fidelity audio compression with improved RVQGAN. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*, pp. 27980–27993. Curran Associates, Inc., 2023.
- Adrian Łafćucki, Jan Chorowski, Guillaume Sanchez, Ricard Marxer, Nanxin Chen, Hans JGA Dolfig, Sameer Khurana, Tanel Alumäe, and Antoine Laurent. Robust training of vector quantized bottleneck models. In *International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7. IEEE, 2020.
- Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11523–11532, 2022.
- Jun-Woo Lee, Min Je Lee, Dong-Joo Min, and Yongchae Cho. Reviving legacy seismic data via machine learning technique part 1: Expanding 3D seismic survey coverage with gated convolution GAN. *IEEE Transactions on Geoscience and Remote Sensing*, 2024.
- C Maddison, A Mnih, and Y Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations (ICLR)*, 2017.
- Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: VQ-VAE made simple. *arXiv preprint arXiv:2309.15505*, 2023.
- Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. On aliased resizing and surprising subtleties in GAN evaluation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, 2022.

-
- Claude E. Shannon. Coding theorems for a discrete source with a fidelity criterion. *IRE National Convention Record*, 4:142–163, 1959.
- 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.
- Mohammad Hassan Vali and Tom Bäckström. NSVQ: Noise substitution in vector quantization for machine learning. *IEEE Access*, 10:13598–13610, 2022.
- Mohammad Hassan Vali and Tom Bäckström. Interpretable latent space using space-filling curves for phonetic analysis in voice conversion. In *Proceedings of Interspeech*, 2023.
- Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, pp. 6309–6318. Curran Associates, Inc., 2017.
- Harry Walsh, Abolfazl Ravanshad, Mariam Rahmani, and Richard Bowden. A data-driven representation for sign language production. In *18th International Conference on Automatic Face and Gesture Recognition*, pp. 1–10. IEEE, 2024.
- Haishan Wang, Mohammad Hassan Vali, and Arno Solin. Compressing 3D Gaussian splatting by noise-substituted vector quantization. In *Scandinavian Conference on Image Analysis*, pp. 338–352. Springer, 2025.
- Rongkun Xue, Yazhe Niu, Shuai Hu, Zixin Yin, Yongqiang Yao, and Jing Yang. HH-Codec: High compression high-fidelity discrete neural codec for spoken language modeling. *arXiv preprint arXiv:2507.18897*, 2025.
- Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. VideoGPT: Video generation using VQ-VAE and transformers. *arXiv preprint arXiv:2104.10157*, 2021.
- Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Sejune Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Yuchen Lin, Lars Liden, Kimin Lee, Jianfeng Gao, Luke Zettlemoyer, Dieter Fox, and Minjoon Seo. Latent action pretraining from videos. In *International Conference on Learning Representations (ICLR)*, 2025.
- Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2016.
- 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. *arXiv preprint arXiv:2110.04627*, 2021.
- Lijun Yu, José Lezama, Nitesh B Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Vighnesh Birodkar, Agrim Gupta, Xiuye Gu, Alexander Hauptmann, Boqing Gong, Ming-Hsuan Yang, Irfan Essa, David Ross, and Lu Jiang. Language model beats diffusion-tokenizer is key to visual generation. *arXiv preprint arXiv:2310.05737*, 2023.
- Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. Sound-Stream: an end-to-end neural audio codec. *arXiv preprint arXiv:2107.03312*, 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.
- Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- Yue Zhao, Yuanjun Xiong, and Philipp Krähenbühl. Image and video tokenization with binary spherical quantization. *arXiv preprint arXiv:2406.07548*, 2024.

-
- Chuanxia Zheng and Andrea Vedaldi. Online clustered codebook. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22798–22807, 2023.
- Lin Zhu, Weifeng Zhu, Shuowen Zhang, Shuguang Cui, and Liang Liu. Scalable transceiver design for multi-user communication in FDD massive MIMO systems via deep learning. *arXiv preprint arXiv:2504.11162*, 2025.
- Zixin Zhu, Xuelu Feng, Dongdong Chen, Jianmin Bao, Le Wang, Yinpeng Chen, Lu Yuan, and Gang Hua. Designing a better asymmetric VQGAN for stablediffusion. *arXiv preprint arXiv:2306.04632*, 2023.

APPENDICES

Here, we provide an overview of the contents covered in the Appendices. In Sec. A, we provide a comprehensive and detailed explanation of the implementations for training the models in both VQ-VAE compression and VQGAN generation tasks. In Sec. B, we present other proposals of the paper including (i) new proposed codebook replacement (Sec. B.1), and (ii) new variants of DiVeQ and SF-DiVeQ techniques (Sec. B.2). Sec. C provides additional qualitative and quantitative evaluations of all VQ optimization methods to substantiate the superior performance of our proposed DiVeQ and SF-DiVeQ techniques in the established VQ-VAE compression and VQGAN generations tasks.

LIST OF APPENDICES

A Implementation details	14
A.1 VQ-VAE compression	14
A.2 VQGAN generation	15
A.3 Gumbel-Softmax implementation	17
A.4 Rotation Trick implementation	17
A.5 Suggestions for training the codebook using SF-DiVeQ	17
B Other proposals	18
B.1 New proposed codebook replacement	18
B.2 Proposed variants of DiVeQ and SF-DiVeQ	20
C Additional results	21
C.1 Quantitative results on other data sets for VQ-VAE compression	21
C.2 Quantitative results on other data sets for VQGAN generation	23
C.3 Ablation on batch size	23
C.4 Ablation on learning rate	25
C.5 Ablation on variance σ^2 of DiVeQ and SF-DiVeQ	26
C.6 Misalignment of codebook and latent representations	27
C.7 DiVeQ and SF-DiVeQ in Residual Vector Quantization	28
C.8 Training logs	29
C.9 More samples generated by VQGAN	31

A IMPLEMENTATION DETAILS

In this paper, we establish the VQ-VAE compression and VQGAN generation tasks to compare the performance of different VQ optimization methods within these two frameworks. This section provides the implementation details of these two tasks. There are some hyperparameters that are shared for both of these two tasks, which are mentioned in the following.

Mutual hyperparameters for VQ-VAE compression and VQGAN generation In both VQ-VAE compression and VQGAN generation tasks, we set the loss coefficients in Eqs. (2) and (6) as $\alpha = \varphi = 1$, and $\beta = 0.25$. The EMA decay factor is fixed at $\gamma = 0.99$ for all experiments in the paper. The ST-GS temperature τ is exponentially annealed from $\tau_{\text{start}} = 1$ to $\tau_{\text{min}} = 0.1$ over the training epochs (for more details see Sec. A.3).

A.1 VQ-VAE COMPRESSION

Hyperparameters In VQ-VAE compression task (for Fig. 6 in Sec. 4 and Figs. 12 to 15 in Sec. C.1), we train all VQ methods for 100 epochs with the batch size of 32 using Adam optimizer with the initial learning rate of $lr = 5.5 \cdot 10^{-4}$ that is halved after 40 and 70 epochs. All these experiments are done over eight different VQ bitrates $B = \{4, 5, 6, 7, 8, 9, 10, 11\}$. For example, for $\bar{B} = 8$ the number of codewords equals $K = 2^8 = 256$. To demonstrate that our proposed DiVeQ and SF-DiVeQ techniques performs consistently superior to the other VQ optimization methods in image compression task, we do ablation studies over different batch sizes and learning rates which are presented in Sec. C.3 and Sec. C.4, respectively. The variance σ^2 of directional noise \mathbf{v}_d for DiVeQ

and SF-DiVeQ is set to $\sigma^2 = 10^{-3}$ for VQ-VAE compression (see the ablation study on σ^2 in Sec. C.5).

Our proposed codebook replacement (Sec. B.1) is actively applied during training for all VQ optimization methods (except SF-DiVeQ which does not require it). The replacement is done in two different phases of

$$\begin{cases} iter \leq 2000 & ; \text{ replacement after each 100 training iterations} \\ 2000 < iter \leq N_{iter} - 1000 & ; \text{ replacement after each 500 training iterations} \end{cases}$$

, where $iter$ is the training iteration number, and N_{iter} is the total number of training iterations. The discarding threshold for the replacement equals 0.01 which means that the codebook replacement discards the codewords which are used less than 1% during the period that the replacement is done.

Model architecture For the image compression task, we aim to use the original VQ-VAE proposed in van den Oord et al. (2017). The VQ-VAE implementation is provided in DeepMind’s GitHub¹, and its PyTorch version in zalandoresearch GitHub repository². These implementations are meant for images of size 32×32 . Hence, to make the VQ-VAE suitable for compressing images of size 256×256 , we make some minor modifications to these implementations, and add the LPIPS (VGG-16) as the perceptual loss to enhance the quality of reconstructions.

Table 3 shows the architecture of the VQ-VAE used in our image compression experiments. The encoder consists of four downsampling blocks, and a stack of six residual blocks, and a pre-VQ Conv2D layer. Each downsampling block is a strided 2D convolutional layer with kernel size = 4 and stride = 2 followed by a ReLU activation function. Each residual block is comprised of ReLU, 3×3 Conv2D (stride = 1), ReLU, and 1×1 Conv2D (stride = 1) in this order. Pre-VQ Conv2D layer is meant to map the input channels to the VQ embedding dimension. The decoder has a symmetric architecture to the encoder but in a reverse order, with a difference that it uses transposed 2D convolutions in the upsampling blocks.

Table 3: Architecture of the VQ-VAE model used in our image compression task.

Encoder	Decoder
$x \in \mathbb{R}^{H \times W \times C}$	$z_q \in \mathbb{R}^{h \times w \times C''}$
$4 \times \{\text{Downsampling Block}\} \rightarrow \mathbb{R}^{h \times w \times C'}$	$\text{Conv2D} \rightarrow \mathbb{R}^{h \times w \times C'}$
$6 \times \{\text{Residual Block}\} \rightarrow \mathbb{R}^{h \times w \times C'}$	$6 \times \{\text{Residual Block}\} \rightarrow \mathbb{R}^{h \times w \times C'}$
$\text{Conv2D} \rightarrow \mathbb{R}^{h \times w \times C''}$	$4 \times \{\text{Upsampling Block}\} \rightarrow \mathbb{R}^{H \times W \times C}$
$H = W = 256, \quad h = w = 16, \quad C = 3, \quad C' = 256, \quad C'' = 512$	

A.2 VQGAN GENERATION

Hyperparameters In VQGAN generation task, we train the generator model (*i.e.*, VQ-VAE) with different VQ optimization methods for 100 epochs over bitrates of $B = \{8, 9, 10, 12\}$ with batch size of 8 and initial learning rate of $lr = 2.5 \cdot 10^{-5}$ that is halved after 50 and 75 epochs. The discriminator starts at epoch 50 when half of the training is passed. All GPT transformers (with different configurations in Table 4) are trained with batch size of 32 and initial learning rate of $lr = 4.5 \cdot 10^{-5}$ with cosine decay that reduces the learning rate to 1% of its initial value. For sampling from VQGAN, we use the temperature of $t = 1.0$ with different top- k values over different bitrates (see Table 5). The variance σ^2 of directional noise v_d for DiVeQ and SF-DiVeQ is set to $\sigma^2 = 10^{-2}$ for VQGAN generation.

¹https://github.com/google-deepmind/sonnet/blob/v2/examples/vqvae_example.ipynb

²<https://github.com/zalandoresearch/pytorch-vq-vae>

When training the generator (VQ-VAE) model for all VQ optimization methods (except SF-DiVeQ), we apply our proposed codebook replacement ([Sec. B.1](#)) actively during training in two phases of

$$\begin{cases} iter \leq 5000 & ; \text{replacement after each 50 training iterations} \\ 5000 < iter \leq N_{iter} - 3000 & ; \text{replacement after each 300 training iterations} \end{cases}$$

, where $iter$ is the training iteration number, and N_{iter} is the total number of training iterations. The discarding threshold for the replacement equals 0.01 which means that the codebook replacement discards the codewords which are used less than 1% during the period that the replacement is done.

Models architecture For the VQGAN generation task, we use the same architectures as in [Esser et al. \(2021\)](#) for the generator (VQ-VAE), discriminator, and transformer models. We adopt the implementation from [dome272 GitHub repository](#)³ with some suggested minor modifications from this repository⁴.

Training transformers For all data sets, the GPT transformer is trained with batch size of 32 and initial learning rate of $lr = 4.5 \cdot 10^{-5}$ with a cosine decay that reduces the learning rate to 1% of its initial value over the coarse of training, *i.e.* $lr = 4.5 \cdot 10^{-7}$. In addition, we adopt a linear warm-up for the learning rate such that lr starts with a really small value and linearly reaches to its initial learning rate during the first 10k training iterations.

When training the transformer, one important variable to tune is the probability of keeping the ground truth indices that is called $p_{\text{keep}} \in (0, 1)$. In other words, the true indices of the tokens are masked with the probability of $1 - p_{\text{keep}}$. The reason to mask the true tokens is that it helps the transformer to become more robust against its own mistakes, such that after one wrong prediction during sampling, it does not propagate this error with a snowball effect to all other next predictions. In addition, masking true tokens helps the transformer not to memorize the token dependencies, which leads to better generalization and prevents overfitting.

For all transformers, we set a scheduled masking rate that increases the p_{keep} from the initial value of 0.5 to 0.95 over training epochs. In the beginning, we mask half the the true token indices forcing the transformer to learn large missing parts, *i.e.*, the transformer learns dependencies of coarse structures in the images. As the training goes to the end, a small portion of token indices will be masked, and as a result, the transformer learns dependencies of finer details in the images.

To train the transformers for FFHQ and CELEBA-HQ data sets, we use the configurations suggested in [Esser et al. \(2021\)](#). Also, we use the same transformer configuration for LSUN Bedroom and LSUN Church data sets. Since AFHQ data set is much smaller than the other data sets (containing 15803 images), we adopted a different and lighter transformer for it. [Table 4](#) shows the transformers configuration for different data sets. Note that when experimenting over different VQ bitrates (or codebook sizes), we keep the transformer configuration fixed.

Table 4: Configurations used to train the transformers for different data sets in VQGAN generation task. n_{layer} is the the number of transformer blocks, n_{head} refers to the number of attention heads, $D_{\text{embedding}}$ is the dimensionality of the transformer embeddings, s is the length of the sequence, $\# \text{params}$ is the number of transformer parameters, dropout refers to the dropout rate used to train the transformer, and D_{latent} is the dimensionality of each codeword.

Data set	n_{layer}	n_{head}	$D_{\text{embedding}}$	length (s)	dropout	$\# \text{params} [M]$	D_{latent}
CELEBA-HQ	28	16	1024	256	0.1	≈ 355	256
FFHQ	28	16	1024	256	0.1	≈ 355	256
LSUN Bedroom	28	16	1024	256	0.1	≈ 355	256
LSUN Church	28	16	1024	256	0.1	≈ 355	256
AFHQ	20	16	768	256	0.1	≈ 143	256

³<https://github.com/dome272/VQGAN-pytorch>

⁴<https://github.com/aa1234241/vqgan>

It is noteworthy to mention again here that since having more data favors for VQGAN generation task, for CELEBA-HQ and AFHQ data sets, we use the full data sets to train the generator (VQ-VAE) and transformer, and to compute the FID scores. CELEBA-HQ and AFHQ full data sets contain 30k and 15803 images, respectively. However, for FFHQ and both LSUN data sets, we only consider the train sets that contain 56k images.

Sampling from VQGAN When the transformers are trained, for each data set, we sample the same number of images as the train set (full data sets for CELEBA-HQ and AFHQ) with the configuration mentioned in [Table 5](#). To compute the FID score between the train set and new sampled images, we use clean-fid from *GaParmar* GitHub repository⁵.

[Table 5](#): Configuration used for generating images from trained VQGAN models for different bitrates. This configuration is fixed for different data sets.

VQ bitrate	Codebook size (K)	Sampling temperature (t)	top- k
8	256	1.0	75
9	512	1.0	150
10	1024	1.0	300
12	4096	1.0	300

A.3 GUMBEL-SOFTMAX IMPLEMENTATION

For both VQ-VAE compression and VQGAN generation tasks, the implementation of Straight-Through Gumbel-Softmax (ST-GS) is adopted from *karpathy* GitHub repository⁶. The ST-GS temperature τ is exponentially annealed from $\tau_{\text{start}} = 1$ to $\tau_{\min} = 0.1$ over training epochs such that

$$\tau = \max\{\tau_{\text{start}} * \eta^{\text{epoch}}, \tau_{\min}\}; \quad \eta = \left(\frac{\tau_{\min}}{\tau_{\text{start}}}\right)^{\frac{1}{N_{\text{epochs}}}}, \quad (13)$$

where η is the decay rate for the temperature annealing, epoch is the current epoch number, and N_{epochs} is the total number of training epochs.

A.4 ROTATION TRICK IMPLEMENTATION

We adopt the implementation of the Rotation Trick (RT) from *lucidrains* GitHub repository⁷. Following what is suggested in the RT paper ([Fifty et al., 2025](#)), for our VQ-VAE compression task we first trained the VQ-VAE by updating the codebook via Exponential Moving Averages (EMA) with the decay rate of $\gamma = 0.8$. Since the results were not good enough, we changed the decay rate to $\gamma = 0.99$, which slightly improved the quality of reconstructions. However, we finally noticed that by skipping the EMA for updating the codebook and using codebook loss (2nd loss term in [Eq. \(2\)](#)), we achieve the best possible performance that we can get from the Rotation Trick in our VQ-VAE compression task. For the VQGAN generation task, the RT paper suggested updating the VQ codebook via gradients using codebook loss. Therefore, for both of our VQ-VAE compression and VQGAN generation tasks, we train the RT using the codebook loss.

A.5 SUGGESTIONS FOR TRAINING THE CODEBOOK USING SF-DIVEQ

As discussed in [Sec. 4](#), for training the codebook via SF-DiVeQ, it is recommended to skip quantization for several initial epochs (or several thousand training iterations) in order to reach a more stable latent representation that already passed its initial big distribution shifts. After that, initialize the codebook vectors with the average of recent latent vectors and start discretizing the latent representation using SF-DiVeQ approach.

⁵<https://github.com/GaParmar/clean-fid>

⁶<https://github.com/karpathy/deep-vector-quantization>

⁷<https://github.com/lucidrains/vector-quantize-pytorch>

For the initialization of the codebook, it is highly recommended to follow these two guidelines; (*i*) take the very recent latent vectors from the last 20 to 50 training iterations as the set of vectors for initialization, (*ii*) according to the batch size, initialize each codebook vector to be the average of at least 20 to 40 recent latent vectors. For instance, suppose the encoder compresses the input image of size 256×256 to a latent of size 256×256 . So, for a batch size of 8, each training iteration would generate $8 \times 16 \times 16 = 2048$ latent vectors. Hence, for a codebook of size of $K = 1024$, to initialize each codebook vector with the average of 40 latent vectors, we need to capture the latent vectors of the last 20 training iterations such that

$$\frac{N_{\text{latents}}}{K} = 40 \Rightarrow N_{\text{iters}} = \frac{N_{\text{latents}}}{2048} = \frac{40 \times 1024}{2048} = 20 \quad (14)$$

where N_{latents} is the number of recent captured latent vectors, and N_{iters} is the number of recent training iterations required for obtaining N_{latents} latent vectors. Therefore, for smaller batch sizes, it is better to capture more training iterations compared to larger batch sizes.

The most important point in training codebook via SF-DiVeQ is to track the codebook usage percentage over training epochs (similar to the plot in Fig. 25). If quantization with SF-DiVeQ is started after big latent distribution shifts and by capturing enough recent latent vectors, the initial codebook usage would be a high value (approximately close to the maximum possible value that equals the number of codebook vectors K). Then, according to the SF-DiVeQ property, it pulls the codewords that are outside the latent representation into inside during training iterations. Therefore, the codebook usage will reach its maximum value K after some epochs of starting quantizing the latent space, and will be kept at its maximum until the end of training. Fig. 25 presents an example of how codebook usage percentage looks like for a proper SF-DiVeQ training.

B OTHER PROPOSALS

B.1 NEW PROPOSED CODEBOOK REPLACEMENT

According to Huh et al. (2023), codebook replacement is a suitable solution to avoid *codebook collapse* (Mentzer et al., 2023) for different VQ optimization techniques. In NSVQ (Vali & Bäckström, 2022), a codebook replacement approach is proposed such that after a period of training iterations, unused codewords are replaced with a permutation of some used codewords. Selection from the used codewords is completely random, regardless of the importance of the used codewords. However, in this paper, we propose a new codebook replacement technique that replaces unused codewords with the used ones based on the importance of the used codewords. In other words, we use an *importance sampling* strategy that computes the probability of codewords occurrences (during a period of training iterations), and replaces the unused codewords by selecting from the used codewords according to these probabilities. Therefore, a codeword with a higher occurrence probability is more likely to be selected for the replacement. This is a logical approach for replacement, as in quantization applications, the main goal is to reduce the average quantization error, and when the number of assigned inputs to a codeword is high, it is better to define a new codeword in that region to reduce the average quantization error.

Fig. 8 compares the perplexity (or average codebook usage) of our proposed codebook replacement and NSVQ’s replacement technique (Vali & Bäckström, 2022) in the VQ-VAE compression of the CELEBA-HQ data set using the DiVeQ approach when the VQ bitrate is 11 (with $K = 2^{11} = 2048$ codewords). For each training iteration, perplexity shows the number of selected codewords involved in quantizing that batch of data, and it is computed as

$$\text{perplexity} = \exp(H(\mathcal{C})) \quad \text{s.t.} \quad H(\mathcal{C}) = - \sum_{k=1}^K p_k \cdot \log p_k, \quad (15)$$

where $H(\mathcal{C})$ is the entropy of the codebook index distribution, and p_k is the empirical usage probability of the k -th codebook vector. Fig. 8 is plotted for the first 1500 training iterations, while the curves are smoothed via Blackman windowing with a window size of 11 and normalized by their maximum value. According to the figure, our proposed codebook replacement reaches higher perplexity values faster than NSVQ’s replacement approach, and as a result, it provides more training opportunity for the active codewords to be trained for more training iterations.

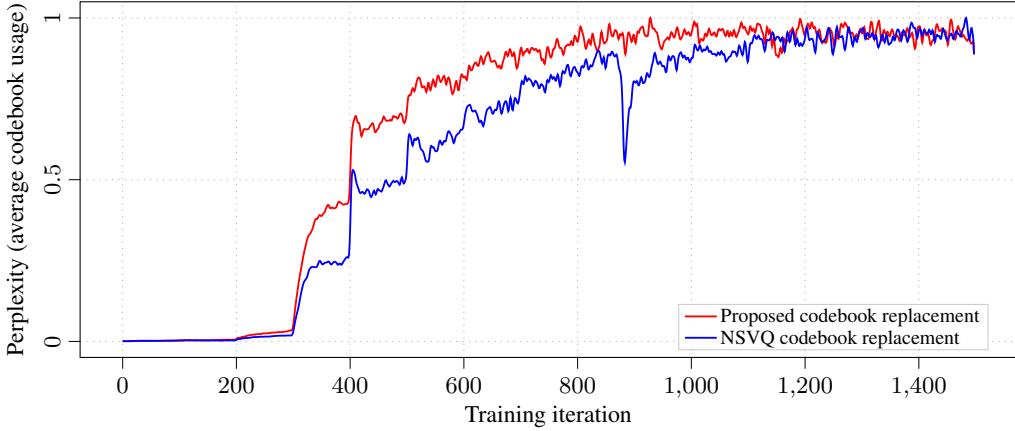


Figure 8: Proposed codebook replacement achieves a more stable codebook utilization faster than NSVQ’s replacement method. Comparison of the perplexity of our proposed codebook replacement with NSVQ’s replacement technique for the initial 1500 iterations when training the VQ-VAEs on CELEBA-HQ data set using the DiVeQ approach. The codebook size equals $K = 2048$.

Fig. 9 shows the quantitative results of the reconstructions when using our proposed codebook replacement and NSVQ’s replacement method, shown in Fig. 8. We observe that our proposed codebook replacement results in slightly better performance by having higher SSIM, PSNR, and lower LPIPS values. Note that based on our investigations, our proposed codebook replacement can result in similar or better performance than NSVQ’s replacement over different experiments. Therefore, we use our proposed codebook replacement for all VQ optimization techniques (*i.e.*, STE, EMA, RT, ST-GS, NSVQ, and DiVeQ) and for all experiments in this paper.

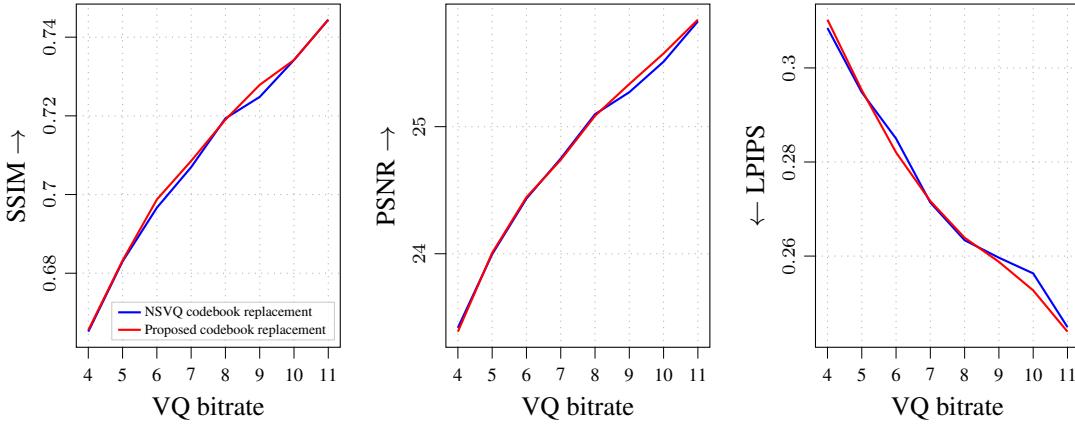


Figure 9: Proposed codebook replacement improves the quality of reconstructions compared to NSVQ’s replacement method. Quantitative comparison of reconstructed images from the CELEBA-HQ test set when using our proposed codebook replacement vs. NSVQ’s replacement method. The VQ-VAEs are trained via DiVeQ approach over different bitrates.

B.2 PROPOSED VARIANTS OF DiVeQ AND SF-DiVeQ

Apart from what is proposed for the formulation of our proposed DiVeQ (Eq. (8)) and SF-DiVeQ (Eq. (12)), we also propose two new variants of them by skipping the directional noise \mathbf{v}_d and using the stop gradient operator (*i.e.*, `detach` in PyTorch). We call these two variants as DiVeQ-*detach* and SF-DiVeQ-*detach*, and their formulation is

$$\begin{aligned} \text{DiVeQ-} &\text{detach : } \mathbf{z}_q = \mathbf{z} + \|\mathbf{c}_{i^*} - \mathbf{z}\|_2 \cdot sg\left[\frac{\mathbf{c}_{i^*} - \mathbf{z}}{\|\mathbf{c}_{i^*} - \mathbf{z}\|_2}\right], \\ \text{SF-DiVeQ-} &\text{detach : } \mathbf{z}_q = \mathbf{z} + \|\mathbf{c}_{i^*} - \mathbf{z}\|_2 \cdot sg\left[\frac{(1 - \lambda_{i^*})(\mathbf{c}_{i^*} - \mathbf{z})}{\|\mathbf{c}_{i^*} - \mathbf{z}\|_2}\right] \\ &+ \|\mathbf{c}_{i^*+1} - \mathbf{z}\|_2 \cdot sg\left[\frac{\lambda_{i^*}(\mathbf{c}_{i^*+1} - \mathbf{z})}{\|\mathbf{c}_{i^*+1} - \mathbf{z}\|_2}\right]. \end{aligned} \quad (16)$$

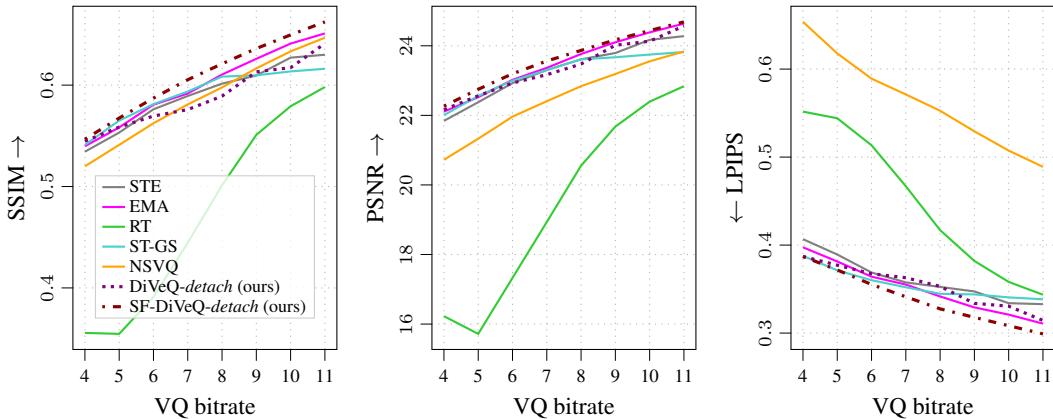


Figure 10: **Variants of our proposed methods: SF-DiVeQ-*detach* consistently improves image reconstruction and DiVeQ-*detach* maintains the reconstruction quality.** Quantitative comparison of reconstructed images from the AFHQ test set using variants of our proposed methods vs. other VQ optimization approaches. Each curve is the average of the metric over all test set reconstructions and over three different individual runs.

We experiment with these two variants only in the VQ-VAE compression task to evaluate how they perform compared to DiVeQ and SF-DiVeQ, and also other existing techniques. We train the VQ-VAE compression model with a learning rate of $5.5 \cdot 10^{-4}$ and a batch size of 32 over 100 epochs on the AFHQ data set. Fig. 10 shows the quality of reconstructions for DiVeQ-*detach* and SF-DiVeQ-*detach* compared to other approaches of STE, EMA, RT, ST-GS, and NSVQ. According to the figure, similar to its original version, SF-DiVeQ-*detach* consistently outperforms other methods for all three objective metrics and over all different VQ bitrates. However, DiVeQ-*detach* performs comparable to other VQ methods. In addition, Fig. 11 compares the quality of reconstructions of DiVeQ-*detach* and SF-DiVeQ-*detach* with their original approaches of DiVeQ and SF-DiVeQ. According to Fig. 11, DiVeQ performs clearly superior to its variant (*i.e.*, DiVeQ-*detach*), while SF-DiVeQ and its variant perform almost similarly.

According to VQ-VAE compression experiments on other data sets of CELEBA-HQ, FFHQ, LSUN Bedroom, and LSUN Church, we find that DiVeQ and SF-DiVeQ always result in higher quality reconstructions compared to their variants. The performance gap is not significant, and even in some cases (like SF-DiVeQ in Fig. 11), DiVeQ-*detach* and SF-DiVeQ-*detach* can perform close to their original versions. The main difference is that the *detach* variants do not need to set the variance σ^2 (in directional noise \mathbf{v}_d). However, we do not consider this an advantage, since the variance in DiVeQ and SF-DiVeQ can be used to control the amount of generalization for the downstream task.

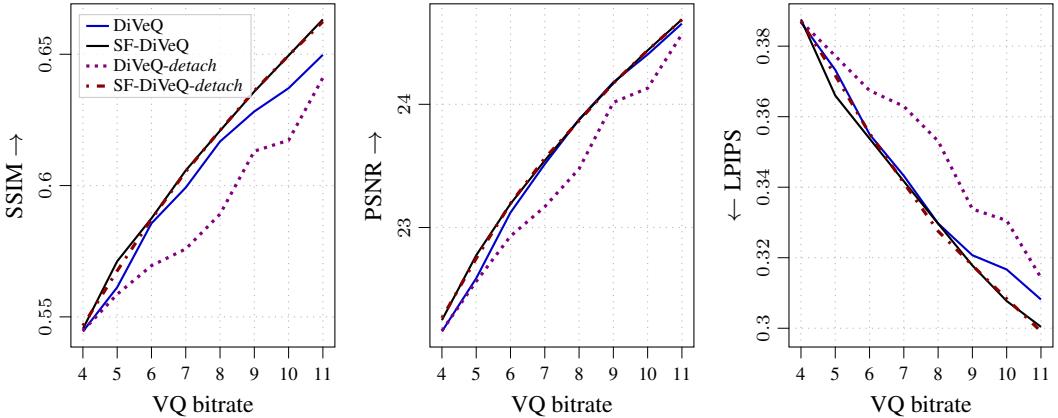


Figure 11: **Variants of our proposed methods:** Regarding reconstruction quality, SF-DiVeQ-detach performs almost similar and DiVeQ-detach performs worse than their original versions. Quantitative comparison of reconstructed images from the AFHQ test set using variants of our proposed methods vs. their original versions. Each curve is the average of the metric over all test set reconstructions and over three different individual runs.

C ADDITIONAL RESULTS

C.1 QUANTITATIVE RESULTS ON OTHER DATA SETS FOR VQ-VAE COMPRESSION

Apart from the results obtained for the AFHQ data set in Fig. 6, we also assess the quality of reconstructions in VQ-VAE compression task for CELEBA-HQ, FFHQ, LSUN Bedroom, and LSUN Church data sets. As mentioned earlier, for all data sets, we train the VQ-VAE models using the configurations mentioned in Sec. A.1. After training, we compress and reconstruct the test set images by doing hard VQ (using *argmin*) in the latent space. Then, we compute the SSIM, PSNR, and LPIPS metrics for the reconstructions by having the original ground-truth image available. Figs. 12 to 15 show the results for CELEBA-HQ, FFHQ, LSUN Bedroom, and LSUN Church data sets, respectively. Note that in the plots, the reported value for each metric is the average of that metric over all test set reconstructions and over three different individual runs.

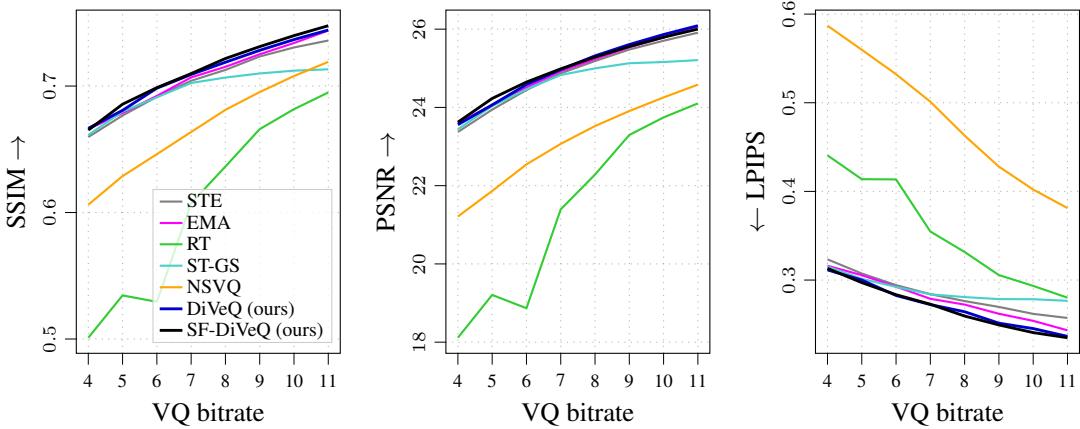


Figure 12: **The proposed DiVeQ and SF-DiVeQ lead to consistent improvement in image reconstruction quality.** Quantitative comparison of reconstructed images from the CELEBA-HQ test set for different VQ optimization methods over different bitrates. Each curve is the average of the metric over all test set reconstructions and over three different individual runs.

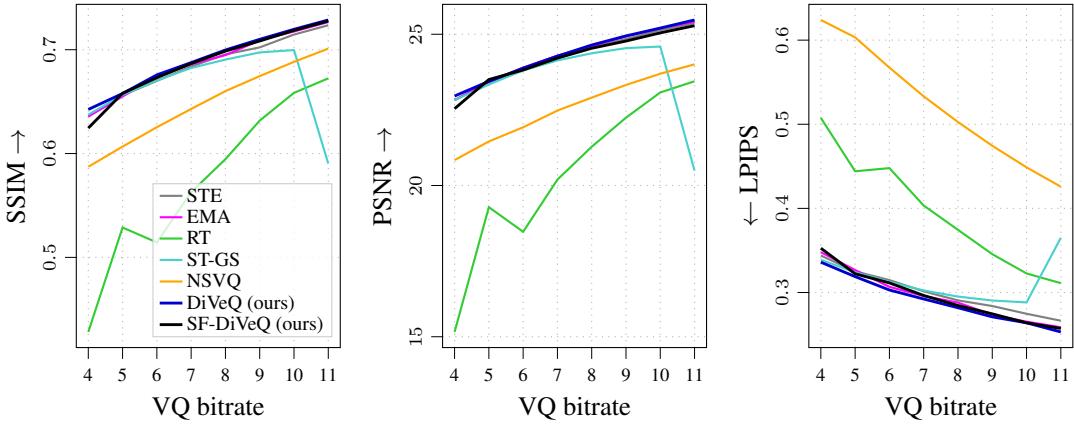


Figure 13: **The proposed DiVeQ and SF-DiVeQ lead to consistent improvement in image reconstruction quality.** Quantitative comparison of reconstructed images from the FFHQ test set for different VQ optimization methods over different bitrates. Each curve is the average of the metric over all test set reconstructions and over three different individual runs.

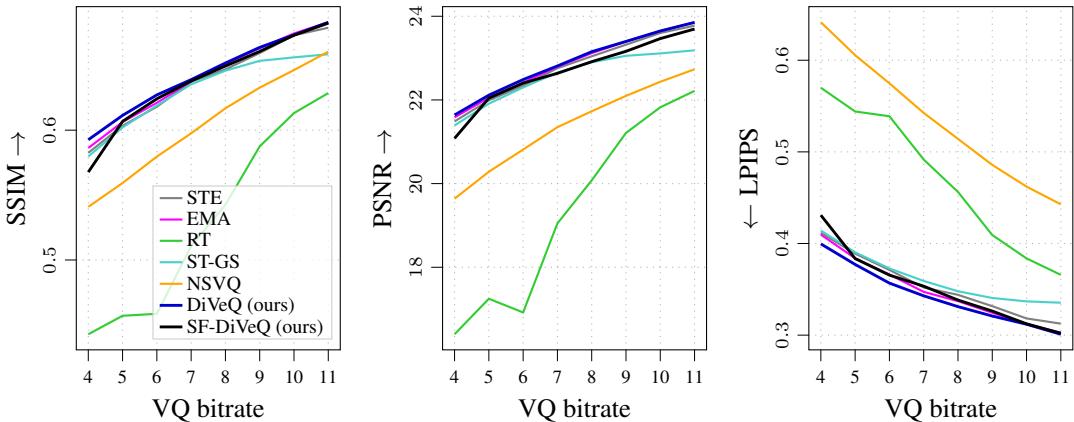


Figure 14: **The proposed DiVeQ and SF-DiVeQ lead to consistent improvement in image reconstruction quality.** Quantitative comparison of reconstructed images from the LSUN Bedroom test set for different VQ optimization methods over different bitrates. Each curve is the average of the metric over all test set reconstructions and over three different individual runs.

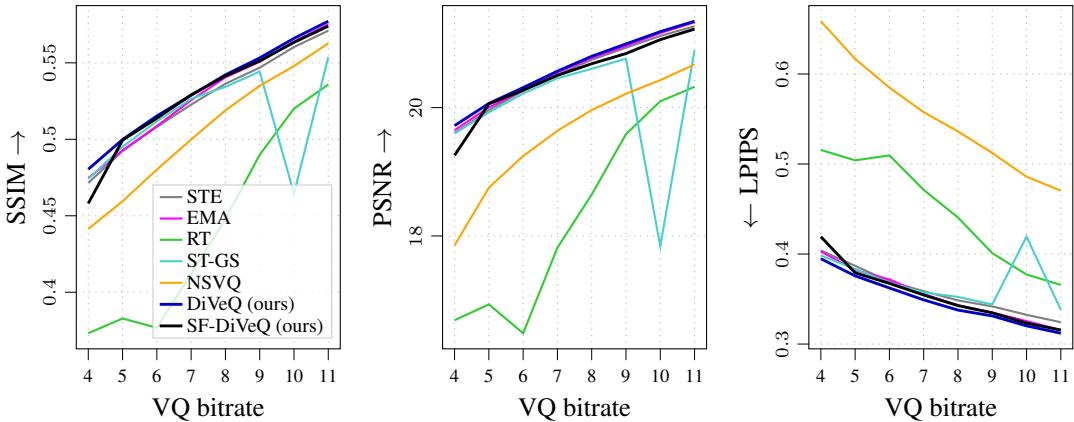


Figure 15: **The proposed DiVeQ and SF-DiVeQ lead to consistent improvement in image reconstruction quality.** Quantitative comparison of reconstructed images from the LSUN Church test set for different VQ optimization methods over different bitrates. Each curve is the average of the metric over all test set reconstructions and over three different individual runs.

C.2 QUANTITATIVE RESULTS ON OTHER DATA SETS FOR VQGAN GENERATION

Apart from the FID values obtained for CELEBA-HQ data set in [Table 2](#), we also evaluate the quality of generations for other data sets by computing the FID scores. As mentioned earlier, for all data sets, we train the VQGAN generator (*i.e.*, VQ-VAE), discriminator, and transformer models using the configurations mentioned in [Sec. A.2](#). Then after the models are trained, we sample new generations from them and compute the FID value between the original training set and the generated samples (see [Table 5](#) for sampling configuration). [Table 6](#) presents the obtained FID scores for FFHQ, AFHQ, LSUN Bedroom and LSUN Church data sets using different VQ optimization methods and over various bitrates. [Figs. 26 to 30](#) provide the qualitative comparison of VQGAN generations for all data sets when VQ bitrate is 9 (or codebook size equals $K = 512$).

Table 6: The proposed DiVeQ and SF-DiVeQ robustify training and maintain the generation quality for challenging low bitrates, where the generation task becomes more challenging. The table presents the $\text{FID} \downarrow$ scores for different VQ optimization methods and over different bitrates for FFHQ, LSUN Bedroom, LSUN Church, and AFHQ data sets.

Data set	Approach	VQ bitrate			
		8	9	10	12
FFHQ	STE	6.74	6.01	5.36	7.31
	EMA	6.53	6.57	6.15	619
	RT	14.2	7.19	6.52	4.73
	ST-GS	22.5	21.9	17.0	7.97
	NSVQ	98.0	88.8	83.8	65.1
	DiVeQ (ours)	6.70	5.61	5.11	8.22
	SF-DiVeQ (ours)	7.91	6.21	5.93	8.60
LSUN Bedroom	STE	5.27	4.89	4.96	7.54
	EMA	6.04	5.36	5.56	5.13
	RT	17.4	8.65	6.69	5.18
	ST-GS	35.9	37.2	36.4	23.0
	NSVQ	67.8	50.5	45.6	37.5
	DiVeQ (ours)	5.71	4.87	5.40	7.94
	SF-DiVeQ (ours)	5.96	6.01	5.40	7.79
LSUN Church	STE	4.33	3.74	4.09	4.91
	EMA	4.39	4.37	3.81	3.88
	RT	9.33	6.37	5.20	4.31
	ST-GS	24.4	26.9	26.5	13.7
	NSVQ	107	92.8	71.7	50.9
	DiVeQ (ours)	4.41	3.92	3.90	4.58
	SF-DiVeQ (ours)	4.35	3.99	3.56	4.40
AFHQ	STE	5.45	5.24	5.41	6.48
	EMA	5.12	5.56	5.23	6.95
	RT	6.88	6.31	5.55	5.54
	ST-GS	31.3	26.5	20.9	14.4
	NSVQ	92.0	81.3	72.2	47.1
	DiVeQ (ours)	6.30	5.63	5.78	7.33
	SF-DiVeQ (ours)	5.63	5.40	5.69	6.86

C.3 ABLATION ON BATCH SIZE

In this section, we study the effect of the batch size on different VQ optimization methods. To this end, in the VQ-VAE compression task, we train all different methods with batch sizes of 64 and 128 at the learning rate of $lr = 5.5 \cdot 10^{-4}$ on the AFHQ data set for 100 epochs. The learning rate is halved after 40 and 70 epochs. [Fig. 16](#) and [Fig. 17](#) show the quality of reconstructions for batch sizes 64 and 128, respectively. Note that in the plots, the reported value for each metric is the average of that metric over all test set reconstructions and over three different individual runs. Taking [Fig. 6](#) (for batch size 32), [Figs. 16 and 17](#) (for batch sizes 64 and 128) into account, we can claim that by increasing the batch size, the performance gap between our proposed methods and other approaches (except EMA) becomes larger. For larger batch sizes, EMA performs comparably but slightly worse

than our DiVeQ and SF-DiVeQ. The reason is that EMA updates the codebook with the average of latent vectors, and as a result, larger batch sizes yield less noisy and more representative updates.

In the VQ-VAE compression task, it is expected that the increase in VQ bitrate (or codebook size) always enhances the quality of reconstructions. However, this does not happen for some cases in Fig. 16 and Fig. 17. The reason is that the codebook representation \mathcal{C}_z is not well fitted to the latent distribution \mathcal{P}_z , and as a result, a misalignment between codebook and latent representations is happened (see Fig. 4, Sec. 4, and Sec. C.6 for more details). In Fig. 16, the misalignments happen for [ST-GS|bitrate=10, 11], [RT|bitrate=5], and in Fig. 16 they arise for [STE|bitrate=11], [EMA|bitrate=11], [ST-GS|bitrate=9, 11].

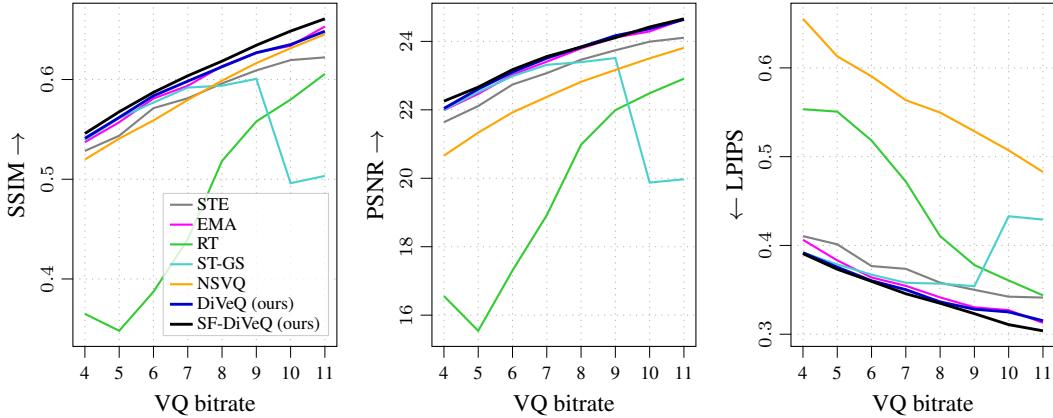


Figure 16: **Ablation on batch size: The proposed DiVeQ and SF-DiVeQ lead to consistent improvement in image reconstruction quality over different batch sizes (this figure, batch size = 64).** Quantitative comparison of reconstructed images from the AFHQ test set for different VQ optimization methods over different bitrates. Each curve is the average of the metric over all test set reconstructions and over three different individual runs.

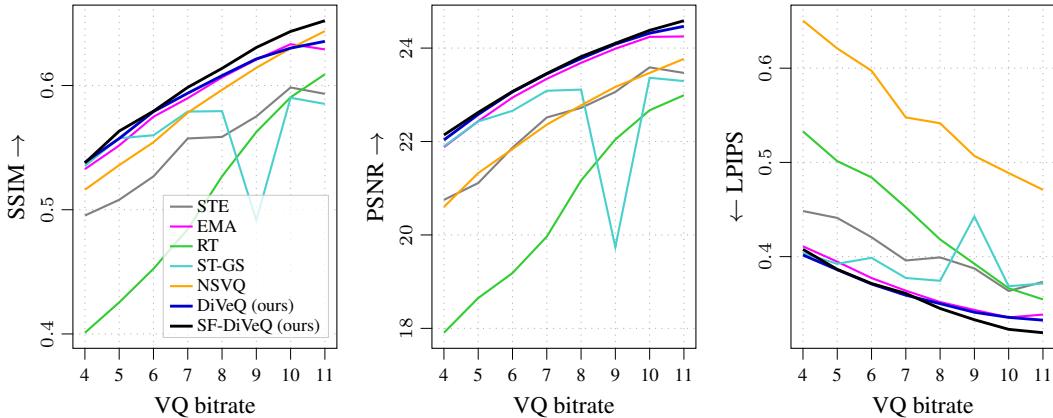


Figure 17: **Ablation on batch size: The proposed DiVeQ and SF-DiVeQ lead to consistent improvement in image reconstruction quality over different batch sizes (this figure, batch size = 128).** Quantitative comparison of reconstructed images from the AFHQ test set for different VQ optimization methods over different bitrates. Each curve is the average of the metric over all test set reconstructions and over three different individual runs.

C.4 ABLATION ON LEARNING RATE

In this section, we evaluate the performance of different VQ optimization techniques in the VQ-VAE compression task on the CELEBA-HQ data set using different learning rates. The ablation is performed over two new learning rates of $lr = \{10^{-3}, 10^{-4}\}$ while training the models for 100 epochs with the batch size of 32. During training, the learning rates are halved after 40 and 70 epochs. [Fig. 18](#) and [Fig. 19](#) show the quality of reconstructions when $lr = 10^{-3}$ and $lr = 10^{-4}$, respectively. Note that in the plots, the reported value for each metric is the average of that metric over all test set reconstructions and over three different individual runs. According to [Fig. 18](#), DiVeQ and SF-DiVeQ outperform other methods, achieving higher SSIM, PSNR values and obtaining lower LPIPS values. According to [Huh et al. \(2023\)](#), when increasing the learning rate in VQ-VAEs, the model is highly prone to misaligned codebook and latent representations. Therefore, in [Fig. 18](#), we find that some misalignments are happened for [ST-GS|bitrates=6, 8, 9, 10, 11], [EMA|bitrate=7], and [DiVeQ|bitrate=11]. The misalignments are visible as sudden jumps in the objective metrics, and happen in cases where the increase in VQ bitrate does not lead to enhancement in the objective metrics (more details in [Fig. 4](#), [Sec. 4](#), and [Sec. C.6](#)).

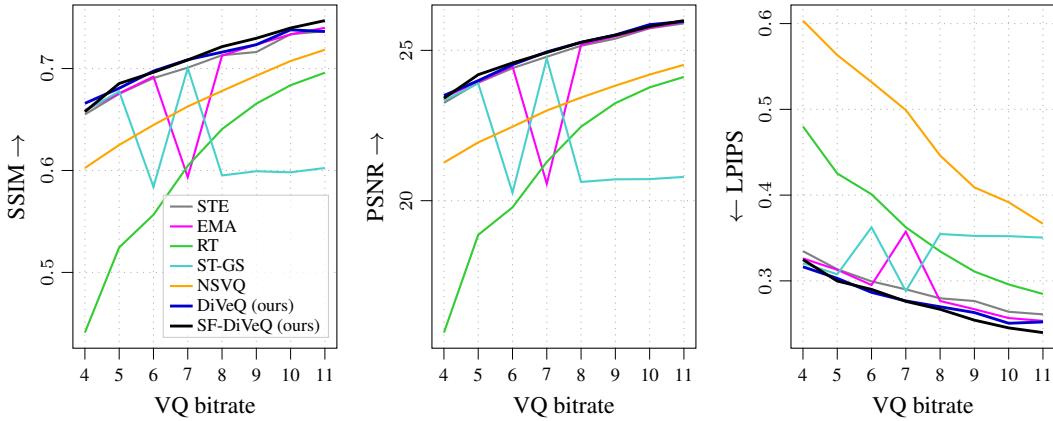


Figure 18: **Ablation on learning rate: The proposed DiVeQ and SF-DiVeQ lead to consistent improvement in image reconstruction quality over different learning rates (this figure, $lr = 10^{-3}$).** Quantitative comparison of reconstructed images from the CELEBA-HQ test set for different VQ optimization methods over different bitrates. Each curve is the average of the metric over all test set reconstructions and over three different individual runs.

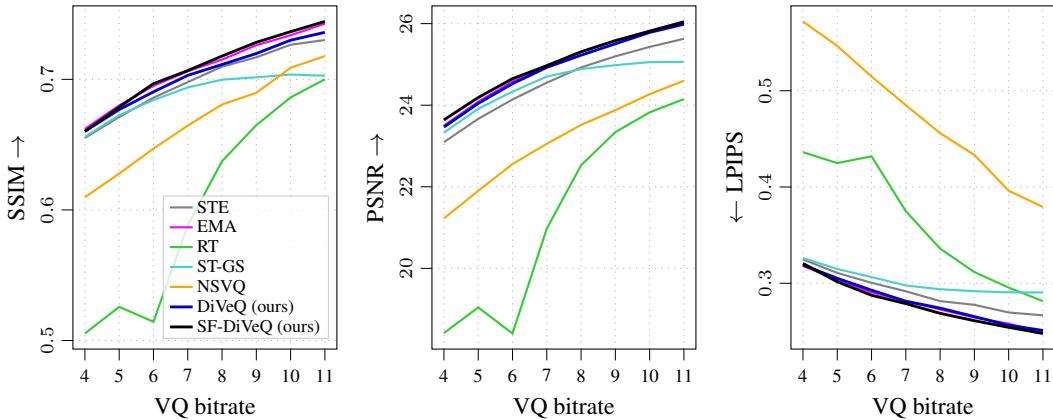


Figure 19: **Ablation on learning rate: The proposed DiVeQ and SF-DiVeQ lead to consistent improvement in image reconstruction quality over different learning rates (this figure, $lr = 10^{-4}$).** Quantitative comparison of reconstructed images from the CELEBA-HQ test set for different VQ optimization methods over different bitrates. Each curve is the average of the metric over all test set reconstructions and over three different individual runs.

In Fig. 19, again, our proposed methods obtain better objective metrics than the other methods, while performing comparable EMA. The reason is that EMA updates the codebook without gradient descent, and thus, there is no explicit learning rate defined for its codebook learning. The decay rate is the hyperparameter responsible for the speed of the codebook updates. As we do not have a proper definition on how to set the decay rate corresponding to a specific learning rate, we use a fixed decay rate of $\gamma = 0.99$ for all experiments in the ablation study on learning rates (*i.e.*, Figs. 18 and 19). In addition, a misalignment is happened for [RT | bitrate=6] in Fig. 19.

C.5 ABLATION ON VARIANCE σ^2 OF DiVeQ AND SF-DiVeQ

To assess the impact of variance σ^2 in our proposed DiVeQ and SF-DiVeQ, we did an ablation study using different variances in the VQ-VAE compression task on the CELEBA-HQ data set. Fig. 20 and Fig. 21 show the quality of reconstructions over different variances of $\sigma^2 = \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$ for DiVeQ and SF-DiVeQ techniques, respectively. Note that in the plots, the reported value for each metric is the average of that metric over all test set reconstructions and over three different individual runs. Conforming with what presented in Fig. 3, results in Figs. 20 and 21 confirm that to achieve higher reconstruction quality, the variance should be small to have more precise nearest-codeword mappings of inputs z to the selected codewords c_{i^*} . That is why in Figs. 20 and 21, when $\sigma^2 = 10^{-1}$ (which is still considered a high value for the variance), the reconstructions are of low quality.

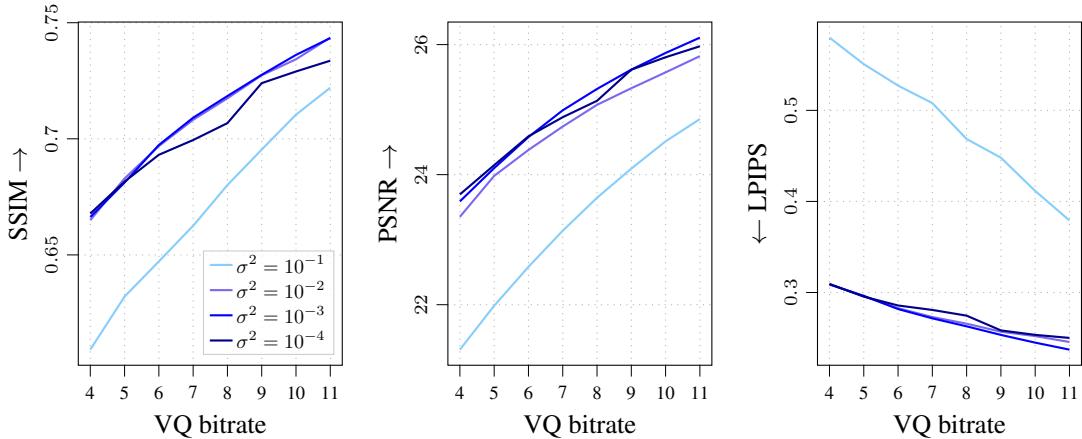


Figure 20: Ablation on variance σ^2 for DiVeQ: Quality of image reconstruction is improved when reducing the variance σ^2 . Quantitative comparison of reconstructed images from the CELEBA-HQ test set for DiVeQ optimization technique over different variances. Each curve is the average of the metric over all test set reconstructions and over three different individual runs.

On the other hand, we observe more or less comparable quality for the reconstructions when $\sigma^2 = \{10^{-2}, 10^{-3}, 10^{-4}\}$. Based on the results, especially for DiVeQ (Fig. 20), we can conclude that $\sigma^2 = 10^{-3}$ is the best value to choose for the VQ-VAE compression task. Furthermore, we notice that a lower σ^2 (that leads to more precise mappings) does not necessarily result in better performance. Hence, we can think of σ^2 as a hyperparameter that controls the amount of generalization for the VQ nearest-codeword assignments. This is the only hyperparameter of our proposed DiVeQ and SF-DiVeQ that is required to be small (*i.e.*, $\sigma^2 \in [10^{-3}, 10^{-2}]$). There is no necessity to spend time to properly tune this hyperparameter, as small values should generally work for different applications, and the performance is not significantly sensitive to its selection.

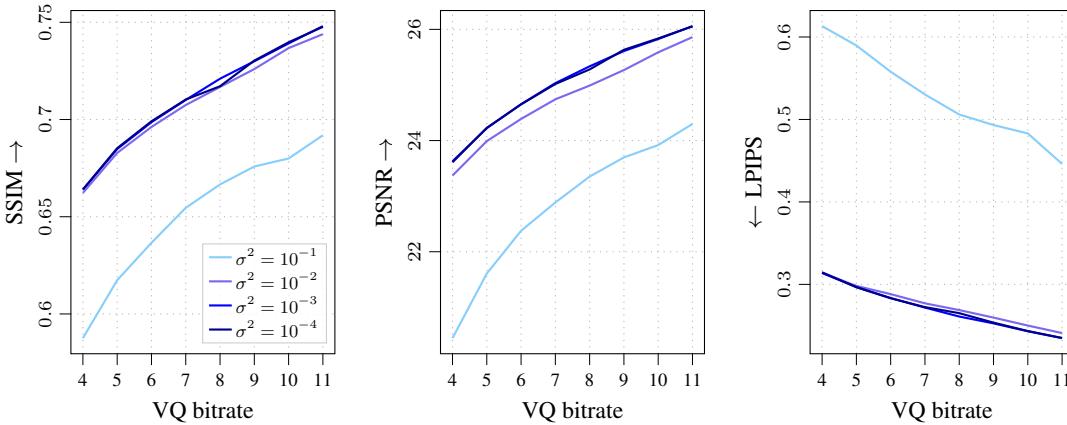


Figure 21: **Ablation on variance σ^2 for SF-DiVeQ: Quality of image reconstruction is improved when reducing the variance σ^2 .** Quantitative comparison of reconstructed images from the CELEBA-HQ test set for SF-DiVeQ optimization technique over different variances. Each curve is the average of the metric over all test set reconstructions and over three different individual runs.

C.6 MISALIGNMENT OF CODEBOOK AND LATENT REPRESENTATIONS

As discussed in Sec. 4, when training the VQ-VAE for the compression task using different learning rates and batch sizes, there would be cases in which the Shannon’s rate-distortion theory (Shannon, 1959) does not hold. In other words, in these cases when the VQ bitrate (or codebook size) is increased, the quality of reconstructions will not be improved with respect to the objective metrics. In Fig. 22, we provide the quantitative results for the cases that the rate-distortion theory does not hold, and as a result, a misalignment between codebook and latent representations is happened (see Fig. 4). The curves represent the results for VQ-VAE trained on CELEBA-HQ data set with various training setups.

Misalignments are highlighted with *red circles* in Fig. 22, which refer to the cases of $[\text{STE}|lr = 10^{-4}, \text{bitrate}=11]$, $[\text{RT}|lr = 5.5 \cdot 10^{-4}, \text{bitrate}=6]$, $[\text{ST-GS}|lr = 10^{-3}, \text{bitrate}=8]$, $[\text{NSVQ}|lr = 10^{-3}, \text{bitrate}=11]$, and $[\text{DiVeQ}|lr = 10^{-3}, \text{bitrate}=11]$. Note that for the EMA approach, we consider the misalignment happened in Fig. 17, when the bath size is 128 and $lr = 5.5 \cdot 10^{-4}$. According to all of our experiments, for our proposed SF-DiVeQ, the rate-distortion theory always hold true and as a result, no misalignment happens for SF-DiVeQ. For the sake of comparison, we consider the case of $[\text{SF-DiVeQ}|lr = 10^{-3}, \text{bitrate}=11]$ for plotting in Fig. 4. Note that in difference to other plots in the paper, Fig. 22 shows the results for only one individual experiment, not the average on several different individual runs.

The reported values in Fig. 4 show the distortion per bit ($D_{\text{per-bit}} \downarrow$) for each of the quantization cases. Since the misalignments of different VQ methods happen in different bitrates (or codebook sizes), it is not fair to report the ordinary distortion for each of the quantization cases. Because, smaller codebook sizes result in higher amounts of distortion. Therefore, for quantitative evaluation of quantization cases in Fig. 4, we measure $D_{\text{per-bit}}$ metric that measures how much distortion remains per bit. $D_{\text{per-bit}}$ is computed as

$$D_{\text{per-bit}} = \frac{D}{H(\mathcal{C})} \quad \text{s.t.} \quad D = \frac{1}{N} \sum_{n=1}^N \|z_n - \hat{z}_n\|_2^2 \quad \text{and} \quad H(\mathcal{C}) = - \sum_{k=1}^K p_k \cdot \log_2 p_k, \quad (17)$$

where D is the quantization distortion, $H(\mathcal{C})$ is entropy of the codebook index distribution, and p_k refers to the empirical usage probability of the k -th codebook vector. $D_{\text{per-bit}}$ is a suitable and fair metric to use to compare the case of misalignments, as it reflects the distortion (D) mixed with the effective codebook usage ($H(\mathcal{C})$).

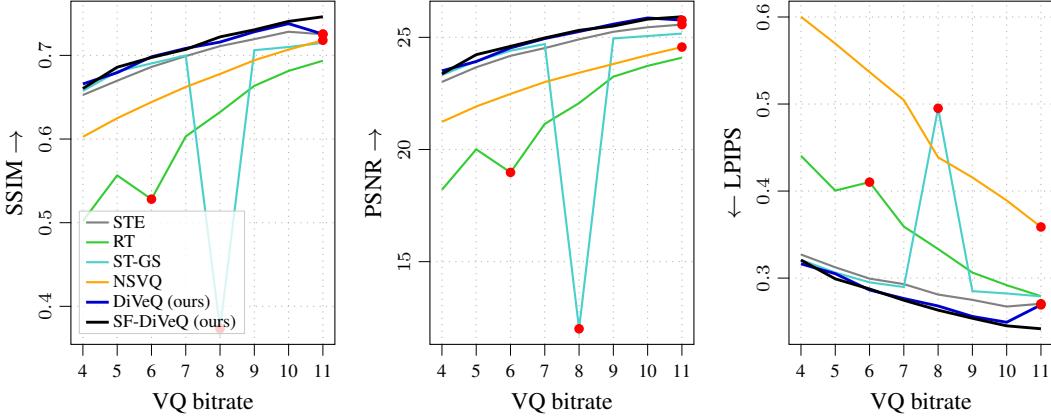


Figure 22: **Misalignment of codebook and latent representations happens where the increase in bitrate does not improve the objective metrics.** The figure shows the quantitative results in VQ-VAE compression experiments on CELEBA-HQ data set for different VQ methods. Cases prone to misalignment are highlighted in *red circles*. Each curve shows the results only for one individual run. For the EMA approach, we consider the misalignment happened in Fig. 17 at VQ bitrate of 11.

C.7 DIVEQ AND SF-DIVEQ IN RESIDUAL VECTOR QUANTIZATION

As mentioned in Sec. 1 of the paper, DiVeQ and SF-DiVeQ can also be used to train other variants of vector quantization like Residual VQ (RVQ, Chen et al., 2010). In different to ordinary vector quantization, RVQ uses multiple codebooks to quantize a distribution. Suppose an RVQ with three codebooks of $\{\mathcal{C}^1, \mathcal{C}^2, \mathcal{C}^3\}$. The first stage of RVQ quantizes the input \mathbf{z} to $\hat{\mathbf{z}}_1$ using the first codebook \mathcal{C}^1

$$\hat{\mathbf{z}}_1 = \mathbf{c}_{i^*}^1 = \arg \min_{\mathbf{c}_j^1} \|\mathbf{z} - \mathbf{c}_j^1\|_2 \quad \text{s.t.} \quad \mathbf{c}_j^1 \text{ is the } j\text{-th codeword of } \mathcal{C}^1, \quad (18)$$

and computes the residual of the first stage as $\mathbf{r}_1 = \mathbf{z} - \hat{\mathbf{z}}_1$. Similarly in the second stage, RVQ takes \mathbf{r}_1 as input and quantizes it to $\hat{\mathbf{r}}_1$ using the second codebook \mathcal{C}^2

$$\hat{\mathbf{r}}_1 = \mathbf{c}_{i^*}^2 = \arg \min_{\mathbf{c}_j^2} \|\mathbf{r}_1 - \mathbf{c}_j^2\|_2 \quad \text{s.t.} \quad \mathbf{c}_j^2 \text{ is the } j\text{-th codeword of } \mathcal{C}^2, \quad (19)$$

and computes the residual of the second stage as $\mathbf{r}_2 = \mathbf{r}_1 - \hat{\mathbf{r}}_1$. Then, for the third (or last) stage, RVQ takes \mathbf{r}_2 as input and quantizes it to $\hat{\mathbf{r}}_2$ using the third codebook \mathcal{C}^3

$$\hat{\mathbf{r}}_2 = \mathbf{c}_{i^*}^3 = \arg \min_{\mathbf{c}_j^3} \|\mathbf{r}_2 - \mathbf{c}_j^3\|_2 \quad \text{s.t.} \quad \mathbf{c}_j^3 \text{ is the } j\text{-th codeword of } \mathcal{C}^3. \quad (20)$$

Finally, RVQ computes the quantized input as

$$\hat{\mathbf{z}} = \mathbf{c}_{i^*}^1 + \mathbf{c}_{i^*}^2 + \mathbf{c}_{i^*}^3 \quad (21)$$

where $\hat{\mathbf{z}}$ is the final hard quantized version of the input \mathbf{z} .

In this section, we compare the performance of different VQ optimization techniques in the VQ-VAE compression task when quantizing the latent space of VQ-VAE by RVQ. We train the VQ-VAE models over 100 epochs with batch size of 32 and initial learning rate of $lr = 5.5 \cdot 10^{-4}$ that is halved after 40 and 70 epochs. In all experiments, we apply RVQ using three different codebooks over four different bitrates of $B = \{12, 18, 24, 30\}$. For instance, for the bitrate 30, we apply RVQ with three quantization stages each with the bitrate of 10 (or codebook size of 1024).

Fig. 23 and Fig. 24 show the quantitative comparison of different VQ optimization techniques when reconstructing the test set images of CELEBA-HQ and AFHQ, respectively. Note that in the plots, the reported value for each metric is the average of that metric over all test set reconstructions and over three different individual runs. According to the figures, our proposed DiVeQ and SF-DiVeQ consistently obtain higher SSIM, PSNR and lower LPIPS values than the other methods over different bitrates (except for the SSIM of NSVQ in Fig. 24). Furthermore, when increasing the VQ bitrate, the performance gap between our proposed methods and STE is escalated. On the other hand, similar to the result for ordinary VQ, the RT performs poorly for low RVQ bitrates, and its performance improves with the increase in bitrate.

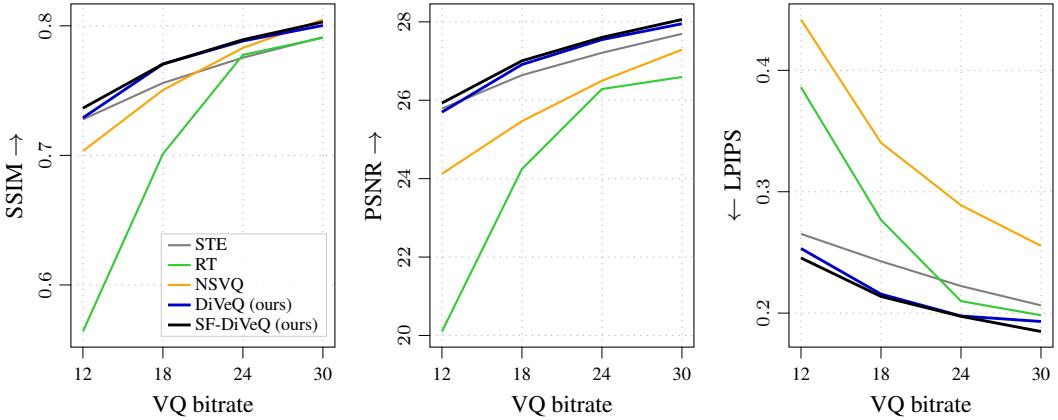


Figure 23: **Quantizing the latent space by Residual VQ: The proposed DiVeQ and SF-DiVeQ lead to consistent improvement in image reconstruction quality.** Quantitative comparison of reconstructed images from the [CELEBA-HQ](#) test set for different VQ optimization methods over different bitrates. Each curve is the average of the metric over all test set reconstructions and over three different individual runs.

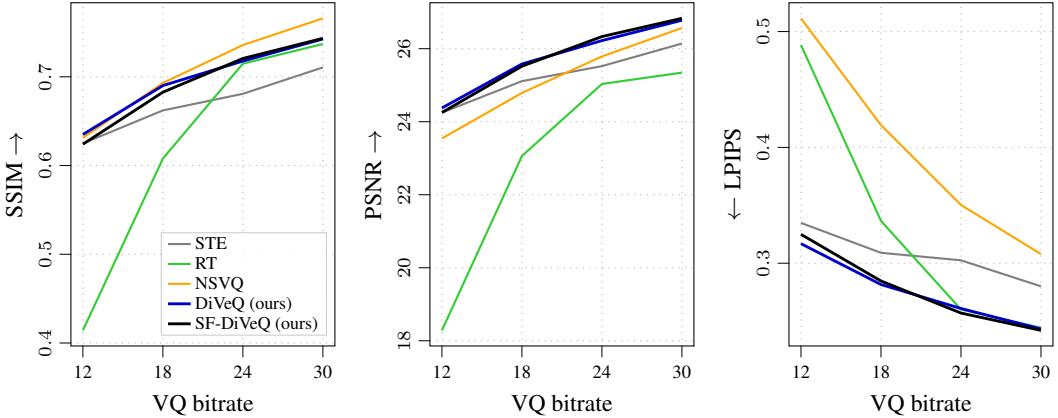


Figure 24: **Quantizing the latent space by Residual VQ: The proposed DiVeQ and SF-DiVeQ lead to consistent improvement in image reconstruction quality.** Quantitative comparison of reconstructed images from the [AFHQ](#) test set for different VQ optimization methods over different bitrates. Each curve is the average of the metric over all test set reconstructions and over three different individual runs.

C.8 TRAINING LOGS

In this section, we study the training logs when training the VQ-VAE for the image compression task on AFHQ data set. Fig. 25 shows the codebook usage, perplexity (Eq. (15)), and reconstruction and perceptual losses over the coarse of training for different VQ optimization methods when VQ bitrate is 11 (or codebook size is $K = 2048$). To make the plots look smoother, the curves are smoothed via Blackman windowing with a window size of 15 (except for the codebook usage plot). The codebook usage represents the percentage of codebook used in each training epoch. Apart from ST-GS method, all other VQ optimization techniques reach the full codebook usage (but it happens a bit late for the EMA approach). Note that the codebook usage is reported when codebook replacement (Sec. B.1) is actively applied during training for all VQ methods. Even though codebook replacement is applied, it cannot compensate for full codebook usage for ST-GS method.

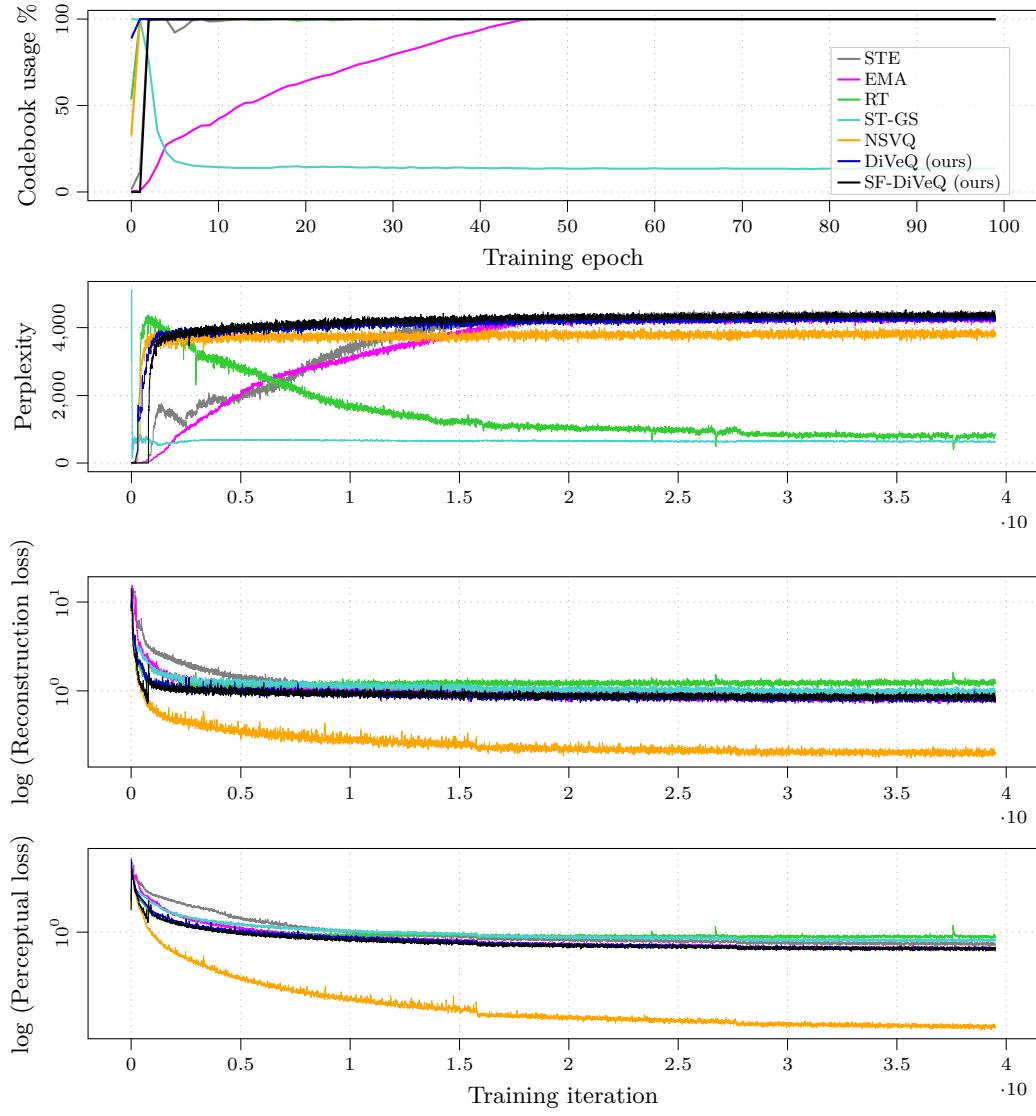


Figure 25: Training logs do not necessarily reflect whether VQ codebook optimization is being carried out appropriately. Comparison of the codebook usage, perplexity, reconstruction and perceptual losses for different VQ optimization methods, when training the VQ-VAE compression model with bitrate 11 on the AFHQ train set.

Perplexity refers to the number of unique codewords selected for quantizing a typical training batch, and it ranges from 1 to codebook size K . Eq. (15) explains how perplexity is computed. According to Fig. 25, the highest perplexities belong to STE, EMA, and our proposed DiVeQ and SF-DiVeQ, while NSVQ reaches a bit lower perplexity compared to these methods. However, for ST-GS and RT approaches, the perplexity is very high at the start of training and it suddenly drops for ST-GS, whereas it smoothly decays down for RT during training. Reconstruction loss is the MSE loss between the input \mathbf{x} and its reconstructed output \mathbf{x}_r (see Fig. 1 and Eq. (1)), and perceptual loss is the LPIPS between \mathbf{x} and \mathbf{x}_r . According to the losses, our proposed DiVeQ and SF-DiVeQ reaches lower loss values than STE, EMA, RT, and ST-GS while converging faster than STE, EMA, and ST-GS. In difference to all methods, NSVQ reaches the lowest loss values with a big margin, especially for the perceptual loss. To make the difference between all VQ methods more clear, we plot the loss values in logarithmic scale.

When training VQ in neural networks, computing the codebook usage once for the whole coarse of training is not enough to make sure everything is fine with the codebook optimization. Because the latent representation \mathcal{P}_z is dynamically changing, and as a result, one used codeword for the initial training iterations might become useless for the rest of the training by falling out of the latent representation. Codebook replacement helps to keep the codebook usage at its maximum during training by actively reinitializing unused codewords. However, even a 100% codebook usage during the whole training does not guarantee a proper learned codebook. Because, it is possible that all codewords are located inside the latent representation \mathcal{P}_z , but they are not necessarily well-fitted to \mathcal{P}_z . For instance, in [Fig. 25](#), RT and NSVQ reach 100% codebook usage, but yield poor qualitative ([Fig. 5](#)) and quantitative ([Fig. 6](#)) results, and they are both prone to misalignment of codebook and latent representations (see [Fig. 4](#)).

Similar to codebook usage and perplexity factors, reconstruction and perceptual losses cannot be a suitable metric to reflect whether a VQ optimization method performs better than the other. For instance, in [Fig. 25](#), NSVQ reaches a much higher perplexity than ST-GS and obtains the lowest loss values than the others, but it leads to the worst qualitative ([Fig. 5](#)) and quantitative ([Fig. 6](#)) results over different data sets compared to all other approaches. Therefore, these metrics are useful only to make sure the training process works fine.

C.9 MORE SAMPLES GENERATED BY VQGAN

Apart from the quantitative evaluation of VQGAN generations over different VQ optimization methods using the FID metric ([Tables 2](#) and [6](#)), it is important to compare the generations qualitatively as well. [Figs. 26 to 30](#) provide completely random generations from VQGAN for all five data sets when the VQ bitrate is 9 (or codebook size equals $K = 512$).



Figure 26: **Generation task.** Qualitative comparison of random generations for the CELEBA-HQ data set in a VQGAN generation task for different VQ optimization methods, when the VQ bitrate is 9 (or codebook size is $K = 512$).



Figure 27: **Generation task.** Qualitative comparison of random generations for the LSUN Church data set in a VQGAN generation task for different VQ optimization methods, when the VQ bitrate is 9 (or codebook size is $K = 512$).



Figure 28: Generation task. Qualitative comparison of random generations for the FFHQ data set in a VQGAN generation task for different VQ optimization methods, when the VQ bitrate is 9 (or codebook size is $K = 512$)

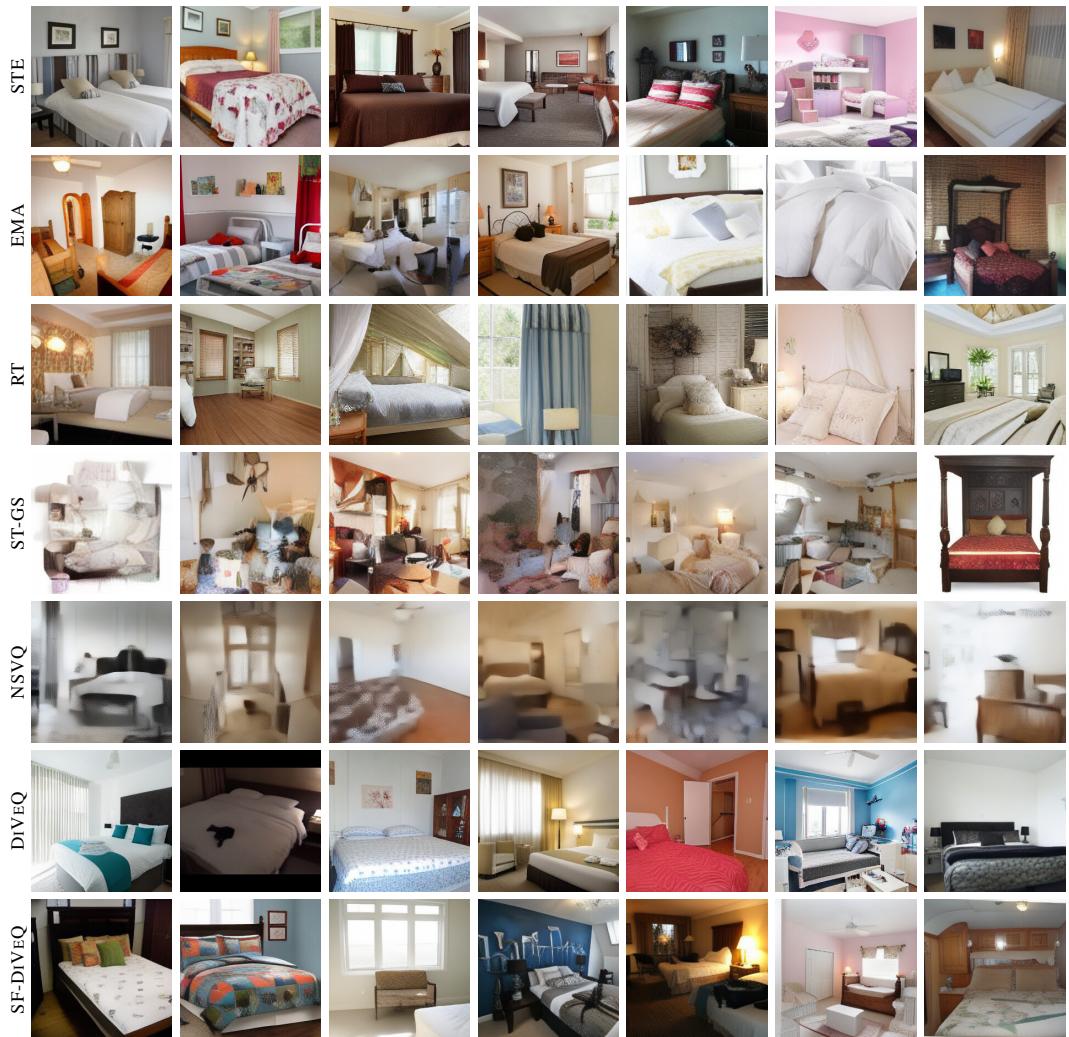


Figure 29: **Generation task.** Qualitative comparison of random generations for the LSUN Bedroom data set in a VQGAN generation task for different VQ optimization methods, when the VQ bitrate is 9 (or codebook size is $K = 512$).



Figure 30: Generation task. Qualitative comparison of random generations for the AFHQ data set in a VQGAN generation task for different VQ optimization methods, when the VQ bitrate is 9 (or codebook size is $K = 512$)