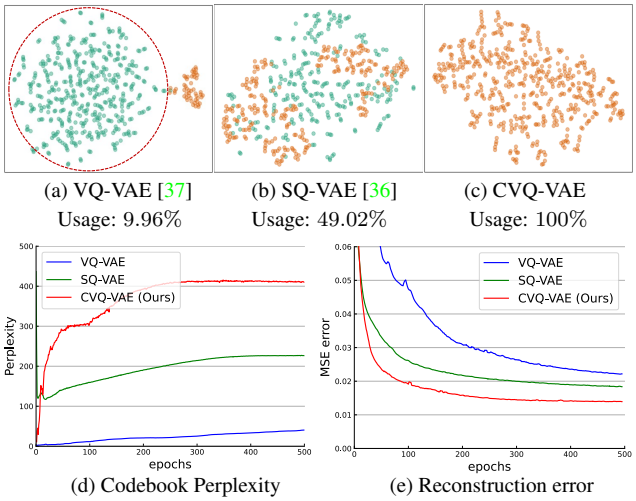
Online Clustered Codebook

Chuanxia Zheng Andrea Vedaldi
Visual Geometry Group, University of Oxford
{cxzheng, vedaldi}@robots.ox.ac.uk

Abstract

*Vector Quantisation (VQ) is experiencing a comeback in machine learning, where it is increasingly used in representation learning. However, optimizing the codevectors in existing VQ-VAE is not entirely trivial. A problem is codebook collapse, where only a small subset of codevectors receive gradients useful for their optimisation, whereas a majority of them simply "dies off" and is never updated or used. This limits the effectiveness of VQ for learning larger codebooks in complex computer vision tasks that require high-capacity representations. In this paper, we present a simple alternative method for online codebook learning, Clustering VQ-VAE (CVQ-VAE). Our approach selects encoded features as anchors to update the "dead" codevectors, while optimising the codebooks which are alive via the original loss. This strategy brings unused codevectors closer in distribution to the encoded features, increasing the likelihood of being chosen and optimized. We extensively validate the generalization capability of our quantiser on various datasets, tasks (*e.g. *reconstruction and generation), and architectures (*e.g. *VQ-VAE, VQGAN, LDM). CVQ-VAE can be easily integrated into the existing models with just a few lines of code.*



(a) VQ-VAE [37] Usage: 9.96% (b) SQ-VAE [36] Usage: 49.02% (c) CVQ-VAE Usage: 100%

(d) Codebook Perplexity (e) Reconstruction error

Figure 1: **Codebook usage and reconstruction error.** The setting is the same as VQ-VAE [37], except for the different quantisers. All models are trained and evaluated on the CIFAR10 [20] dataset. VQ-VAE has many "dead" vectors (green points) which are *not* used. CVQ-VAE updates these unoptimized vectors by using online sampled feature anchors, leading to a 100% usage of the codebook. CVQ-VAE achieves substantially higher codebook perplexity and better reconstruction results than with the fixed initialization.

1. Introduction

Vector Quantisation (VQ) [12] is a basic building block of many machine learning techniques. It is often used to help learning unsupervised representations for vision and language tasks, including data compression [1, 39, 36], recognition [26, 3, 44, 24, 23], and generation [37, 31, 11, 32, 47, 34, 33]. VQ quantises continuous feature vectors into a discrete space by embedding them to the closest vectors in a codebook of representatives or codevectors. Quantisation has been shown to simplify optimization problems by reducing a continuous search space to a discrete one.

Despite its success, VQ has some drawbacks when applied to deep networks [37]. One of them is that quantisation stops gradients from back-propagating to the codevectors. This has been linked to *codebook collapse* [36], which means that only a small subset of active codevectors are optimized alongside the learnable features, while the majority of them *are not used at all* (see the green "dead" points in Fig. 1(a)). As a result, many recent methods [11, 10, 44, 32, 6, 47] fail to utilise the full expressive power of a codebook due to the low codevector utilisation, especially when the codebook size is large. This significantly limits VQ's effectiveness.

To tackle this issue, we propose a new alternative quantiser called *Clustering VQ-VAE* (CVQ-VAE). We observe that classical clustering algorithms, such as refined initialization k-means [4] and k-means++ [2], use a dynamic cluster initialization approach. For example, k-means++ randomly selects a data point as the first cluster centre, and

then chooses the next new centre based on a weighted probability calculated from the distance to the previous centres. Analogously, CVQ-VAE *dynamically* initializes unoptimized codebooks by resampling them from the learned features (Fig. 2). This simple approach can avoid codebook collapse and significantly enhance the usage of larger codebooks by enabling optimization of all codevectors (achieving 100% codebook utilisation in Fig. 1(c)).

While CVQ-VAE is inspired by previous dynamic cluster initialization techniques [4, 2], its implementation in deep networks requires careful consideration. Unlike traditional clustering algorithms [25, 4, 14, 2] where source data points are fixed, in deep networks features and their corresponding codevectors are mutually and incrementally optimized. Thus, simply sampling codevectors from a single snapshot of features would not work well because any mini-batch used for learning *cannot* capture the true data distribution, as demonstrated in our offline version in Tab. 3. To fix this issue, we propose to *compute running averages* of the encoded features across different training mini-batches and use these to improve the dynamic reinitialization of the collapsed codevectors. This operation is similar to an online feature clustering method that calculates average features across different training iterations (Fig. 2). While this may seem a minor change, it leads to a very significant improvement in terms of performance (Fig. 1(e)).

As a result of these changes, CVQ-VAE significantly outperforms the previous models VQ-VAE [37] and SQ-VAE [36] on various datasets under the same setting, and with no other changes except for swapping in the new quantiser. Moreover, we conduct thorough ablation experiments on variants of the method to demonstrate the effectiveness of our design and analyse the importance of various design factors. Finally, we incorporate CVQ-VAE into large models (*e.g.* VQ-GAN [11] and LDM [32]) to further demonstrate its generality and potential in various applications.

2. Related Works

VQ-VAE [37] learns to quantise the continuous features into a discrete space using a restricted number of codebook vectors. By clustering features in the latent space, VQ-VAE can automatically learn a crucially compact representation and store the domain information in the decoder that does not require supervision. This discrete representation has been applied to various downstream tasks, including image generation [31, 44, 6, 22, 17], image-to-image translation [11, 30, 10, 32], text-to-image synthesis [30, 9, 29, 18], conditional video generation [28, 40, 42], image completion [11, 10, 46], recognition [26, 3, 44, 24, 23] and 3D reconstruction [27, 34, 33].

Among them, VQ-GAN [11], ViT-VQGAN [44], RQ-VAE [22], and MoVQ [46] aim to train a better discrete representation through deeper network architectures, additional loss functions, multichannel or higher resolution representations. However, none of them tackle the *codebook collapse* issue for the unoptimized "dead" point.

To address this issue, additional training heuristics are proposed in recent works. SQ-VAE [36] improves VQ-VAE with stochastic quantisation and a trainable posterior categorical distribution. VQ-WAE [38] builds upon SQ-VAE by directly encouraging the discrete representation to be a uniform distribution via a *Wasserstein* distance. The most related works are HVQ-VAE [39] and Jukebox [8] that use *codebook reset* to randomly reinitialize unused or low-used codebook entries. However, they only assign a single sampled anchor to each unoptimized codevector. In contrast, our CVQ-VAE considers the changing of features in deep networks and designs an online clustering algorithm by running average updates across the training mini-batch. Additionally, our work bridges codebook reset in Jukebox for music generation to the more general class of running average updates that are applicable to image compression and generation problems in computer vision.

3. Method

VQ is in the context of unsupervised representation learning. Our main goal is to learn a discrete codebook that efficiently utilizes *all codebook entries within it*. To achieve this, our quantisation method, as illustrated in Fig. 2, is conceptually similar to VQ-VAE [37], except that our codevectors are *dynamically initialized* rather than being sampled from a *fixed* uniform or Gaussian distribution. In the following sections, we provide a general overview of VQ (Sec. 3.1), followed by our proposed CVQ-VAE (Sec. 3.2).

3.1. Background: VQ-VAE

Given a high dimensional image $x \in \mathbb{R}^{H\times W\times c}$, VQ-VAE [37] learns to embed it with low dimensional codevectors $z_q \in \mathbb{R}^{h\times w\times n_q}$, where n_q is the dimensionality of the vectors in the codebook. Then, the feature tensor can be equivalently described as a compact representation with $h \times w$ indices corresponding to the codebook entries z_q. This is done via an autoencoder

$$\hat{x} = \mathcal{G}_\theta(z_q) = \mathcal{G}_\theta(\mathbf{q}(\hat{z})) = \mathcal{G}_\theta(\mathbf{q}(\mathcal{E}_\phi(x))). \quad (1)$$

Here $\mathcal{E}_\phi$ and $\mathcal{G}_\theta$ refer to the encoder and decoder, respectively. The encoder embeds images into the continuous latent space, while the decoder inversely maps the latent vectors back to the original image. $\mathbf{q}(\cdot)$ is a quantisation operation that maps the continuous encoded observations $\hat{z}$ into the discrete space by looking up the closest codebook entry e_k for each grid feature $\hat{z}_i$ using the following equation:

$$z_{q_i} = \mathbf{q}(\hat{z}_i) = e_k, \quad \text{where} \quad k = \underset{e_k\in\mathcal{Z}}{\operatorname{argmin}} \|\hat{z}_i - e_k\|, \quad (2)$$

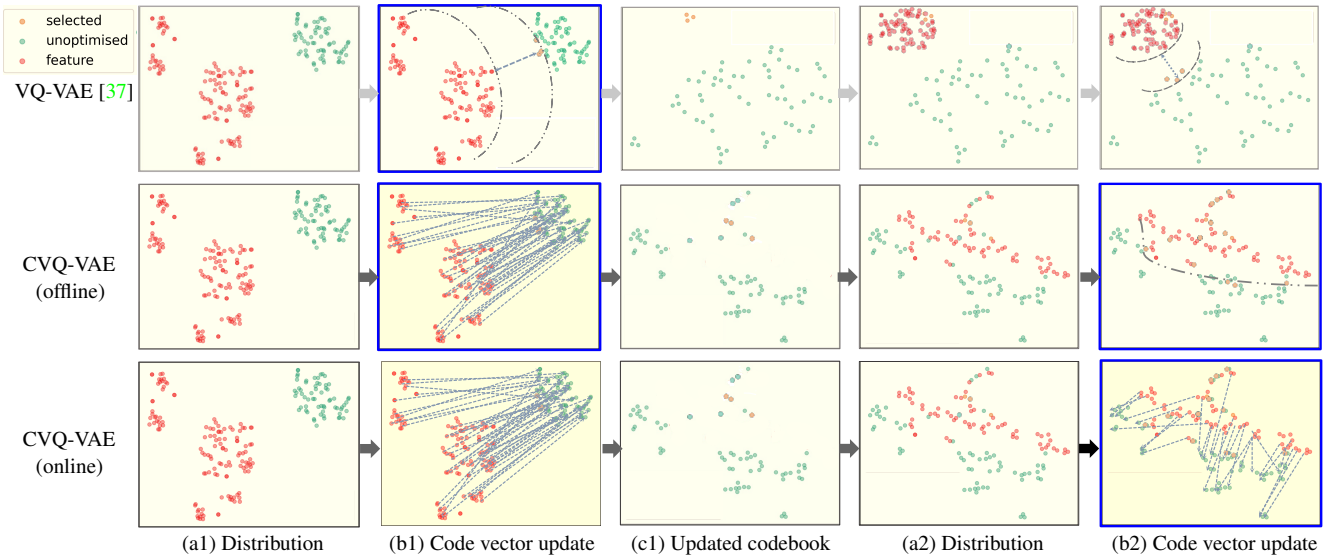


Figure 2: **Codebook optimization**. The Red points indicate the encoded features, while the Green and Peach points denote the unoptimized and active vectors in the codebook, respectively. 1) In VQ-VAE [37] (row 1), only the active "lucky" seeds (in Peach) are optimized alongside the encoded features (in Red) during training. The other "dead" vectors (in Green) are *not* given attention and remain fixed. 2) In our CVQ-VAE (offline) (row 2), we reinitialize the codevectors based on the anchors sampled from the encoded features (in Red), encouraging the "dead" ones to be closer to the features in distribution. 3) To address the difficulty of covering all samples by single sampling in mini-batch learning, we further propose an online learning variant (row 3), where the anchor is obtained by calculating the moving average of the encoded features in different batches. We highlighted the difference between various methods in the blue thickened border.

where $\mathcal{Z} = \{e_k\}_{k=1}^{K}$ is the codebook that consists of K entries $e_k \in \mathbb{R}^{n_q}$ with dimensionality n_q. During training, the encoder $\mathcal{E}_\phi$, decoder $\mathcal{G}_\theta$ and codebook $\mathcal{Z}$ are jointly optimized by minimizing the following objective:

$$\mathcal{L} = \|x - \hat{x}\|_2^2 + \|\text{sg}[\mathcal{E}_\psi(x)] - z_q\|_2^2 + \beta\|\mathcal{E}_\psi(x) - \text{sg}[z_q]\|_2^2, \tag{3}$$

where sg denotes a stop-gradient operator, and β is the hyperparameter for the last term *commitment loss*. The first term is known as *reconstruction loss*, which measures the difference between the observed x and the reconstructed $\hat{x}$. The second term is the *codebook loss*, which encourages the codevectors to be close to the encoded features. In practice, the codebook $\mathcal{Z}$ is optimized using either the *codebook loss* [37] or using an exponential moving average (EMA) [31]. However, these methods work only for the active codevectors, *leaving the "dead" ones unoptimized*.

3.2. Clustering VQ-VAE (CVQ-VAE)

The choice of initial points is a crucial aspect of unsupervised codebook learning. Classical clustering methods like refined k-means [4] and k-means++ [2] are *dynamically-initialized*, which means that each new clustering centre is initialized based on previously calculated distance or points. This leads to a more robust and effective clustering result, as reported in comparative studies [5].

Analogously, we build a *dynamically-initialized* vector quantized codebook in deep networks. However, unlike traditional clustering settings, the data points, *i.e.* the encoded features $\hat{z}$ in the deep network, are also updated during training instead of being fixed. Therefore, a dynamical initialization strategy should take into account the changing feature representations during training.

Running average updates. To build the online initialization for a codebook, we start by accumulatively counting the average usage of codevectors in each training mini-batch:

$$N_k^{(t)} = N_k^{(t-1)} \cdot \gamma + \frac{n_k^{(t)}}{Bhw} \cdot (1 - \gamma), \tag{4}$$

where $n_k^{(t)}$ is the number of encoded features in a training mini-batch that will be quantised to the closest codebook entry e_k, and Bhw denotes the number of features on Batch, height, and width. γ is a decay hyperparameter with a value in $(0, 1)$ (default $\gamma = 0.99$). $N_k^{(0)}$ is initially set as zero.

We then select a subset $\bar{\mathcal{Z}}$ with K vectors from the encoded features $\hat{z}$, which we denote as **anchors**. Instead of directly using the anchors to reinitialize the unoptimized codevectors, we expect that *codevectors that are less-used or unused should be modified more than frequently used ones*. To achieve this goal, we compute a decay value $a_k^{(t)}$ for each entry e_k using the accumulative average usage $N_k^{(t)}$

and reinitialize the features as follows:

$$\alpha_k^{(t)} = \exp^{-N_k^{(t)} K \frac{10}{1-\gamma} - \epsilon}, \tag{5}$$

$$e_k^{(t)} = e_k^{(t-1)} \cdot (1 - \alpha_k^{(t)}) + \hat{z}_k^{(t)} \cdot \alpha_k^{(t)}, \tag{6}$$

where ϵ is a small constant to ensure the entries are assigned with the average values of features along different mini-batches, and $\hat{z}_k^{(t)} \in \bar{\mathcal{Z}}^{K \times z_q}$ is the sampled anchor.

This running average operation differs from the exponential moving average (EMA) used in VQ-VAE [31]. Our equation is applied to reinitialize unused or low-used codevectors, instead of updating the active ones. Furthermore, our decay parameter in Eq. (5) is computed based on the average usage, which is *not a pre-defined hyperparameter*.

Choice of the anchors. Next, we describe several versions of the anchor sampling methods. Interestingly, experimental results (Tab. 4c) show that our online version is *not* sensitive to these choices. However, the different anchor sampling methods have a direct impact on the *offline* version, suggesting that our running average updates behaviour is the primary reason for the observed improvements.

- **Random.** Following the codebook reset [8, 39], a natural choice of anchors is randomly sampled from the encoded features.
- **Unique.** To avoid repeated anchors, a random permutation of integers within the number of features (Bhw) is performed. Then, we select the first K features.
- **Closest.** A simple way is inversely looking up the closest features of each entry, *i.e.* $i = \underset{\hat{z}_i \in \mathcal{E}_\phi(x)}{\operatorname{argmin}} \|\hat{z}_i - e_k\|$.
- **Probabilistic random.** We can also sample anchors based on the distance $D_{i,k}$ between the codevectors and the encoded features. In this paper, we consider the probability $p = \frac{\exp(-D_{i,k})}{\sum_{i=1}^{Bhw} \exp(-D_{i,k})}$.

Contrastive loss. We further introduce a contrastive loss $-\log \frac{e^{sim(e_k, \hat{z}_i^+)/\tau}}{\sum_{i=1}^{N} e^{sim(e_k, \hat{z}_i^-)/\tau}}$ to encourage sparsity in the codebook. In particular, for each codevector e_k, we directly select the closest feature $\hat{z}_i^+$ as the positive pair and sample other farther features $\hat{z}_i^-$ as negative pairs using the $D_{i,k}$.

Relation to prior work. To mitigate the codebook collapse issue, several methods have been proposed, like normalized codevectors in ViT-VQGAN [44]. However, these methods only optimize the *active* entries, rather than the entire codebook. Recently, SQ-VAE [36], SeQ-GAN [13], and VQ-WAE [38] assume that the codebook follows a fixed distribution. Although these methods achieve high perplexity, the reconstruction quality is *not* always improved (Tab. 4). The most relevant work to ours is codebook reset, which randomly reinitializes the unused or low-used codevectors to high-usage ones [39] or encoder outputs [8]. However, these methods rely only on a temporary single value for initialization and miss the opportunity of exploiting online clustering across different training steps.

4. Experiments: Image Quantisation

4.1. Experimental Details

Implementation. CVQ-VAE can be easily implemented by a few lines of code in Pytorch, where the gradient for the selected codevectors is preserved. The code is available at https://github.com/lyndonzheng/CVQ-VAE.

Our implementation is built upon existing network architectures. We set all hype-parameters following the original code, except that we replace these quantisers with our online clustering codebook. In particular, we first demonstrate our assumption on small datasets with the officially released VQ-VAE [37] implementation [1,2]. Then, we verify the generality of our quantiser on large datasets using the officially released VQ-GAN [11] architecture [3].

Datasets. We evaluated the proposed quantiser on various datasets, including MNIST [21], CIFAR10 [20], Fashion MNIST [41], and the higher-resolution FFHQ [19] and the large ImageNet [7].

Metrics. Following existing works [11, 47, 13], we evaluated the image quality between reconstructed and original images on different scales, including patch-level structure similarity index (SSIM), feature-level Learned Perceptual Image Patch Similarity (LPIPS) [45], and dataset-level Fréchet Inception Distance (FID) [15]. We also report the perplexity score for the codebook ablation study as in SQ-VAE [36] and VQ-WAE [38]. It is defined as $e^{-\sum_{k=1}^{K} p_{e_k} \log p_{e_k}}$, where $p_{e_k} = \frac{n_k}{\sum_{i=i}^{K} n_k}$, and n_k is the number of encoded features associated with codevector e_k.

4.2. Main Results

Quantitative Results: We first evaluated our CVQ-VAE and various quantisers, including VQ-VAE [37]$_{\text{NeurIPS'2017}}$, HVQ-VAE [39]$_{\text{NeurIPS'2020}}$, and SQ-VAE [36]$_{\text{ICML'2022}}$, under the identical experimental settings in Tab. 1. All instantiations of our model outperform the baseline variants of previous state-of-the-art models. Although the latest SQ-VAE [36] optimizes all code entries by explicitly enforcing the codebook to be a defined distribution, this assumption may not hold for all datasets. For instance, code entries that respond to the background elements like sky and ground should take more count than code entries that represent specific objects, such as vehicle wheels. In contrast, our quantiser only encourages all code entries to be optimized, leaving the association to be automatically learned.

[1] https://github.com/deepmind/sonnet/blob/v2/sonnet/src/nets/vqvae.py

[2] https://github.com/deepmind/sonnet/blob/v1/sonnet/examples/vqvae_example.ipynb

[3] https://github.com/CompVis/taming-transformers

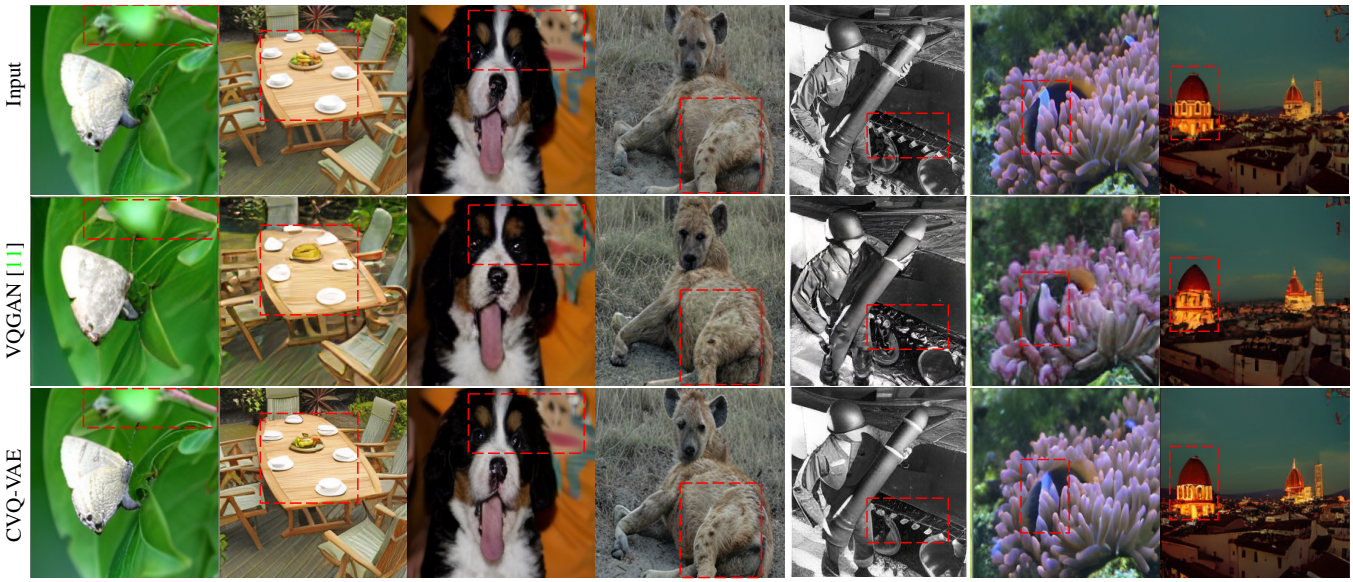


Figure 3: **Reconstructions from different models.** The two models are trained under the same settings, except for the different quantisers. Compared with the state-of-the-art baseline VQGAN [11], the proposed model significantly improves the reconstruction quality (highlight in red box) under the same compression ratio (768×, with 16× downsampling).

Method	**Dataset**	SSIM ↑	LPIPS ↓	rFID ↓
VQ-VAE [37]	MNIST	0.9777	0.0282	3.43
HVQ-VAE [39]		0.9790	0.0270	3.17
SQ-VAE [36]		0.9819	0.0256	3.05
CVQ-VAE		**0.9833**	**0.0222**	**1.80**
VQ-VAE [37]	CIFAR10	0.8595	0.2504	39.67
HVQ-VAE [39]		0.8553	0.2553	41.08
SQ-VAE [36]		0.8779	0.2333	37.92
CVQ-VAE		**0.8978**	**0.1883**	**24.73**

Table 1: **Reconstruction results** on the validation sets of MNIST (10,000 images) and CIFAR10 (10,000 images). All models are trained with the same experimental settings, except for the different quantisers.

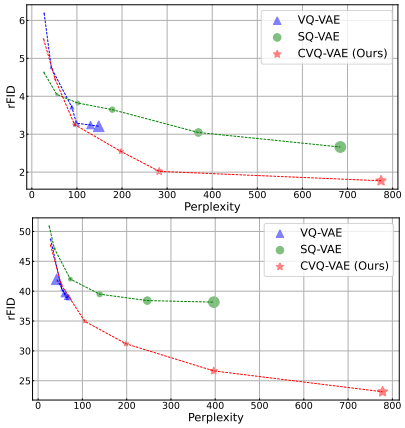
Method	**Dataset**	$\mathcal{S}\downarrow$	$\mathcal{K}\downarrow$	Usage ↑	rFID ↓
VQGAN [11]	FFHQ	16^2	1024	42%	4.42
ViT-VQGAN [44]		32^2	8192	—	3.13
RQ-VAE [22]		$16^2\times4$	2048	—	3.88
MoVQ [47]		$16^2\times4$	1024	56%	2.26*
SeQ-GAN [13]		16^2	1024	100%	3.12
CVQ-VAE (ours)		16^2	1024	100%	2.80
CVQ-VAE (ours)		$16^2\times4$	1024	100%	**2.03**
VQGAN [11]	ImageNet	16^2	1024	44%	7.94
ViT-VQGAN [44]		32^2	8192	96%	1.28
RQ-VAE [22]		$8^2\times16$	16384	—	1.83
MoVQ [47]		$16^2\times4$	1024	63%	**1.12**
SeQ-GAN [13]		16^2	1024	100%	1.99
CVQ-VAE (ours)		16^2	1024	100%	1.57
CVQ-VAE (ours)		$16^2\times4$	1024	100%	1.20*

Table 2: **Reconstruction results** on validation sets of ImageNet (50,000 images) and FFHQ (10,000 images). $\mathcal{S}$ denotes the latent size of encoded features, and $\mathcal{K}$ is the number of codevectors in the codebook. Usage indicates how many entries in a codebook are used during the quantisation on the validation set. More evaluation metrics are reported in Appendix Table B.2.

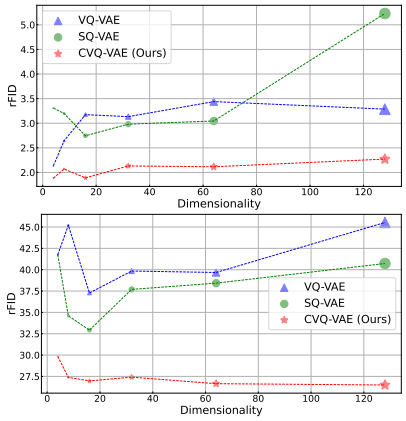
Then, we compared our CVQ-VAE with the state-of-the-art methods, including VQGAN [11]$_{\text{CVPR'2021}}$, ViT-VQGAN [44]$_{\text{ICLR'2022}}$, RQ-VAE [22]$_{\text{CVPR'2022}}$, and MoVQ [45]$_{\text{NeurIPS'2022}}$ for the task of reconstruction. Table 2 shows quantitative results on two large datasets. Under the same compression ratio (768×, *i.e.* 256×256×3→16×16), our model significantly outperforms the state-of-the-art models, including the baseline VQGAN [11] and the concurrent SeQ-GAN [13]. Interestingly, on the FFHQ dataset, our model even outperforms ViT-VQGAN [44] and RQ-VAE [22], which utilize 4× tokens for the representation. This suggests that the high usage of codevectors is significant for maintaining information during data compression. Additionally, we also run 4× tokens experiments, as in MoVQ [47]. The CVQ-VAE further achieves a relative 10.1% improvement. Although our 4× version shows a slightly lower rFID score than the MoVQ [47] on ImageNet dataset (1.12 *vs.* 1.20), we achieve better performance on other metrics (as shown in Appendix Tab. B.2).

Method	MNIST (28×28)			CFAIR10 (32×32)			Fashion MNIST (28×28)		
	SSIM ↑	LPIPS ↓	rFID ↓	SSIM ↑	LPIPS ↓	rFID ↓	SSIM ↑	LPIPS ↓	rFID ↓
𝔸 Baseline VQ-VAE [37]$_{\text{NeurIPS'2017}}$	0.9777	0.0282	3.43	0.8595	0.2504	39.67	0.9140	0.0801	12.73
𝔹 + Cosine distance	0.9791	0.0266	3.06	0.8709	0.2303	35.14	0.9160	0.0764	11.40
ℂ + Anchor initialization (offline)	0.9810	0.0253	2.78	0.8829	0.2132	31.10	0.9145	0.0773	11.92
𝔻 + Anchor initialization (online)	0.9823	0.0236	2.23	**0.8991**	0.1897	26.62	**0.9254**	**0.0683**	9.27
𝔼 + Contrastive loss	**0.9833**	**0.0222**	**1.80**	0.8978	**0.1883**	**24.73**	0.9233	0.0693	**8.85**

Table 3: **Results on various settings.** We report patch-level SSIM, feature-level LPIPS, and dataset-level FID. All evaluation metrics are reported in Appendix Table B.3.



(a) **Codebook size**. The blobs' size is proportional to the number of codebook vectors {32, 64, 128, 256, 512, 1024}. The larger size naturally leads to better results in our CVQ-VAE.



(b) **Codebook dimensionality**. The blob's size refers to the dimensionality of codebook vectors {4,8,16,32,64,128}. The higher dimensionality does not ensure a better representation.

Method	Dataset	rFID↓ (offline)	rFID↓ (online)
random	MNIST	3.20	2.27
unique		2.84	2.24
probability		2.78	**2.23**
closest		**2.51**	2.59
random	CIFAR10	34.49	26.04
unique		36.99	26.02
probability		**31.10**	26.62
closest		32.31	**25.99**

(c) **Anchor sampling methods.** The choice of anchor sampling method has a significant impact on offline (one-time) feature initialization, while the online clustered method is robust for various samplings.

Methods	MNIST (28×28)			CIFAR10 (32×32)			FFHQ (256×256)		
	SSIM ↑	LPIPS ↓	rFID↓	SSIM ↑	LPIPS ↓	rFID↓	SSIM ↑	LPIPS ↓	rFID↓
near codevectors [39]	0.9790	0.0270	3.17	0.8553	0.2553	41.08	0.7282	0.1085	4.31
hard encoded features [8]	0.9814	0.0243	2.25	0.8988	0.1978	29.16	0.7646	0.0870	3.91
running average (ours)	**0.9823**	**0.0236**	**2.23**	**0.8991**	**0.1897**	**26.62**	**0.8193**	**0.0603**	**2.94**

(d) **Codebook reinitialization methods.** In previous works [39, 8], each code entry is associated only with a single feature.

Table 4: **Ablations** for CVQ-VAE on image quantisation. We mainly train on MNIST and CIFAR10 training set, and evaluate on the validation set unless otherwise noted.

Qualitative Results: The qualitative comparisons are presented in Fig. 3. Our model achieves superior results even under challenging conditions. Compared to the baseline model VQGAN [11], our CVQ-VAE provides higher-quality reconstructed images that retain much more details. In particular, VQGAN struggles with reconstructing abundant scene elements, as evidenced by the artifacts on the bowls. In contrast, our CVQ-VAE shows no such artifacts. These fine-grained details are crucial for downstream generation-related tasks, such as generation, completion, and translation [47].

4.3. Ablation Experiments

We ran a number of ablations to analyse the effects of core factors in codebook learning. Results are reported in Tabs. 3, 4, B.3 and B.4.

Core Factors. We evaluated core components in our redesigned online clustering quantiser in Tab. 3, which shows that the new quantiser considerably enhances the reconstruction quality. We started by implementing the baseline configuration (𝔸) from VQ-VAE [37]. Next, we explored different distance metrics, which are used to look up the closest entry for each encoded feature. We found that us-

ing cosine similarity (𝔹) improved performance on some datasets, which is consistent with the findings in previous works such as ViT-VQGAN [44]. In configuration (ℂ), we reinitialized the unoptimized code entries with the selected anchors, but only in the first training batch, which we refer to as the *offline* version. This improved the usage of the codebook, resulting in slightly better gains. Significantly, when we applied the proposed *running average updates* across different training mini-batches in configuration (𝔻), the performance on all metrics in various datasets improved substantially. This suggests that our proposed online clustering is significant for handling the changing encoded feature representation in deep networks. Finally, we naturally introduced contrastive loss to each entry based on its similarity to features (𝔼), which further improved the results.

Codebook Size. VQ embeds the continuous features into a discrete space with a finite size, *i.e*. K codebook entries. The codebook size has significant effects on traditional clustering. In Tab. 4a, we showed the performances of various quantisers with different codebook sizes. Our CVQ-VAE benefits greatly from a larger number of codebook entries, while SQ-VAE [36] shows smaller improvements. It is worth noting that *not* all quantizers automatically benefit from a larger codebook size, such as VQ-VAE's performance on the CIFAR10 dataset shown in Tab. 4a (bottom).

Perplexity *vs.* rFID. Recent concurrent studies [36, 13, 38] have explicitly promoted a large perplexity by optimizing a perplexity-related loss. However, as illustrated in Tab. 4a, a larger perplexity does *not* always guarantee a lower rFID. This suggests that a uniform distribution of perplexity, represented by the highest score, may not be the optimal solution for the codebook.

Codebook Dimensionality. Table 4b presents the results on various codebook dimensionalities. Interestingly, the performance of the quantizers *does not exhibit a straightforward positive correlation* with the number of codebook dimensionality. In fact, some smaller codebook dimensionalities yield better results than larger ones, indicating that the choice of codebook dimensionality should be carefully considered depending on the specific application and dataset. Based on this observation, a low-dimensional codebook can be employed to represent images and used in downstream tasks, as demonstrated in the latent diffusion model (LDM) [32]. The relevant downstream applications on generation can be found in Sec. 5.

Anchor Sampling Methods. An evaluation of various *anchor sampling methods* is reported in Tab. 4c. The results indicate that the *offline* version with only one reinitialization is highly sensitive to the anchor sampling methods. Interestingly, the random, unique, closest, and probabilistic random versions perform similarly for *online* version, up to some random variations (rFID from 2.23 to 2.59 on MNIST, and from 25.99 to 26.62 on CIFAR10). As discussed in Sec. 3.2, different anchor sampling methods have significant effects on traditional clustering [4, 14]. However, our experiments demonstrate that the codebook reinitialization needs to consider the fact that *the encoded features change along with the deep network is trained*. The results highlight the effectiveness of our *online* version with the running average updates, which is insensitive to the different instantiations.

Methods	FID↓	
	Churches	Bedrooms
StyleGAN [19]	4.21	2.35
DDPM [16]	7.89	4.90
ImageBART [10]	7.32	5.51
Projected-GAN [35]	**1.59**	**1.52**
LDM [32]-8*	4.02	-
LDM [32]-4	-	2.95
LDM [32]-8 (reproduced)	4.15	3.57
CVQ-VAE-LDM [32]-8	3.86	3.02

Table 5: **Quantitative comparisons on unconditional image generation.** The better quantiser can improve the generation quality without modifying the training settings in the second stage. *: trained in KL-regularized latent space, instead of the VQ discrete space.

Reinitialization Methods. Some latest works [39, 8] also consider updating the unoptimized codevectors, called *codebook reset*. In Tab. 4d, we compare these methods with VQ-VAE's architecture [37] under the same experimental setting, except for the different quantisers. As discussed in Sec. 3.2, HVQ-VAE [39] resets the low usage codevectors using the high usage ones, which learns a narrow space codebook, resulting in limited improvement. The hard encoded features presented in [8] achieve better results (3.17→2.25, 41.08→29.16, and 4.31→3.91) than the HVQ-VAE [39] by adding noise signal to ensure the independent anchors for each codebook entry. In contrast, our CVQ-VAE calculates the running average updates, resulting in a significant improvement. This further suggests that the online clustering centre along with the different training mini-batches is crucial for proper codebook reinitialization.

5. Experiments: Applications

Except for data compression, our CVQ-VAE can also be easily applied to downstream tasks, such as generation and completion. Following existing works [11, 32, 47], we conduct a simple experiment to verify the effectiveness of the proposed quantisers. Although this simple yet effective quantiser can be applied to more applications, it is beyond the main scope of this paper.

(a) LSUN-Bedrooms (b) LSUN-Churches (c) ImageNet

Figure 4: **Unconditional** image generation on LSUN [43], and ***class*-conditional** image generation on Imagenet [7]. Following the baseline LDM [32], our results are generated on 256×256 resolution. Our training parameters are the same as in LDM, except for the different quantisers and 8× downsampling for the latent representations.

Model	FFHQ		ImageNet	
	Steps	FID↓	Steps	FID↓
RQVAE [22]$_{\text{CVPR'2022}}$	256	10.38	1024	7.55
MoVQ [44]$_{\text{NeurIPS'2022}}$	1024	8.52	1024	7.13
SQ-VAE [33]$_{\text{ICML'2022}}$	200	5.17	250	9.31
LDM-4 [31]$_{\text{CVPR'2022}}$	200	4.98	250	10.56
CVQ-VAE (ours)	200	**4.46**	250	**6.87**

Table 6: Quantitative results for unconditional generation on FFHQ and *class*-conditional generation on ImageNet.

Implementation Details. We made minor modifications to the baseline LDM [32] system when adapting it with our quantiser for the downstream tasks. We first replace the original quantiser from VQGAN [11] with our proposed CVQ-VAE's quantiser. Then, we trained the models on LSUN [43] and ImageNet [7] for generation (8× downsampling). Following the setting in LDM [32], we set the sampling step as 200 during the inference.

5.1. Unconditional Generation

Tables 5 and 6 compares our proposed CVQ-VAE to the state-of-the-art methods on LSUN and ImageNet datasets for unconditional and *class*-conditional image generation. The results show that our model consistently improves the generated image quality under the sample compression ratio, as in the reconstruction. This confirms the advantages of using a better codebook for downstream tasks. Our CVQ-VAE also outperforms the LDM-8* that is trained with KL-regularized latent space, indicating that exploring a better discrete codebook is worth pursuing for unsupervised representation learning. Our CVQ-VAE also achieves comparable results to LDM-4 (3.02 *vs.* 2.95), whereas the LDM-4 uses a 4× higher resolution representation, requiring more computational costs.

Example results are presented in Fig. 4. As we can see, even with 8× downsampling, the proposed CVQ-VAE is still able to generate reasonable structures for these complicated scenes with various instances. Although there are artifacts on windows in the two scenarios, the other high-frequency details are realistic, such as the sheet on the bed.

6. Conclusion and Limitation

We have introduced CVQ-VAE, a novel codebook reinitialization method that tackles the *codebook collapse* issue by assigning the online clustered anchors to unoptimized code entries. Our proposed quantiser is a simple yet effective solution that can be integrated into many existing architectures for representation learning. Experimental results show that our CVQ-VAE significantly outperforms the state-of-the-art VQ models on image modeling, yet without increasing computational cost and latent size. We hope this new plug-and-play quantiser will become an important component of future vector methods that use VQ in their learned architecture.

Ethics. We use the MNIST, Fashion-MNIST, CIFAR10, LSUN, and ImageNet datasets in a manner compatible with their terms. While some of these images contain personal information (*e.g.*, faces) collected without consent, algorithms in this research do not extract biometric information. For further details on ethics, data protection, and copyright please see https://www.robots.ox.ac.uk/~vedaldi/research/union/ethics.html.

Acknowledgements. This research is supported by ERC-CoG UNION 101001212.

References

[1] Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc V Gool. Soft-to-hard vector quantization for end-to-end learning compressible representations. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS)*, volume 30, 2017. 1

[2] D ARTHUR. k-means++: the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, New Orleans, Louisiana, 2007*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007. 1, 2, 3

[3] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022. 1, 2

[4] Paul S Bradley and Usama M Fayyad. Refining initial points for k-means clustering. In *ICML*, volume 98, pages 91–99. Citeseer, 1998. 1, 2, 3, 7

[5] M Emre Celebi, Hassan A Kingravi, and Patricio A Vela. A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert systems with applications*, 40(1):200–210, 2013. 3

[6] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. Maskgit: Masked generative image transformer. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022. 1, 2

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition (CVPR)*, pages 248–255. Ieee, 2009. 4, 8

[8] 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. 2, 4, 6, 7

[9] Ming Ding, Zhuoyi Yang, Wenyi Hong, Wendi Zheng, Chang Zhou, Da Yin, Junyang Lin, Xu Zou, Zhou Shao, Hongxia Yang, et al. Cogview: Mastering text-to-image generation via transformers. *Advances in Neural Information Processing Systems (NeurIPS)*, 34, 2021. 2

[10] Patrick Esser, Robin Rombach, Andreas Blattmann, and Bjorn Ommer. Imagebart: Bidirectional context with multinomial diffusion for autoregressive image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021. 1, 2, 7

[11] 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)*, pages 12873–12883, 2021. 1, 2, 4, 5, 6, 7, 8, 11, 12

[12] R. Gray. Vector quantization. *IEEE ASSP Magazine*, 1(2):4–29, 1984. 1

[13] Yuchao Gu, Xintao Wang, Yixiao Ge, Ying Shan, Xiaohu Qie, and Mike Zheng Shou. Rethinking the objectives of vector-quantized tokenizers for image synthesis. *arXiv preprint arXiv:2212.03185*, 2022. 4, 5, 7, 12

[14] Greg Hamerly and Charles Elkan. Alternatives to the k-means algorithm that find better clusterings. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 600–607, 2002. 2, 7

[15] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS)*, pages 6626–6637, 2017. 4

[16] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:6840–6851, 2020. 7

[17] Minghui Hu, Yujie Wang, Tat-Jen Cham, Jianfei Yang, and Ponnuthurai N Suganthan. Global context with discrete diffusion in vector quantised modelling for image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11502–11511, 2022. 2

[18] Minghui Hu, Chuanxia Zheng, Heliang Zheng, Tat-Jen Cham, Chaoyue Wang, Zuopeng Yang, Dacheng Tao, and Ponnuthurai N Suganthan. Unified discrete diffusion for simultaneous vision-language generation. *arXiv preprint arXiv:2211.14842*, 2022. 2

[19] 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*, pages 4401–4410, 2019. 4, 7

[20] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 1, 4

[21] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 4

[22] 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)*, pages 11523–11532, 2022. 2, 5, 12

[23] Wei Li, Can Gao, Guocheng Niu, Xinyan Xiao, Hao Liu, Jiachen Liu, Hua Wu, and Haifeng Wang. Unimo-2: End-to-end unified vision-language grounded learning. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3187–3201, 2022. 1, 2

[24] Alex Liu, SouYoung Jin, Cheng-I Lai, Andrew Rouditchenko, Aude Oliva, and James Glass. Cross-modal discrete representation learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3013–3035, 2022. 1, 2

[25] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982. 2

[26] Chengzhi Mao, Lu Jiang, Mostafa Dehghani, Carl Vondrick, Rahul Sukthankar, and Irfan Essa. Discrete representations strengthen vision transformer robustness. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022. 1, 2

[27] Paritosh Mittal, Yen-Chi Cheng, Maneesh Singh, and Shubham Tulsiani. Autosdf: Shape priors for 3d completion, reconstruction and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition(CVPR)*, pages 306–315, 2022. 2

[28] Ruslan Rakhimov, Denis Volkhonskiy, Alexey Artemov, Denis Zorin, and Evgeny Burnaev. Latent video transformer. In *16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, VISIGRAPP 2021*, pages 101–112. SciTePress, 2021. 2

[29] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022. 2

[30] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning (ICML)*, pages 8821–8831. PMLR, 2021. 2

[31] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019. 1, 2, 3, 4

[32] 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)*, pages 10684–10695, 2022. 1, 2, 7, 8, 11

[33] Aditya Sanghi, Pradeep Kumar Jayaraman, Arianna Rampini, Joseph Lambourne, Hooman Shayani, Evan Atherton, and Saeid Asgari Taghanaki. Sketch-a-shape: Zero-shot sketch-to-3d shape generation. *arXiv preprint arXiv:2307.03869*, 2023. 1, 2

[34] Kyle Sargent, Jing Yu Koh, Han Zhang, Huiwen Chang, Charles Herrmann, Pratul Srinivasan, Jiajun Wu, and Deqing Sun. Vq3d: Learning a 3d-aware generative model on imagenet. *arXiv preprint arXiv:2302.06833*, 2023. 1, 2

[35] Axel Sauer, Kashyap Chitta, Jens Müller, and Andreas Geiger. Projected gans converge faster. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:17480–17492, 2021. 7

[36] 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. In *International Conference on Machine Learning (ICML)*, pages 20987–21012. PMLR, 2022. 1, 2, 4, 5, 7, 11

[37] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS)*, 2017. 1, 2, 3, 4, 5, 6, 7, 11, 12

[38] Tung-Long Vuong, Trung Le, He Zhao, Chuanxia Zheng, Mehrtash Harandi, Jianfei Cai, and Dinh Phung. Vector quantized wasserstein auto-encoder. *arXiv preprint arXiv:2302.05917*, 2023. 2, 4, 7

[39] Will Williams, Sam Ringer, Tom Ash, David MacLeod, Jamie Dougherty, and John Hughes. Hierarchical quantized autoencoders. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020. 1, 2, 4, 5, 6, 7, 11

[40] Chenfei Wu, Jian Liang, Lei Ji, Fan Yang, Yuejian Fang, Daxin Jiang, and Nan Duan. Nüwa: Visual synthesis pre-training for neural visual world creation. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XVI*, pages 720–736. Springer, 2022. 2

[41] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv*, 2017. 4

[42] Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. Videogpt: Video generation using vq-vae and transformers. *arXiv preprint arXiv:2104.10157*, 2021. 2

[43] 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*, 2015. 8

[44] Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. Vector-quantized image modeling with improved VQGAN. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022. 1, 2, 4, 5, 7, 12

[45] 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 Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 586–595, 2018. 4, 5

[46] Chuanxia Zheng, Guoxian Song, Tat-Jen Cham, Jianfei Cai, Dinh Phung, and Linjie Luo. High-quality pluralistic image completion via code shared vqgan. *arXiv preprint arXiv:2204.01931*, 2022. 2

[47] Chuanxia Zheng, Long Tung Vuong, Jianfei Cai, and Dinh Phung. Movq: Modulating quantized vectors for high-fidelity image generation. In *Proceedings of the 36st International Conference on Neural Information Processing Systems (NeurIPS)*, 2022. 1, 4, 5, 6, 7, 12

Online Clustered Codebook

A. Experiment Details

For data compression, we first demonstrate our method on small datasets with the officially released VQ-VAE [37] implementation [4][5], and we then verify the generality of our quantiser on large datasets using the officially released VQ-GAN [11] architecture [6]. For image generation application, we apply our CVQ-VAE's quantiser on LSUN dataset using the officially released LDM [32] code[7].

For the small datasets (MNIST, CIFAR10, and Fashion MNIST), we use the submitted code to train the model. The training hyperparameters match the original VQ-VAE settings, and we train all models with batch size 1,024 across 4× NVIDIA GeForce GTX TITAN X (12GB per GPU) with 500 epochs (2-3 hours).

For the high resolution datasets (FFHQ and ImageNet), we just replace the original quantiser in VQGAN with our CVQ-VAE quantiser. The training hyperparameters also follow the original settings, and we train all models with batch size 64 across 4× NVIDIA RTX A6000 (48GB per GPU) with 4 days on FFHQ and 8 days on ImageNet until converge.

For the generation (LSUN bedrooms and Churches), we use the lSUN-beds256 config file for default setting with two modifications: 1) we also replace the VQGAN's quantiser with our CVQ-VAE quantiser; 2) we reduce the images' resolution for faster training with 8× downsampling. For stage **a)** codebook learning, two models are trained with batch size 32 across 2× NVIDIA RTX A4000 (48GB per GPU) with 5 days. Then, for stage **b)** latent diffusion model with $32 \times 32 \times 4$ resolution, we train the models with batch size 128 across 2× NVIDIA RTX A4000 (48GB per GPU) with 7 days. During the inference, we follow the default settings to sample the images with 200 steps.

B. Quantitative Results

Method	**Dataset**	$\ell_1$loss ↓	SSIM ↑	PSNR ↑	LPIPS ↓	rFID ↓
VQ-VAE [37]	MNIST	0.0207	0.9777	26.48	0.0282	3.43
HVQ-VAE [39]		0.0202	0.9790	26.90	0.0270	3.17
SQ-VAE [36]		0.0197	0.9819	27.49	0.0256	3.05
CVQ-VAE		**0.0180**	**0.9833**	**27.87**	**0.0222**	**1.80**
VQ-VAE [37]	CIFAR10	0.0527	0.8595	23.32	0.2504	39.67
HVQ-VAE [39]		0.0533	0.8553	23.22	0.2553	41.08
SQ-VAE [36]		0.0482	0.8779	24.07	0.2333	37.92
CVQ-VAE		**0.0448**	**0.8978**	**24.72**	**0.1883**	**24.73**

Table B.1: **Reconstruction results** on the validation sets of MNIST (10,000 images) and CIFAR10 (10,000 images).

Table B.1 provides a comparison of our results to state-of-the-art quantisers under the same training settings, except for the different quantisers, on the small datasets. This is an extension of Tab. 1 in the main paper. All images are normalised to the range [0,1] for quantitative evaluation. See the code for more details. While the proposed CVQ-VAE achieve relative small improvements on traditional pixel-level ℓ_1 loss, peak signal-to-noise ration (PSNR), and patch-level structure similarity index (SSIM), it significantly improves the feature-level LPIPS and dataset-level rFID, suggesting that our CVQ-VAE is more capable of reconstructing the content closer to the dataset distribution.

We further compare our CVQ-VAE to the state-of-the-art methods in data compression in Tab. B.2. This is an extension of Tab. 2 in the main paper. Here, we add the pixel-level PSNR, patch-level SSIM and feature-level LPIPS. For FFHQ dataset, our CVQ-VAE model outperforms baseline variants of previous state-of-the-art models. As for ImageNet dataset, while our 4× channels setting does not achieve the better rFID than the latest MoVQ model, the other instantiations (PSNR, SSIM and LPIPS) significantly outperform existing state-of-the-art models.

Tables B.3 and B.4 are the extension of Tabs. 3 and 4c in the main paper, respectively. Even reported with the different metrics, The conclusions are still the same. For instance, the offline version is significantly affected by different anchor

[4] https://github.com/deepmind/sonnet/blob/v2/sonnet/src/nets/vqvae.py
[5] https://github.com/deepmind/sonnet/blob/v1/sonnet/examples/vqvae_example.ipynb
[6] https://github.com/CompVis/taming-transformers
[7] https://github.com/CompVis/latent-diffusion

Method	Dataset	$\mathcal{S}\downarrow$	$\mathcal{K}\downarrow$	Usage ↑	PSNR ↑	SSIM ↑	LPIPS ↓	rFID ↓
VQGAN [11]	FFHQ	16^2	1024	42%	22.24	0.6641	0.1175	4.42
ViT-VQGAN [44]		32^2	8192	–	–	–	–	3.13
RQ-VAE [22]		$16^2\times4$	2048	–	22.99	0.6700	0.1302	3.88
MoVQ [47]		$16^2\times4$	1024	56%	26.72	0.8212	0.0585	2.26
SeQ-GAN [13]		16^2	1024	100%	–	–	–	3.12
CVQ-VAE (ours)		16^2	1024	100%	26.82	0.8313	0.0608	2.80
CVQ-VAE (ours)		$16^2\times4$	1024	100%	**26.87**	**0.8398**	**0.0533**	**2.03**
VQGAN [11]	ImageNet	16^2	1024	44%	19.07	0.5183	0.2011	7.94
ViT-VQGAN [44]		32^2	8192	96%	–	–	–	1.28
RQ-VAE [22]		$8^2\times16$	16384	-	–	–	–	1.83
MoVQ [47]		$16^2\times4$	1024	63%	22.42	0.6731	0.1132	**1.12**
SeQ-GAN [13]		16^2	1024	100%	–	–	–	1.99
CVQ-VAE (ours)		16^2	1024	100%	21.95	0.6612	0.1340	1.57
CVQ-VAE (ours)		$16^2\times4$	1024	100%	**23.37**	**0.7115**	**0.1099**	1.20

Table B.2: **Reconstruction results** on validation sets of ImageNet (50,000 images) and FFHQ (10,000 images). $\mathcal{S}$ denotes the latent size of encoded features, and $\mathcal{K}$ is the number of codevectors in the codebook.

	Method	**MNIST** (28×28)			**CFAIR10** (32×32)			**Fashion MNIST** (28×28)		
		$\ell_1\downarrow$	PSNR ↑	rFID ↓	$\ell_1\downarrow$	PSNR ↑	rFID ↓	$\ell_1\downarrow$	PSNR ↑	rFID ↓
$\mathbb{A}$	Baseline VQ-VAE [37]$_{\text{NeurIPS'2017}}$	0.0207	26.48	3.43	0.0527	23.32	39.67	0.0377	23.93	12.73
$\mathbb{B}$	+ Cosine distance	0.0200	26.77	3.06	0.0509	23.66	35.14	0.0378	24.01	11.40
$\mathbb{C}$	+ Anchor initialization (offline)	0.0192	27.24	2.78	0.0481	24.16	31.10	0.0373	24.04	11.92
$\mathbb{D}$	+ Anchor initialization (online)	0.0186	27.58	2.23	**0.0445**	**24.79**	26.62	0.0349	**24.69**	9.27
$\mathbb{E}$	+ Contrastive loss	**0.0180**	**27.87**	**1.80**	0.0448	24.72	**24.73**	**0.0344**	24.66	**8.85**

Table B.3: **Results on various settings.** We add pixel-level ℓ_1 and PSNR metrics.

Method	Dataset	Offline					Online				
		$\ell_1$loss ↓	SSIM ↑	PSNR ↑	LPIPS ↓	rFID ↓	$\ell_1$loss ↓	SSIM ↑	PSNR ↑	LPIPS ↓	rFID ↓
random	MNIST	0.0195	0.9802	27.11	0.0262	3.20	**0.0185**	**0.9823**	**27.58**	**0.0236**	2.27
unique		0.0191	0.9811	27.25	0.0255	2.84	0.0186	0.9820	27.51	0.0237	2.24
probability		0.0192	0.9810	27.24	0.0253	2.78	0.0186	**0.9823**	**27.58**	**0.0236**	**2.23**
closest		**0.0186**	**0.9823**	**27.59**	**0.0242**	**2.51**	0.0187	0.9819	27.49	0.0244	2.59
random	CIFAR10	0.0494	0.8755	23.91	0.2256	34.49	0.0440	**0.9010**	24.88	0.1881	26.04
unique		0.0507	0.8705	23.15	0.2346	36.99	**0.0439**	0.9007	**24.91**	**0.1877**	26.03
probability		**0.0481**	**0.8829**	**24.16**	**0.2131**	**31.10**	0.0445	0.8991	24.79	0.1898	26.62
closest		0.0487	0.8804	24.06	0.2156	32.31	0.0444	0.8994	24.83	0.1900	**25.99**

Table B.4: **Anchor sampling methods.** The choice of anchor sampling method has a significant impact on offline (one-time) feature initialization, while the online clustered method is robust for various samplings.

sampling methods, but the online version is not sensitive to various anchor sampling methods. The online version holds very close performance with these anchor sampling methods.