

LooC: Effective Low-Dimensional Codebook for Compositional Vector Quantization

Jie Li Kwan-Yee K. Wong Kai Han
The University of Hong Kong

jieli23@hku.hk, kykwong@cs.hku.hk, kaihanx@hku.hk

Abstract

Vector quantization (VQ) is a prevalent and fundamental technique that discretizes continuous feature vectors by approximating them using a codebook. As the diversity and complexity of data and models continue to increase, there is an urgent need for high-capacity, yet more compact VQ methods. This paper aims to reconcile this conflict by presenting a new approach called LooC, which utilizes an effective Low-dimensional codebook for Compositional vector quantization. Firstly, LooC introduces a parameter-efficient codebook by reframing the relationship between codevectors and feature vectors, significantly expanding its solution space. Instead of individually matching codevectors with feature vectors, LooC treats them as lower-dimensional compositional units within feature vectors and combines them, resulting in a more compact codebook with improved performance. Secondly, LooC incorporates a parameter-free extrapolation-by-interpolation mechanism to enhance and smooth features during the VQ process, which allows for better preservation of details and fidelity in feature approximation. The design of LooC leads to full codebook usage, effectively utilizing the compact codebook while avoiding the problem of collapse. Thirdly, LooC can serve as a plug-and-play module for existing methods for different downstream tasks based on VQ. Finally, extensive evaluations on different tasks, datasets, and architectures demonstrate that LooC outperforms existing VQ methods, achieving state-of-the-art performance with a significantly smaller codebook.

1. Introduction

Vector quantization (VQ) [11, 32, 41] is a widely used technique that converts continuous feature representation into a finite set of discrete vectors, known as the codebook, allowing for efficient analysis and processing for various downstream applications such as representation learning [41, 55, 60], data compression [33, 44, 58], clustering [17, 24, 38, 51], pattern recognition [14, 42, 45], etc. The VQ codebook is essential

for minimizing distortion between input and matched codevectors. This is achieved by clustering features in the latent space to store domain information, resulting in a compact and informative representation.

With the increasing diversity and complexity of data and models, there is an urgent demand for enhanced VQ methods that incorporate efficient and effective codebooks with larger representation capacity. Increasing the codebook size can potentially improve performance by allowing for a more extensive set of representative codevectors and higher precision in data representation. However, the benefits may plateau while the computational and storage burden continues to grow. Furthermore, larger codebooks may require more training data to ensure adequate representation, which can be a limiting factor in specific applications. The trade-off of determining the optimal codebook and codevector sizes involves finding the right balance between accuracy, computational complexity, and storage requirements. It often requires empirical evaluation and experimentation to identify the ideal trade-off for a specific application or dataset.

Many methods have been developed to improve VQ. For example, [28, 33, 54] combine multiple codebooks to increase the capacity and expressiveness of the codebook for VQ. [1, 29] attempt to reduce the size of a learned codebook with a post-processing method while avoiding too much information loss caused by the reduction. [49] introduces a learnable module to quantize lower-dimensional feature vectors by projecting a learned large codebook with high-dimensional codevectors. Recent studies [40, 53] highlight the prevalent issue of codebook collapse in VQ, where only a small subset of codevectors is effectively utilized, limiting the expressive capacity. In response, the state-of-the-art (SOTA) VQ method, CVQ-VAE [53], remedies codebook collapse by updating inactive codevectors using encoded features as anchors, thereby enabling the effective learning of larger codebooks.

Product Quantization (PQ) [19] is a popular method for compressing high-dimensional vectors (such as SIFT descriptors), initially introduced for vector similarity search. It can generate an exponentially large codebook at very low

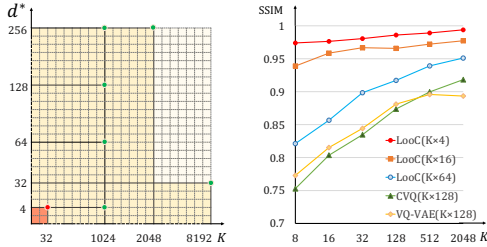


Figure 1. **Codebook size and reconstruction performance.** Left: Typical configurations (green dots) of codevector number K and dimension d^* in a codebook. LooC (red dot) stands out with a significantly smaller codebook size of 32×4 . Right: Reconstruction results on CIFAR10 [25]. LooC performs significantly better with a much smaller codebook than other SOTA methods.

memory/time cost. PQ introduces the idea of decomposing the space into a Cartesian product of low-dimensional subspaces and quantizing each subspace separately. The final codebook of PQ is constructed by taking the Cartesian product of multiple sub-codebooks, with each sub-codebook corresponding to a specific subspace. However, for complex or high-dimensional data, decomposing can generate numerous sub-codebooks, causing the final codebook to become excessively large. Despite the large codebook size overall, PQ uses separate sub-codebooks for distinct quantizers. This limits the quantization space for each subvector to a single sub-codebook. However, this constraint may hinder PQ’s ability to accurately capture fine-grained details of individual subvectors. To quantize subvectors separately, PQ utilizes multiple distinct quantizers. However, this approach can significantly increase the number of quantizers, especially when decomposing into finer granularity. Consequently, the complexity in the number of quantizers adds intricacy to the network model and its implementation, which may be considered less elegant. In this paper, we revisit the idea of decomposing the feature space into compositional sub-spaces and propose a remarkably simple and effective method called LooC. LooC introduces several innovative ideas enabling it to not only possess a very compact codebook but also achieve remarkable performance.

Firstly, we propose a parameter-efficient low-dimensional codebook (LDC), which reframes the relationship between codevectors and feature vectors by considering codevectors as compositional elements within the feature vectors. This perspective considers the codevectors in LDC as a group of “visual characters” that depict the entire visual content rather than individually associating each object feature with a “visual word”. Each vector consists of multiple codevectors within a shared codebook, which are obtained using a unified quantizer. This design greatly enhances codebook parameterization efficiency, leading to significantly reduced codebook size and notably improved performance.

Secondly, to improve the fidelity of the feature approximation by the codevectors, we further introduce a parameter-

free extrapolation-by-interpolation mechanism to enhance and smooth the features during the VQ process to preserve the details better. Notably, before mapping each feature vector to multiple codevectors, we first expand the feature map by interpolation across the spatial dimensions with a factor of β . The interpolated features are then quantized using our low-dimension codebook, resulting in an extrapolated feature map. We obtain the smoothed feature map with the original spatial dimensions by pooling the extrapolated features.

Thirdly, LooC can be seamlessly integrated as a plug-and-play module to enhance the performance of existing methods for different downstream tasks based on VQ, such as image reconstruction and generation. We extensively evaluate LooC on different tasks, datasets, and architectures, obtaining SOTA performance across the board with a much more compact codebook. Remarkably, LooC achieves comparable performance on various datasets using a significantly smaller codebook with lower-dimensional codevectors than the SOTA method CVQ [52], while maintaining a codebook utilization rate of 100%. For instance, in image reconstruction, LooC utilizes a $1024 \times$ smaller codebook than CVQ to achieve better results. With a much more compact codebook, LooC surpasses the comparative methods in image generation by a large margin, producing highly detailed and realistic images.

2. Related Work

Vector quantization (VQ) [11] is a fundamental research area with origins dating back to the 1980s and remains an enduring subject of interest for many downstream applications. Classic methods such as LBG [32] and Classified-VQ [37] use codebooks to represent a set of clustering centers for the input data. PQ [19] and OPQ [9] propose a core paradigm of decomposing high-dimensional features into low-dimensional sub-vectors and quantizing them independently. Additive Quantization (AQ) [2] compresses high-dimensional vectors by summing codewords from multiple codebooks without orthogonal subspace decomposition, reducing approximation error and boosting search/classification accuracy while maintaining PQ-like efficiency. LOPQ [21] partitions high-dimensional data into cells, locally optimizes rotation/space decomposition per cell for residual encoding, with fixed data-size-independent overhead—lower distortion, faster search on billion-scale datasets. With the rise of deep neural networks, VQ-VAE [41] introduces VQ into representation learning and employs unsupervised or self-supervised approaches to learn prior knowledge using codebooks. VQ-GAN [7] incorporates the vector quantization technique into GAN [10, 36], leveraging its benefits to enhance the generative model’s representation capacity and elevate the quality of sample generation. The effectiveness of VQ has led to its widespread adoption in diverse applications across different domains and

downstream tasks. For instance, it has been used in visual recognition [20, 24, 51], image compression and reconstruction [33, 44, 58], image retrieval [59], image segmentation [23, 52], visual and audio generation [5, 8, 12], neural radiance field [30, 43, 56], knowledge distillation [15], cross-modal retrieval and translation [3, 14, 26], vision-language models [6], and other various fields [17, 26, 38, 42, 45, 47]. In these methods, the codebook serves as the prior distribution of the discrete latent space, enabling effective modeling and manipulation of the data distribution.

Significant advancements have been made in optimizing various aspects of the codebook in vector quantization. Firstly, to improve codebook’s expressiveness and representation capacity, researchers have explored different approaches. AdaCode [33] learns a set of basis codebooks for each image category and introduces a weight map for adaptive image restoration. MoVQ [54] incorporates two convolutional layers into the decoder to learn modulation parameters from embedding vectors, converting discrete representations into scaled and biased values. Additionally, efforts have been made to address the codebook collapse problem and increase its usage. SQ-VAE [40] introduces stochastic dequantization and quantization techniques to tackle the problem. The SOTA method, CVQ [53], proposes Online Clustered Codebook learning as a solution by updating inactive codevectors using encoded features as anchors. Moreover, various techniques have been proposed to reduce the size of codebooks in VQ. One approach in [1] entails sorting the codevectors and then utilizing Huffman coding on the differences between adjacent codevectors.

[34] adjusts the codebook size using the K-means clustering algorithm. HyperHill [29] utilizes hyperbolic embeddings to enhance codebook vectors with co-occurrence information and rearranges the codebook using the Hilbert curve. Another method [49] adopts a smaller codebook by introducing a linear projection from the encoder’s output to a low-dimensional latent variable space, albeit with an increased number of codebooks. Furthermore, several methods for composite quantizations have been introduced. SQ [35] introduces the approach of iteratively quantizing a vector and its residuals to represent the vector as a stack of codes, known as stacked quantization. RQ [28] uses residual to approximate the feature vector recursively in a coarse-to-fine manner. TQR [31] enhances RQ effect for ternary neural networks by combining binarized stem and residual parts.

Overall, these techniques consider a feature vector as a unified entity and apply codebooks with the same dimension to process it. In contrast, we consider each feature vector as the composition of multiple smaller codevectors combined through concatenation. The effectiveness of segmenting the feature vector into subvectors during quantization has been demonstrated by PQ [19] and SPQ [18] in the for visual search. LooC extends this design to image reconstruction

and enhances it to generate embeddings with enhanced representation capabilities. Unlike PQ, which employs several distinct codebooks, LooC utilizes a single codebook shared across all subvectors to increase the combinatorial nature of all codevectors and thus expand the matching space of the codebook. PQ constructs the product quantization codebook using handcrafted features and enforces orthogonal constraints. In contrast, LooC directly learns a lightweight and expressive codebook that can faithfully reconstruct the image by training on image reconstruction. Besides, LooC employs a unified quantizer for all subvectors, unlike PQ which employs multiple distinct sub-quantizers, thus enabling an elegant and unified treatment for all features and codevectors.

3. Method

This section introduces our proposed quantizer, LooC, with an effective low-dimensional codebook for compositional VQ. We first briefly review VQ-VAE [41] and PQ [19] as the preliminary. We then present LooC and elucidate how LooC attains remarkable performance with a low-dimensional codebook and feature enhancement through extrapolation-by-interpolation. We also demonstrate how LooC resolves the collapse problem, ensuring the complete utilization of its representation space.

3.1. Preliminary

VQ-VAE for Visual Representation. Consider an encoder $\Phi_{\text{ENC}}(\cdot)$ and a decoder $\Phi_{\text{DEC}}(\cdot)$. Let $\mathcal{Q}(\cdot)$ denote the quantization operator. The feature vector $z_{i,j} \in \mathbb{R}^d$ at each spatial location (i, j) is then quantized by replacing it with the most similar codevector in the codebook \mathcal{C} , *i.e.*,

$$\mathcal{Q}(z_{i,j}; \mathcal{C}) = \arg \min_{k \in [K]} \|z_{i,j} - c(k)\|, \quad \mathcal{C} = \{(k, c(k))\}_{k=1}^K, \quad (1)$$

where k is the code index, $c(k) \in \mathbb{R}^{d^*}$ is the corresponding codevector in \mathcal{C} . To ensure that $z_{i,j}$ can be replaced by $c(k)$, they must have the same dimension, *i.e.*, $d^* = d$.

Product Quantization (PQ). In PQ, an input vector \bar{z} is split into m distinct subvectors \bar{z}_λ , $1 \leq \lambda \leq m$, of dimension $d^* = d/m$, where d is a multiple of m . The subvectors are quantized separately using m distinct quantizers. The given vector $\bar{z} = (a_1, a_2, \dots, a_d)$ is therefore mapped as follows:

$$\underbrace{a_1, \dots, a_{d^*}}_{\bar{z}_1(a)}, \dots, \underbrace{a_{d-d^*+1}, \dots, a_d}_{\bar{z}_m(a)} \rightarrow q_1(\bar{z}_1(a)), \dots, q_m(\bar{z}_m(a)), \quad (2)$$

where q_λ is a low-complexity quantizer associated with the λ -th subvector. The codebook \mathcal{C}_λ can be associated with the corresponding reproduction values $c_{\lambda,k}$, using sub-quantizer q_λ . The complete codebook \mathcal{C} is defined as the Cartesian product of m distinct sub-codebooks $\mathcal{C} = \mathcal{C}_1 \times \dots \times \mathcal{C}_m$.

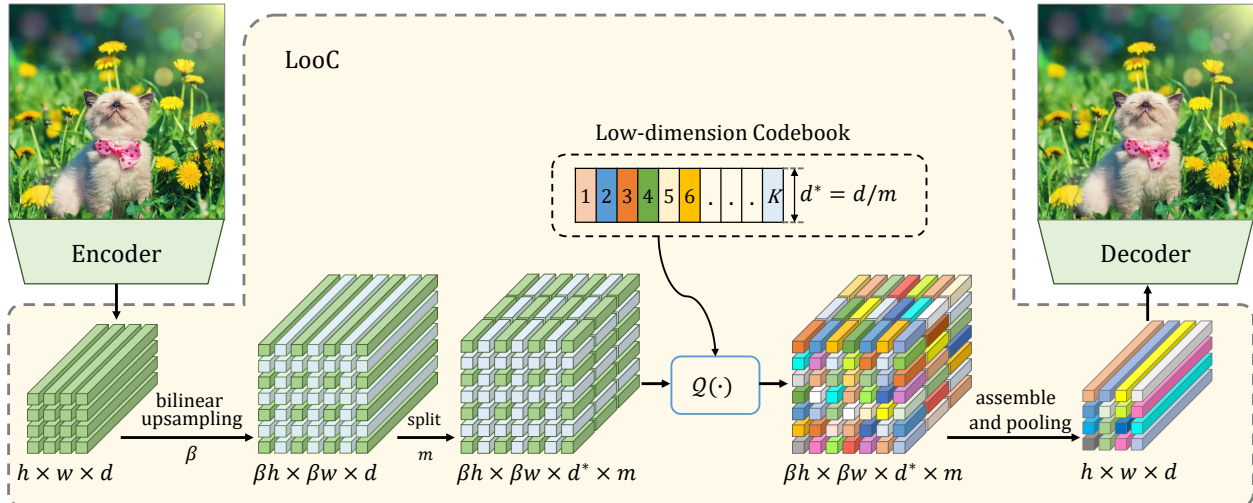


Figure 2. **Framework of Low-dimensional codebook for Compositional vector quantization (LooC).** The encoder transforms the input image into a continuous latent feature map z . z is then upsampled using bilinear interpolation with scale factor β . Simultaneously, each feature vector in z is divided into m units and quantized using a shared codebook \mathcal{C} containing K codevectors of dimension $d^* = d/m$. The quantized units are then reassembled and smoothed using average pooling to restore the shape as z . Finally, the decoder converts the feature map back to the image.

3.2. LooC: Learning low-dimensional Codebook for Compositional VQ

Instead of employing multiple separate sub-quantizers as in PQ, we adopt a unified quantizer applied to all subvectors in LooC. Additionally, LooC utilizes a shared codebook for subvectors, eliminating the need for multiple codebooks and preventing the explosive growth of codebooks in PQ by avoiding Cartesian products. Moreover, LooC incorporates an extrapolation-by-interpolation mechanism that enhances and smooths features, preserving details and ensuring accurate feature approximation.

Low-dimensional Codebook (LDC). Instead of treating codevectors as prototypes like PQ, we adopt a different perspective by considering codevectors as compositional units within feature vectors. Analogously, instead of considering the codevectors as “visual words” we consider them as “visual characters”. By leveraging the inherent compositional nature of feature vectors, LooC effectively captures the underlying structure of the data using a reduced number of codevectors. Specifically, for each feature vector $z_{i,j} \in \mathbb{R}^d$, we use a unified quantizer $\mathcal{Q}(\cdot)$ to quantize m sub-vectors at the same time to achieve compositional quantization of $z_{i,j}$ with m codevectors in the codebook $\mathcal{C} = \{(k, c(k))\}_{k=1}^K$, where k is the code index, $c(k) \in \mathbb{R}^{d^*}$ is the corresponding codevector and $d^* = d/m$ is the dimension of each codevector. The m code indices can be obtained by quantization operation as follows $\mathcal{I}_{i,j} = \{\mathcal{Q}(z_{i,j}[(q-1)d^* : qd^*]; \mathcal{C})\}_{q=1}^m$ with which $z_{i,j}$ can be quantized by concatenating the codevectors corresponding to $\mathcal{I}_{i,j}$ in \mathcal{C} . Different from PQ, we utilize a shared quantizer for all subvectors instead of employing several distinct quantizers. Furthermore, we elim-

inate the need for multiple sub-codebooks and Cartesian product to obtain the final codebook; instead, a single shared codebook with a compact size is sufficient.

Feature Enhancement and Smoothness. In addition to the low-dimensional codebook design, LooC incorporates a parameter-free extrapolation-by-interpolation mechanism to enhance and smooth features during the vector quantization process, preserving details and ensuring accurate feature approximation. After obtaining the latent feature map z from the encoder $\Phi_{\text{ENC}}(\cdot)$, we first employ bilinear interpolation to interpolate the feature map across the spatial dimensions by a scaling factor β . This step leads to a larger feature map $z^{it} \in \mathbb{R}^{\beta h \times \beta w \times (d^* \times m)}$ with an increased number of feature vectors for VQ. Next, for each original feature vector and its interpolated neighbors, we quantize them based on our LDC \mathcal{C} , leading to an extrapolated feature map $z^{ex} \in \mathbb{R}^{\beta h \times \beta w \times (d^* \times m)}$. This way, the feature expressiveness is enhanced. Finally, we smooth the feature by adopting average pooling on features around each quantized original feature, resulting in a feature map $\tilde{z} \in \mathbb{R}^{h \times w \times (d^* \times m)}$, which can then be decoded by the decoder $\Phi_{\text{DEC}}(\cdot)$ to reconstruct the input image.

3.3. Codebook Compactness and Exponential Representation Capacity

A key strength of LDC is that it can produce a large set of “visual words” by combining a small set of “visual characters”. The feature vector of length d is divided into m sub-vectors, each with a length of $d^* = d/m$. Thus, the value of d^* determines the granularity of the codevectors in LDC. A smaller d^* indicates a more fine-grained representation of

the individual components in visual features. Conversely, a larger d^* signifies a coarser level of compositional units. Using a shared codebook with K codevectors allows for possible combinations of K^m when considering m sub-vectors. In contrast, PQ needs multiple independent codebooks to achieve such combinations. Note that, when $d^* = d$, LDC degenerates to the vanilla codebook. In this case, $m = 1$, which means that each feature vector only has K possible choices from the codebook.

The increase of m leads to the gradual improvement of VQ reconstruction accuracy [57]. This effect is particularly pronounced when the number of codevectors K is small. As depicted in Fig 1-Right, we can observe that as m increases, $d^* = d/m$ decreases, and the use of LooC effectively mitigates the adverse impact of reducing K . For detailed analysis, see Sec. 4.2. Note that, in the extreme case where $d^* = 1$, each value in the vector $z_{i,j}$ are all quantized separately.

The parameter β in the extrapolation-by-interpolation operation controls the feature enhancement and smoothness level. During extrapolation, this operation improves the capacity of its own representation vector by integrating information from neighboring vectors. Our experiments show that setting β to 2 consistently leads to strong performance.

3.4. Training of LooC

An input image x is converted into a feature map z through the encoder $\Phi_{\text{ENC}}(\cdot)$ by $z = \Phi_{\text{ENC}}(x)$. Let \hat{z} be the quantized feature map of z . The image can then be reconstructed by the decoder with $\hat{x} = \Phi_{\text{DEC}}(\hat{z})$.

The encoder $\Phi_{\text{ENC}}(\cdot)$, decoder $\Phi_{\text{DEC}}(\cdot)$, and codebook \mathcal{C} are jointly optimized by minimizing the loss:

$$\mathcal{L} = \|x - \hat{x}\|_2^2 + \|\text{sg}[z] - \hat{z}\|_2^2 + \mu \|z - \text{sg}[\hat{z}]\|_2^2, \quad (3)$$

where sg is a stop gradient operator, the first term is referred to as the reconstruction loss, the second as the codebook loss, and the third as the commitment loss. We develop our method based on VQ-VAE [41]. Moreover, we follow [53] to update codevectors to avoid codebook collapse using features as anchors.

4. Experiments

Applications. To validate the effectiveness of LooC, we conduct experiments in two downstream tasks: *image reconstruction* and *image generation*. In the reconstruction task, we compare LooC with various VQ methods. For a fair comparison, we integrate LooC into VQ-VAE’s training framework by replacing the VQ module, following the SOTA method CVQ [53]. We also reimplement the quantizers in PQ [19] and, like other approaches, apply the VQ-VAE network architecture for training. Afterward, we assess the generalizability of LooC on larger datasets by employing the VQ-GAN [7] architecture. Our experiments use the

Method	$K \times d^* \downarrow$	LPIPS \downarrow	rFID \downarrow	SSIM \uparrow	PSNR \uparrow
VQ-VAE [41]	1024 \times 128	0.0282	3.43	0.9777	26.48
HVQ-VAE [44]	1024 \times 128	0.0270	3.17	0.9790	26.90
SQ-VAE [40]	1024 \times 128	0.0256	3.05	0.9819	27.49
CVQ-VAE [53]	1024 \times 128	0.0222	1.80	0.9833	27.87
PQ [19]	256 \times 4 \times #32	0.0120	1.76	0.9933	32.32
LooC	32 \times 4	0.0083	1.70	0.9961	35.15
LooC	256 \times 4	0.0058	1.31	0.9976	37.58
VQ-VAE [41]	1024 \times 128	0.2504	39.67	0.8595	23.32
HVQ-VAE [44]	1024 \times 128	0.2553	41.08	0.8553	23.22
SQ-VAE [40]	1024 \times 128	0.2333	37.92	0.8779	24.07
CVQ-VAE [53]	1024 \times 128	0.1883	24.73	0.8978	24.72
PQ [19]	256 \times 4 \times #32	0.0953	27.15	0.9527	28.27
LooC	32 \times 4	0.0435	24.53	0.9805	32.22
LooC	256 \times 4	0.0285	19.22	0.9880	34.51

Table 1. **Image reconstruction results** on low-resolution datasets of MNIST [27] and CIFAR10 [25]. LooC outperforms other SOTA methods with a significantly reduced codebook size of 32 \times 4, which is 1024 \times smaller than 1024 \times 128 used by most SOTAs.

Method	$K \times d^* \downarrow$	Usage \uparrow	rFID \downarrow	SSIM \uparrow	PSNR \uparrow
VQGAN [7]	1024 \times 256	42%	4.42	0.6641	22.24
ViT-VQGAN [49]	8192 \times 32	–	3.13	–	–
RQ-VAE [28]	2048 \times 256	–	3.88	0.6700	22.99
MoVQ [54]	1024 \times 64	56%	2.26	0.8212	26.72
SeQ-GAN [13]	1024 \times 256	100%	3.12	–	–
CVQ-VAE [53]	1024 \times 256	100%	2.03	0.8398	26.87
LooC-VAE	256 \times 4	100%	1.97	0.8499	27.73
LooC-VAE	1024 \times 4	100%	1.37	0.9276	32.44
VQGAN [7]	1024 \times 256	44%	7.94	0.5183	19.07
ViT-VQGAN [49]	8192 \times 32	96%	1.28	–	–
RQ-VAE [28]	2048 \times 256	–	1.83	–	–
MoVQ [54]	1024 \times 64	63%	1.12	0.6731	22.42
SeQ-GAN [13]	1024 \times 256	100%	1.99	–	–
CVQ-VAE [53]	1024 \times 256	100%	1.57	0.7115	23.37
LooC-VAE	256 \times 4	100%	1.68	0.7233	23.64
LooC-VAE	1024 \times 4	100%	1.01	0.7160	29.15

Table 2. **Image reconstruction** on high-resolution datasets of FFHQ [22] and ImageNet [4]. LooC has a compact codebook size of 256 \times 4, which is 256 \times smaller than most SOTA methods’ 1024 \times 256.

same backbone network and codebook update method as CVQ-VAE [53]. For image generation, we use the LDM framework [39] and replace the VQ module with LooC and other VQ methods.

Datasets. We examine and verify our method on various datasets: MNIST [27], CIFAR10 [25], and FASHION-MNIST [46]. After that, we evaluate our method on larger datasets such as ImageNet [4], FFHQ [22], and LSUN [48].

Metrics. Following previous works [41, 53], we compare the quality of reconstructed images to their original counterparts using various metrics, including the patch-level structure similarity index (SSIM), feature-level Learned Perceptual Image Patch Similarity (LPIPS) [50], image-level Peak Signal-to-Noise Ratio (PSNR), and dataset-level Fréchet Inception Distance (FID) [16].

4.1. Comparison to Prior Work

Quantitative Results. We conduct experiments on both small datasets like MNIST [27] and CIFAR10 [25], as well as large datasets including ImageNet[4] and FFHQ [22]. In

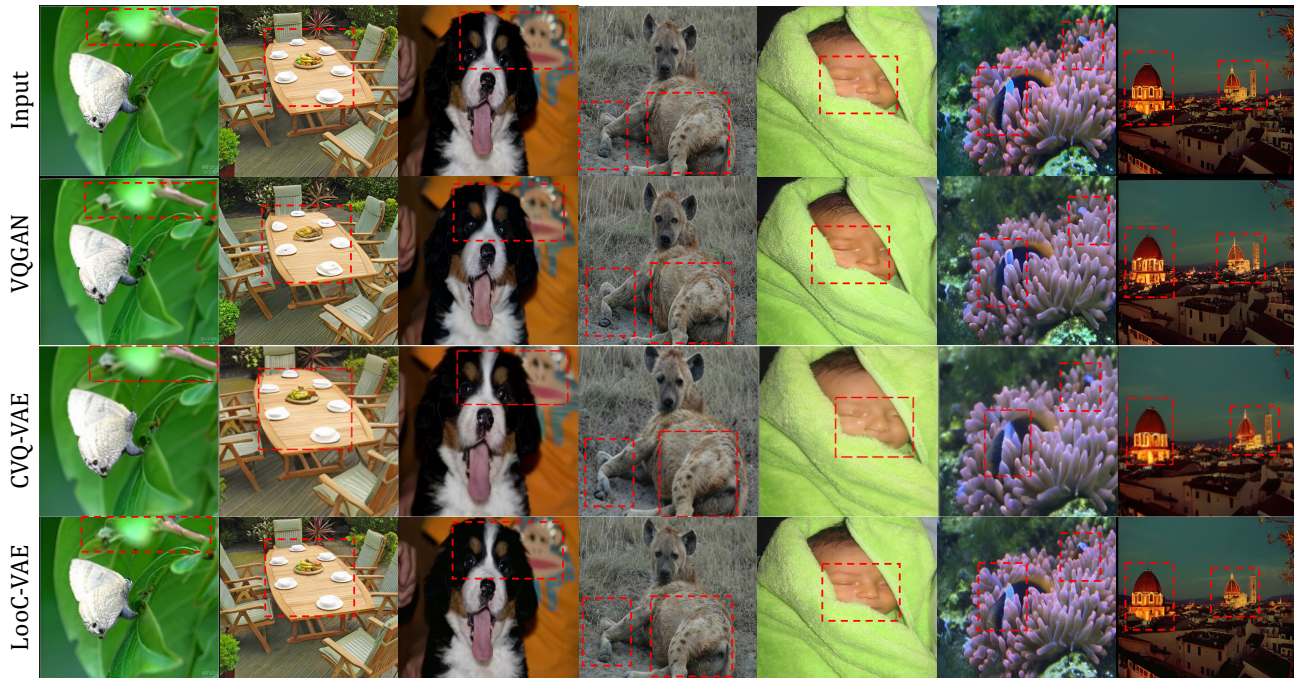


Figure 3. **Qualitative results.** Reconstructed images using VQGAN [7], CVQ [53], and LooC. LooC significantly enhances reconstruction quality by preserving image details and restoring texture structures, as highlighted in the red boxes (best viewed in PDF with zoom).

Tab. 1, we compare LooC with various VQ methods such as those developed in VQ-VAE [41], HVQ [44], SQVAE [40], and CVQ-VAE [53], as well as the reimplemented PQ [19] with 32 independent codebooks. As shown in Tab. 1, it is evident that our method outperforms other techniques across multiple metrics. One of the key advantages of our method is that LooC’s codebook size is significantly smaller than that of the previous SOTA method CVQ-VAE’s. LooC with a size of only 32×4 achieves better performance than CVQ-VAE with a size of 1024×128 . In this case, LooC has a $1024 \times$ smaller codebook size than CVQ-VAE. Despite the smaller codebook size, our method achieves significantly better results than CVQ-VAE, as evidenced by the notably reduced rFID score of 1.31 on MNIST and 19.22 on CIFAR10. Our method also clearly outperforms PQ in all aspects, despite using a significantly smaller codebook. Furthermore, our shared codebook allows us to achieve better results than PQ. The results using the PSNR metric further highlight the strength of our method.

Next, in Tab.2, we verify the effectiveness of our method on more challenging large-scale datasets with high resolution, FFHQ [22] and ImageNet [4]. Our LooC-VAE is compared to current SOTA methods, including CVQ-VAE [53], VQ-GAN [7], ViT-VQGAN [49], RQ-VAE [28], SeQ-GAN [13], and MoVQ [54], for the reconstruction task. Our method consistently outperforms previous SOTA methods across both datasets, as revealed in Tab. 2. In comparison to CVQ-VAE [53], our LooC-VAE (256×4) achieves similar or better results while using a $256 \times$ smaller codebook.

Simultaneously, our method achieves 100% codebook utilization as shown in Tab.2. This enables the optimization of all codevectors and ensures the parameter efficiency of the codebook. These experimental results demonstrate that our codebook has significant advantages in being lightweight in terms of dimension and quantity of codevectors.

Qualitative Results. In Fig. 3, we compare the visualization results between our method, LooC-VAE, and the SOTA techniques, including VQ-GAN [7] and CVQ-VAE [53]. Our method excels in preserving image details and restoring texture structures, offering significant advantages over other methods. This is evident when examining the regions highlighted by the red boxes in Fig 3. Notably, the *fruits on the table* in the second column, the *paws of the hyena* in the fourth column, and the *buildings* in the last column demonstrate the superiority of our approach. Our approach achieves high fidelity in restoring the original image’s appearance, unlike other methods that may cause losses or distortions. This is crucial for downstream tasks.

4.2. Expressiveness of LooC

In this section, we investigate the reasons behind LooC’s exceptional performance and compact design. We primarily focus on the compositional VQ and extrapolation-by-interpolation operations, which are the two most critical components of LooC.

Enhanced Capacity via Fine-Grained Combination.

Analyzing Fig. 1-Right, we make two notable observations during our preliminary performance analysis of LooC.

Method	$K \times d^* \downarrow$	MNIST(28 × 28) / CIFAR10(32 × 32) / FASHION-MNIST(28 × 28)					
		l_1 loss↓	LPIPS↓	rFID↓	SSIM↑	PSNR↑	
VQ-VAE	1024 × 128	0.0207 / 0.0527 / 0.0377	0.0282 / 0.2504 / 0.0801	3.43 / 39.67 / 12.73	0.9777 / 0.8595 / 0.9140	26.48 / 23.32 / 23.93	
CVQ-VAE	1024 × 128	0.0180 / 0.0448 / 0.0344	0.0222 / 0.1883 / 0.0693	1.80 / 24.73 / 8.85	0.9833 / 0.8978 / 0.9233	27.87 / 24.72 / 24.66	
LooC-VAE	256 × 4	0.0062 / 0.0144 / 0.0103	0.0058 / 0.0285 / 0.0098	1.31 / 19.22 / 6.24	0.9976 / 0.9880 / 0.9924	37.58 / 34.51 / 35.34	
LooC-VAE	256 × 8	0.0068 / 0.0188 / 0.0115	0.0064 / 0.0430 / 0.0114	1.40 / 24.17 / 6.93	0.9972 / 0.9809 / 0.9907	36.79 / 32.30 / 34.42	
LooC-VAE	256 × 16	0.0097 / 0.0246 / 0.0180	0.0100 / 0.0681 / 0.0211	1.88 / 31.55 / 10.93	0.9949 / 0.9679 / 0.9796	33.73 / 29.97 / 30.62	
LooC-VAE	256 × 32	0.0118 / 0.0295 / 0.0236	0.0125 / 0.0934 / 0.0325	2.19 / 37.58 / 13.95	0.9928 / 0.9551 / 0.9660	32.05 / 28.43 / 28.20	
LooC-VAE	256 × 64	0.0138 / 0.0367 / 0.0288	0.0157 / 0.1435 / 0.0476	2.51 / 51.06 / 17.08	0.9903 / 0.9293 / 0.9497	30.58 / 26.46 / 26.42	

Table 3. **Results under various compositional granularity** using a codebook with $K = 256$ codevectors. As the value of d^* decreases from 64 to 4, achieved by increasing the compositional granularity m from 2 to 32, our method consistently improves performance on all three datasets.

Method	$K \times d^* \downarrow$	MNIST(28 × 28) / CIFAR10(32 × 32) / FASHION-MNIST(28 × 28)					
		l_1 loss↓	LPIPS↓	rFID↓	SSIM↑	PSNR↑	
VQ-VAE	1024 × 128	0.0207 / 0.0527 / 0.0377	0.0282 / 0.2504 / 0.0801	3.43 / 39.67 / 12.73	0.9777 / 0.8595 / 0.9140	26.48 / 23.32 / 23.93	
CVQ-VAE	1024 × 128	0.0180 / 0.0448 / 0.0344	0.0222 / 0.1883 / 0.0693	1.80 / 24.73 / 8.85	0.9833 / 0.8978 / 0.9233	27.87 / 24.72 / 24.66	
LooC-VAE	256 × 128	0.0166 / 0.0464 / 0.0344	0.0213 / 0.2144 / 0.0680	3.17 / 64.71 / 21.76	0.9853 / 0.8923 / 0.9256	28.63 / 24.46 / 24.71	
LooC-VAE	512 × 64	0.0122 / 0.0343 / 0.0266	0.0133 / 0.1249 / 0.0410	2.28 / 45.23 / 16.02	0.9923 / 0.9392 / 0.9566	31.77 / 27.09 / 27.16	
LooC-VAE	1024 × 32	0.0094 / 0.0261 / 0.0197	0.0095 / 0.0735 / 0.0243	1.84 / 33.51 / 11.59	0.9952 / 0.9643 / 0.9760	34.10 / 29.45 / 29.83	
LooC-VAE	2048 × 16	0.0073 / 0.0205 / 0.0138	0.0070 / 0.0490 / 0.0144	1.51 / 26.29 / 8.29	0.9968 / 0.9773 / 0.9873	36.14 / 31.55 / 32.84	
LooC-VAE	4096 × 8	0.0056 / 0.0146 / 0.0098	0.0049 / 0.0293 / 0.0091	1.17 / 19.40 / 5.91	0.9980 / 0.9878 / 0.9929	38.39 / 34.40 / 35.71	
LooC-VAE	8192 × 4	0.0047 / 0.0099 / 0.0083	0.0040 / 0.0154 / 0.0070	0.99 / 12.65 / 4.86	0.9984 / 0.9937 / 0.9947	39.71 / 37.49 / 37.05	

Table 4. **Results under a fixed codebook size of $s = K \times d^*$ with varying K and d^* .** The compositional granularity parameter m changes proportionally to d^* as $m = d/d^*$.

Firstly, as the number of codevectors K increases, the performance of all methods shows improvement. Secondly, our approach surpasses other SOTA methods, exhibiting a more substantial advantage specifically when K is smaller (e.g., $K = 32$). Additionally, our method demonstrates improved performance as m increases, leading to a lower dimension d^* in our LooC.

Two key findings emerge from this analysis. Firstly, the increase in codebook size by adopting a larger K substantially impacts VQ’s accuracy. This is consistent with the fact that existing methods exploit larger K to enhance the effectiveness of VQ, highlighting the challenging nature of reducing codebook size. Secondly, our LooC exhibits greater adaptability when K is small, while maintaining excellent accuracy performance. LooC achieves this by controlling the granularity parameter m in compositional VQ, with larger m values demonstrating better adaptability to small K values.

We further verify the importance of the compositional VQ through experiments, which are shown in Tab. 3. In the experiment, we set $K = 256$ and vary the value of m , gradually increasing it from 2 to 32. The dimension d of each latent feature vector is 128, resulting in a decrease of d^* from 64 to 4. Our method demonstrates consistent performance improvement across three different datasets. For instance, on CIFAR10, the PSNR gradually increases from 26.46 to 34.51, and the rFID decreases from 51.06 to 19.22. Similarly, other indicators also exhibit a gradual and consistent improvement. This highlights the effectiveness of our approach, which exploits fine-grained combinations, enhancing the capacity of the codebook and leading to better overall performance.

Compact Codebook with Low-dimension. In Tab. 4, we maintain a constant total codebook size of $s = K \times d^*$, while adjusting K and d^* . As d^* decreases, resulting in an increase of $m = d/d^*$, K increases as well. This indicates that smaller individual codevectors allow for more codevectors to be added while still maintaining the same overall codebook size. The experimental results clearly demonstrate that as d^* decreases (or m increases), the effectiveness of our method gradually improves, expanding its advantages over existing SOTA approaches. For instance, by increasing the value of m from 1 to 32 in LooC($K = 256, d^* = 128/m$) on the MNIST dataset, we observe a significant improvement in PSNR, increasing from 28.6 to 39.7. All three datasets showed improved performance in various metrics. Based on this fact, we investigate the possibility of reducing the number of codevectors in the codebook. The exploration results are shown in Tab. 5. We kept m fixed at 32 by maintaining $d^* = 4$, and only varied the value of K in our experiments. Our method is still able to maintain a high performance even when K is small. Furthermore, the performance of our method improves progressively as K increases. For instance, when referring to the rFID, our method with a codebook ($K = 32, d^* = 4$) achieves results similar to CVQ ($K = 1024, d^* = 128$) on the CIFAR10 dataset. This corresponds to a reduction in codebook size by a factor of $1024 \times$. Similar observations are made on the MNIST and FASHION-MNIST datasets.

Parameter-free Extrapolation-by-interpolation. Tab. 6 evaluates the effectiveness of the extrapolation-by-interpolation operation from two perspectives. Firstly, we compare our approach with the SOTA method CVQ [53]

Method	$K \times d^* \downarrow$	MNIST(28 × 28) / CIFAR10(32 × 32) / FASHION-MNIST(28 × 28)					
		l_1 loss \downarrow	LPIPS \downarrow	rFID \downarrow	SSIM \uparrow	PSNR \uparrow	
VQ-VAE	1024 × 128	0.0207 / 0.0527 / 0.0377	0.0282 / 0.2504 / 0.0801	3.43 / 39.67 / 12.73	0.9777 / 0.8595 / 0.9140	26.48 / 23.32 / 23.93	
CVQ-VAE	1024 × 128	0.0180 / 0.0448 / 0.0344	0.0222 / 0.1883 / 0.0693	1.80 / 24.73 / 8.85	0.9833 / 0.8978 / 0.9233	27.87 / 24.72 / 24.66	
LooC-VAE	8 × 4	0.0095 / 0.0220 / 0.0168	0.0097 / 0.0559 / 0.0197	1.86 / 28.35 / 9.92	0.9950 / 0.9740 / 0.9819	33.90 / 30.96 / 31.26	
LooC-VAE	16 × 4	0.0079 / 0.0211 / 0.0162	0.0080 / 0.0521 / 0.0184	1.61 / 27.13 / 9.89	0.9964 / 0.9763 / 0.9833	35.53 / 31.32 / 31.53	
LooC-VAE	32 × 4	0.0082 / 0.0189 / 0.0145	0.0083 / 0.0435 / 0.0158	1.70 / 24.53 / 8.73	0.9961 / 0.9805 / 0.9864	35.15 / 32.22 / 32.49	
LooC-VAE	64 × 4	0.0075 / 0.0178 / 0.0133	0.0072 / 0.0407 / 0.0140	1.54 / 23.70 / 8.05	0.9967 / 0.9825 / 0.9883	36.02 / 32.73 / 33.26	
LooC-VAE	128 × 4	0.0068 / 0.0158 / 0.0114	0.0065 / 0.0332 / 0.0112	1.43 / 21.15 / 6.90	0.9972 / 0.9859 / 0.9910	36.77 / 33.70 / 34.59	
LooC-VAE	256 × 4	0.0062 / 0.0144 / 0.0103	0.0058 / 0.0285 / 0.0098	1.31 / 19.22 / 6.24	0.9976 / 0.9880 / 0.9924	37.58 / 34.51 / 35.34	
LooC-VAE	512 × 4	0.0055 / 0.0137 / 0.0092	0.0051 / 0.0258 / 0.0083	1.18 / 18.14 / 5.50	0.9980 / 0.9891 / 0.9937	38.40 / 34.92 / 36.24	
LooC-VAE	1024 × 4	0.0051 / 0.0116 / 0.0083	0.0045 / 0.0192 / 0.0073	1.10 / 14.10 / 5.07	0.9982 / 0.9918 / 0.9946	38.97 / 36.38 / 37.10	

Table 5. Results with various $K \in \{8, 16, \dots, 1024\}$ of codevectors with fixed $d^* = 4$.



Figure 4. Unconditional image generation on LSUN [48] and class-conditional image generation on Imagenet [4].

Method	β	$K \times d^* \downarrow$	LPIPS \downarrow	rFID \downarrow	SSIM \uparrow	PSNR \uparrow
VQ-VAE	-	1024 × 256	0.1175	4.42	0.6641	22.24
CVQ-VAE	-	1024 × 256	0.0533	2.03	0.8398	26.87
LooC-VAE	2	1024 × 256	0.0532	1.83	0.8627	27.02
LooC-VAE	1	256 × 4	0.0528	2.27	0.8571	27.19
LooC-VAE	2	256 × 4	0.0501	1.97	0.8499	27.73
LooC-VAE	3	256 × 4	0.0478	1.89	0.8523	27.00
LooC-VAE	4	256 × 4	0.0490	2.01	0.8510	26.88

Table 6. Results on various settings on FFHQ [22].

on the high-resolution dataset FFHQ [22]. For this comparison, we set $m = 1$, making our LDC equivalent to the standard codebook. Other parameters such as $K = 1024$ and $d^* = d = 256$ remain consistent with CVQ. We then apply our proposed extrapolation-by-interpolation mechanism with $\beta = 2$. The results show that our proposed mechanism significantly improves VQ’s accuracy. For instance, compared to CVQ, our method reduces the rFID score from 2.03 to 1.83, while increasing the PSNR from 26.87 to 27.02.

Subsequently, we utilize LooC with $K = 256$ and $d^* = 4$, setting different $\beta \in \{1, 2, 3, 4\}$. The corresponding experimental results are presented in Tab. 6. We find that employing a bilinear difference with $\beta = 2$ can result in significant improvements. As β increases, the spatial correlation of the vectors weakens, leading to a gradual decline in the effectiveness of the improvement. Therefore, we choose $\beta = 2$, and accordingly, we can combine this mechanism with other

VQ techniques to further improve accuracy.

4.3. Plug-and-play for Image Generation

We use our quantizer LooC to assess its effectiveness in image generation, following the advanced CVQ [53] method. We replace the VQ module in VQGAN [7] with LooC and then apply it to the LDM [39] method, training it on LSUN [48] and ImageNet [4]. The results of unconditional image generation on LSUN and category-conditional image generation on ImageNet are presented in Fig. 4. Our approach produces highly detailed and realistic images, demonstrating that our method is a practical plug-and-play module suitable for various downstream tasks.

5. Conclusion

We have presented LooC, a highly efficient quantizer with a low-dimensional codebook for compositional vector quantization. LooC not only delivers exceptional performance but also has a remarkably compact codebook. By treating codevectors as compositional units within feature vectors, LooC achieves a more condensed codebook without compromising performance. Moreover, LooC incorporates an extrapolation-by-interpolation mechanism that enhances and smooths features, ensuring accurate feature approximation and preserving intricate details. Our quantizer offers a simple yet effective solution that can seamlessly integrate into existing architectures for representation learning.

Acknowledgement

This work is supported by National Natural Science Foundation of China (Grant No. 62306251), Hong Kong Research Grants Council - General Research Fund (Grant No.: 17211024), and HKU Seed Fund for Basic Research.

References

- [1] Sahar K Ahmed. New method to reduce the size of codebook in vector quantization of images. *AL-Rafidain Journal of Computer Sciences and Mathematics*, 2(1):53–62, 2005. 1, 3
- [2] Artem Babenko and Victor Lempitsky. Additive quantization for extreme vector compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 931–938, 2014. 2
- [3] Yue Cao, Mingsheng Long, Jianmin Wang, and Shichen Liu. Collective deep quantization for efficient cross-modal retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017. 3
- [4] 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*, pages 248–255. Ieee, 2009. 5, 6, 8, 3, 4
- [5] 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. 3
- [6] 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*, pages 552–560, 2023. 3
- [7] 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*, pages 12873–12883, 2021. 2, 5, 6, 8, 3, 4
- [8] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion. In *International Conference on Learning Representations*, 2023. 3
- [9] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. Optimized product quantization. *IEEE transactions on pattern analysis and machine intelligence*, 36(4):744–755, 2013. 2
- [10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. 2
- [11] Robert Gray. Vector quantization. *IEEE Assp Magazine*, 1(2):4–29, 1984. 1, 2
- [12] Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. Vector quantized diffusion model for text-to-image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10696–10706, 2022. 3
- [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. 5, 6, 3
- [14] Haohan Guo, Fenglong Xie, Xixin Wu, Frank K Soong, and Helen MengFellow. Msmc-tts: Multi-stage multi-codebook vq-vae based neural tts. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023. 1, 3
- [15] Liyong Guo, Xiaoyu Yang, Quandong Wang, Yuxiang Kong, Zengwei Yao, Fan Cui, Fangjun Kuang, Wei Kang, Long Lin, Mingshuang Luo, et al. Predicting multi-codebook vector quantization indexes for knowledge distillation. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1–5. IEEE, 2023. 3
- [16] 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. *Advances in neural information processing systems*, 30, 2017. 5
- [17] Qiyu Hu, Guangyi Zhang, Zhijin Qin, Yunlong Cai, Guanding Yu, and Geoffrey Ye Li. Robust semantic communications with masked vq-vae enabled codebook. *IEEE Transactions on Wireless Communications*, 2023. 1, 3
- [18] Young Kyun Jang and Nam Ik Cho. Self-supervised product quantization for deep unsupervised image retrieval. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12085–12094, 2021. 3
- [19] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2010. 1, 2, 3, 5, 6, 4
- [20] Frederic Jurie and Bill Triggs. Creating efficient codebooks for visual recognition. In *Tenth IEEE International Conference on Computer Vision*, pages 604–610. IEEE, 2005. 3
- [21] Yannis Kalantidis and Yannis Avrithis. Locally optimized product quantization for approximate nearest neighbor search. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2321–2328, 2014. 2
- [22] 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. 5, 6, 8, 1, 3, 4
- [23] HB Kekre, Tanuja K Sarode, and Bhakti Raul. Color image segmentation using vector quantization techniques based on the energy ordering concept. *International Journal of Computing Science and Communication Technologies*, 1:2, 2009. 3
- [24] Muhammad Hassan Khan, Muhammad Shahid Farid, and Marcin Grzegorzec. A comprehensive study on codebook-based feature fusion for gait recognition. *Information Fusion*, 92:216–230, 2023. 1, 3
- [25] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 2, 5, 3
- [26] Zhibin Lan, Jiawei Yu, Xiang Li, Wen Zhang, Jian Luan, Bin Wang, Degen Huang, and Jinsong Su. Exploring better

- text image translation with multimodal codebook. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, page 3479–3491, 2023. 3
- [27] 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. 5, 3
- [28] 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*, pages 11523–11532, 2022. 1, 3, 5, 6
- [29] Lei Li, Tingting Liu, Chengyu Wang, Minghui Qiu, Cen Chen, Ming Gao, and Aoying Zhou. Resizing codebook of vector quantization without retraining. *Multimedia Systems*, pages 1–14, 2023. 1, 3
- [30] Lingzhi Li, Zhongshu Wang, Zhen Shen, Li Shen, and Ping Tan. Compact real-time radiance fields with neural codebook. *arXiv preprint arXiv:2305.18163*, 2023. 3
- [31] Yue Li, Wenrui Ding, Chunlei Liu, Baochang Zhang, and Guodong Guo. Trq: Ternary neural networks with residual quantization. In *Proceedings of the AAAI conference on artificial intelligence*, pages 8538–8546, 2021. 3
- [32] Yoseph Linde, Andres Buzo, and Robert Gray. An algorithm for vector quantizer design. *IEEE Transactions on communications*, 28(1):84–95, 1980. 1, 2
- [33] Kechun Liu, Yitong Jiang, Inchang Choi, and Jinwei Gu. Learning image-adaptive codebooks for class-agnostic image restoration. *arXiv preprint arXiv:2306.06513*, 2023. 1, 3
- [34] Qi Mao, Tinghan Yang, Yinuo Zhang, Shuyin Pan, Meng Wang, Shiqi Wang, and Siwei Ma. Extreme image compression using fine-tuned vqgan models. *arXiv preprint arXiv:2307.08265*, 2023. 3
- [35] Julieta Martinez, Holger H Hoos, and James J Little. Stacked quantizers for compositional vector compression. *arXiv preprint arXiv:1411.2173*, 2014. 3
- [36] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations*, 2016. 2
- [37] B. Ramamurthi and A. Gersho. Classified vector quantization of images. *IEEE Transactions on Communications*, 34(11):1105–1115, 1986. 2
- [38] Arturo Ribes, Senshan Ji, Arnau Ramisa, and Ramon Lopez de Mantaras. Self-supervised clustering for codebook construction: An application to object localization. In *Artificial Intelligence Research and Development - Proceedings of the 14th International Conference of the Catalan Association for Artificial Intelligence*, pages 208–217. IOS Press, 2011. 1, 3
- [39] 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*, pages 10684–10695, 2022. 5, 8, 2
- [40] 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*, pages 20987–21012. PMLR, 2022. 1, 3, 5, 6
- [41] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017. 1, 2, 3, 5, 6
- [42] Vivek Venugopal, Surbhi Pillai, and Suresh Sundaram. A hierarchical codebook descriptor approach for online writer identification. In *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 398–403. IEEE, 2018. 1, 3
- [43] Matthew Wallingford, Aditya Kusupati, Alex Fang, Vivek Ramanujan, Aniruddha Kembhavi, Roozbeh Mottaghi, and Ali Farhadi. Neural radiance field codebooks. In *International Conference on Learning Representations*, 2023. 3
- [44] Will Williams, Sam Ringer, Tom Ash, David MacLeod, Jamie Dougherty, and John Hughes. Hierarchical quantized autoencoders. *Advances in Neural Information Processing Systems*, 33:4524–4535, 2020. 1, 3, 5, 6
- [45] Rui-Qi Wu, Zheng-Peng Duan, Chun-Le Guo, Zhi Chai, and Chongyi Li. Ridcp: Revitalizing real image dehazing via high-quality codebook priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22282–22291, 2023. 1, 3
- [46] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017. 5, 3
- [47] Peng Ye and David Doermann. No-reference image quality assessment using visual codebooks. *IEEE Transactions on Image Processing*, 21(7):3129–3138, 2012. 3
- [48] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, 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. 5, 8, 4
- [49] 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 *International Conference on Learning Representations*, 2022. 1, 3, 5, 6
- [50] 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*, pages 586–595, 2018. 5
- [51] Wei Zhang, Akshat Surve, Xiaoli Fern, and Thomas Dietterich. Learning non-redundant codebooks for classifying complex objects. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1241–1248, 2009. 1, 3
- [52] Tianchen Zhao, Niansong Zhang, Xuefei Ning, He Wang, Li Yi, and Yu Wang. Codedvtr: Codebook-based sparse voxel transformer with geometric guidance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1435–1444, 2022. 2, 3

- [53] Chuanxia Zheng and Andrea Vedaldi. Online clustered codebook. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22798–22807, 2023. [1](#), [3](#), [5](#), [6](#), [7](#), [8](#), [2](#)
- [54] Chuanxia Zheng, Tung-Long Vuong, Jianfei Cai, and Dinh Phung. Movq: Modulating quantized vectors for high-fidelity image generation. *Advances in Neural Information Processing Systems*, 35:23412–23425, 2022. [1](#), [3](#), [5](#), [6](#)
- [55] Sipeng Zheng, Bohan Zhou, Yicheng Feng, Ye Wang, and Zongqing Lu. Unicode: Learning a unified codebook for multimodal large language models. In *European Conference on Computer Vision*, pages 426–443. Springer, 2025. [1](#)
- [56] Hongliang Zhong, Jingbo Zhang, and Jing Liao. Vq-nerf: Neural reflectance decomposition and editing with vector quantization. *arXiv preprint arXiv:2310.11864*, 2023. [3](#)
- [57] Lei Zhu, Fangyun Wei, Yanye Lu, and Dong Chen. Scaling the codebook size of vq-gan to 100,000 with a utilization rate of 99%. In *Advances in Neural Information Processing Systems*, pages 12612–12635. Curran Associates, Inc., 2024. [5](#)
- [58] Xiaosu Zhu, Jingkuan Song, Lianli Gao, Feng Zheng, and Heng Tao Shen. Unified multivariate gaussian mixture for efficient neural image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17612–17621, 2022. [1](#), [3](#)
- [59] Xiaosu Zhu, Jingkuan Song, Lianli Gao, Xiaoyan Gu, and Heng Tao Shen. Revisiting multi-codebook quantization. *IEEE Transactions on Image Processing*, 2023. [3](#)
- [60] Wenbin Zou, Hongxia Gao, Tian Ye, Liang Chen, Weipeng Yang, Shasha Huang, Hongsheng Chen, and Sixiang Chen. Vqcnir: Clearer night image restoration with vector-quantized codebook. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 7873–7881, 2024. [1](#)

LooC: Effective Low-Dimensional Codebook for Compositional Vector Quantization

— Supplementary Material —

1. Storage and Computational Efficiency

Sec. 3.3 shows that our codebook theoretically requires only $K' = K^{\frac{1}{m}}$ codevectors to achieve the same capacity as an ordinary VQ with K codevectors. Particularly, when $m > 1$, K' is significantly smaller than K . Here, we delve deeper into analyzing the storage and computing costs associated with LooC. It shows that LooC provides greater capacity while consuming less space and computation than traditional VQ, making LooC a much more efficient alternative.

1.1. Storage Efficiency

Storage Cost of Codebook. LooC requires fewer codevectors (K') and lower dimension (d) compared to traditional VQ, resulting in less storage cost. When using a codebook containing K codevectors of dimension d to perform a VQ operation on the feature map $z \in \mathbb{R}^{h \times w \times d}$, the codebook needs to store $K \times d$ values, usually in 32-bit floating point format. Therefore, a total of $S_{\text{codebook}} = 32 \times K \times d$ bits of storage is required. However, in our LooC, there are $K' \times d^* = K' \times d/m$ values to be stored. In theory, this only requires $S'_{\text{codebook}} = 32 \times K^{\frac{1}{m}} \times d/m$ bits of storage when $K' = K^{\frac{1}{m}}$. As the K' value increases, our method can achieve larger capacity while consuming less storage than traditional methods.

Storage Cost of Indices. When performing the VQ operation on the codebook of K codevectors, the $h \times w$ indices are needed to store the corresponding relationship of quantized matching. Each index requires $\log_2 K$ bits of storage. Therefore, the total required storage is $S_{\text{index}} = h \times w \times \log_2 K$. In our LooC method, since each feature vector is divided into m component units, $h \times w \times m$ indices are required, occupying $S'_{\text{index}} = h \times w \times m \times \log_2 K'$ bits of storage. It is worth noting that $K' < K$. When $K' = K^{\frac{1}{m}}$, S'_{index} is equivalent to S_{index} .

1.2. Computational Efficiency

To determine the most similar matches between the K codevectors in the codebook and the feature representation $z \in \mathbb{R}^{h \times w \times d}$, a total of $h \times w \times K$ similarity calculations are needed. Here, we use the widely adopted cosine similarity for matching purposes. Consequently, in conventional VQ, $h \times w \times K \times d$ multiplication operations are needed, while omitting the addition operations. However, in our LooC with K' codevectors, the feature representation z is decomposed into m segments, leading to the need for $h \times w \times m \times K' \times d^*$ multiplication operations. Here, $d^* = d/m$. Therefore, it requires $h \times w \times m \times K' \times (d/m) = h \times w \times K' \times d$

multiplication operations. Since K' is much smaller than K , the computational cost of our method is also much smaller than that of conventional VQ.

2. Codebook Usage

2.1. Usage of Codevectors

Overall Usage. In Tab. 2, our method shows a remarkable overall usage of 100% of the codebook. In this section, we employ tSNE for visualization purposes to gain insights into the learned codebooks of both VQ-VAE and our proposed LooC-VAE. Fig. 5(a) showcases the codebook visualization of VQ-VAE, where it becomes apparent that numerous codevectors remain unused. However, in the case of our LooC-VAE, as depicted in Fig. 5(b) and (c), the usage rate reaches 100% when employing codebook sizes of 1024 and 256, respectively. This demonstrates the effectiveness of our approach in fully utilizing the available codebook capacity.

Codebook usage without CVQ’s update method To further investigate this, we conduct experiments by removing the CVQ update strategy in LooC and counting the codebook usage. We calculate the usage on large dataset FFHQ [22]. The results are shown in Tab. 7. Our findings indicate that the codebook usage is 100% at both $K = 256$ and $K = 1024$. This clearly reveals that LooC’s strength on codebook usage is not simply stemmed from the CVQ update strategy.

Per-segment Usage. As we divide the feature map into m segments in the compositional VQ, we analyze each segment’s codebook usage in this study. We take CIFAR10

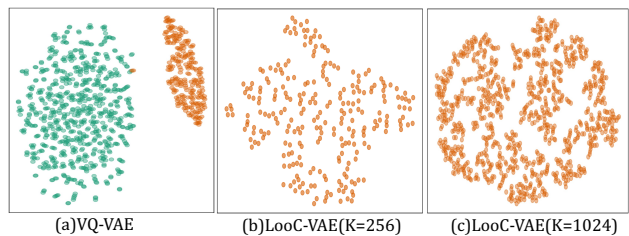


Figure 5. **Codebook visualization with t-SNE** for models trained on CIFAR10 and evaluated on the validation set. VQ-VAE has unused codevectors (green points) with only 24.12% usage. LooC achieves 100% usage at both $K = 256$ and $K = 1024$.

Method	dataset	$K \times d^* \downarrow$	usage
LooC-VAE	FFHQ	256×4	100%
LooC-VAE		1024×4	100%

Table 7. **Usage of codevectors** of our LooC without CVQ’s codebook update strategy.

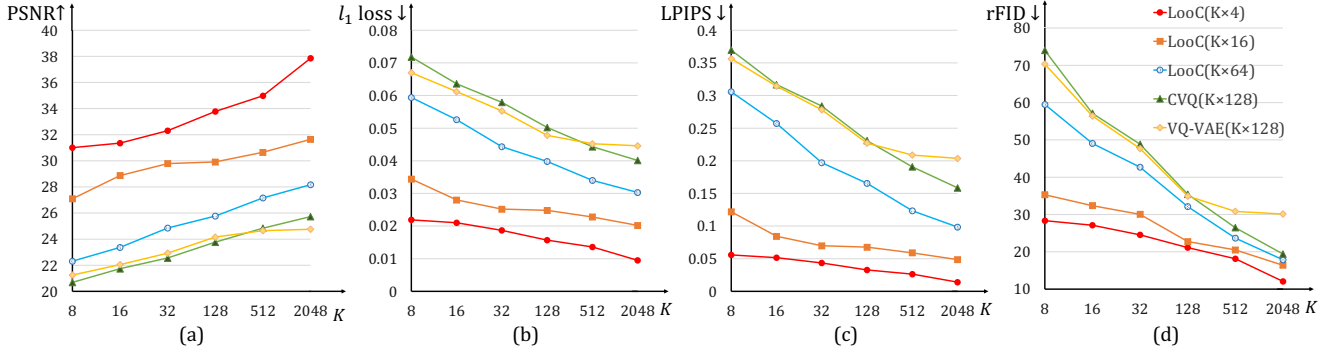


Figure 6. **Reconstruction results** of LooC and other SOTA methods on CIFAR10 [25] with various numbers of codevectors demonstrate LooC’s superior performance with a smaller codebook, showcasing its flexibility and efficiency. Reducing the codevector dimension d^* , *i.e.*, increasing the value of $m = d/d^*$, in LooC leads to a more detailed combinational quantization and improved performance, especially for small values of K .

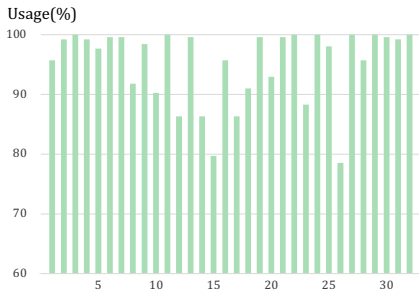


Figure 7. **Usage** of the codevectors for each segments. The experiment is conducted on the CIFAR10 dataset with $K = 256$ and $m = 32$. A high per-segment usage rate of codevectors also suggests a considerable need for codevector sharing between different segments.

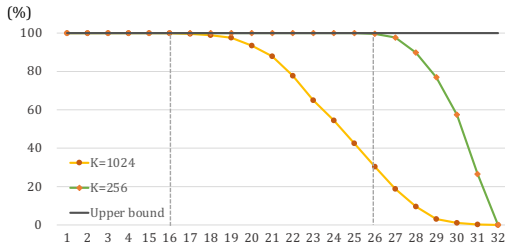


Figure 8. **(a)** With $K = 256$, codevectors are shared among at least 26 out of 32 segments. **(b)** With $K = 1024$, codevectors are shared among at least 16 segments. Smaller K values result in higher codevector sharing rates. Larger K values reduce the need for a high sharing rate.

as the dataset in our analysis and consider the case with $K = 256$ and $m = 32$, as shown in Tab. 3. When considering the codebook usage across all 32 segments collectively, the overall usage remained 100%. Fig. 7 shows the per-segment usage of codevectors. The horizontal axis corresponds to the segment index, while the vertical axis represents the codebook usage counted on each segment. Upon closer examination of the individual segments, we discover that except for 6 segments, the usages on the remaining segments exceed 90%.

2.2. Codevector Sharing

In our previous usage analysis, the high codebook usage of each segment indicated significant codevector sharing among different segments. This higher usage is directly correlated with a higher sharing degree. To further explore the impact of different K values on codevector sharing, we analyze with $K = 256$ and $K = 1024$, using $m = 32$ segments on CIFAR10 dataset. We analyze the percentage of codevectors that are shared by a minimum of $n \in \{1, 2, \dots, 32\}$ segments. Fig. 8(a) shows that when $K = 256$, all codevectors are shared among at least 26 out of 32 segments, most of which are shared among at least 30 segments. Fig. 8(b) reveals that with $K = 1024$, all codevectors are shared among at least 16 segments, most of which are shared among at least 20. A smaller K value results in a higher codevector sharing rate between different segments. This finding explains why our method performs better with a smaller K . Conversely, and a larger K value diminishes the urgency for pursuing a high sharing rate.

3. Implementation Details

In the image reconstruction task, we compare LooC with various VQ modules. To ensure a fair comparison, we integrate LooC into VQ-VAE’s network structure by replacing the VQ module, following CVQ [53]. We also reimplement the quantizers in PQ [19] and, like other approaches, apply the VQ-VAE structure for training. Afterwards, we assess the generalizability of our LooC method on larger datasets by employing the VQ-GAN [7] architecture. Our experiments utilize the same encoder and decoder as CVQ-VAE [53], both based on convolutional neural networks. Our model was trained using a single RTX3090 GPU, and the settings for optimizer, batch size, learning rate, and number of epochs are consistent with CVQ.

For image generation task, we use the LDM framework [39] and replace VQ module with LooC and other comparative methods. Similar to the baseline LDM, we

Method		$K \times d^* \downarrow$	l_1 loss \downarrow	LPIPS \downarrow	rFID \downarrow	SSIM \uparrow	PSNR \uparrow
VQ-VAE [41]	MNIST	1024×128	0.0207	0.0282	3.43	0.9777	26.48
HVQ-VAE [44]		1024×128	0.0202	0.0270	3.17	0.9790	26.90
SQ-VAE [40]		1024×128	0.0197	0.0256	3.05	0.9819	27.49
CVQ-VAE [53]		1024×128	0.0180	0.0222	1.80	0.9833	27.87
LooC(32×4)		32×4	0.0082	0.0083	1.70	0.9961	35.15
LooC(256×4)		256×4	0.0062	0.0058	1.31	0.9976	37.58
VQ-VAE [41]	CIFAR10	1024×128	0.0527	0.2504	39.67	0.8595	23.32
HVQ-VAE [44]		1024×128	0.0533	0.2553	41.08	0.8553	23.22
SQ-VAE [40]		1024×128	0.0482	0.2333	37.92	0.8779	24.07
CVQ-VAE [53]		1024×128	0.0448	0.1883	24.73	0.8978	24.72
LooC-VAE		32×4	0.0189	0.0435	24.53	0.9805	32.22
LooC-VAE		256×4	0.0144	0.0285	19.22	0.9880	34.51
VQ-VAE [41]	FASHION-MNIST	1024×128	0.0377	-	12.73	-	23.93
CVQ-VAE [53]		1024×128	0.0344	-	8.85	-	24.66
LooC-VAE		32×4	0.0145	0.0158	8.7310	0.9864	32.4850
LooC-VAE		256×4	0.0103	0.0098	6.2368	0.9924	35.3393

Table 8. **Reconstruction results** on the validation sets of MNIST [27], CIFAR10 [25], and FASHION-MNIST [46]. Our approach outperforms other SOTA methods and maintains comparable results even with significant reductions in the codebook size.

Method		$K \times d^* \downarrow$	Usage \uparrow	LPIPS \downarrow	rFID \downarrow	SSIM \uparrow	PSNR \uparrow
VQGAN [7]	FFHQ	1024×256	42%	0.1175	4.42	0.6641	22.24
ViT-VQGAN [49]		8192×32	-	-	3.13	-	-
RQ-VAE [28]		2048×256	-	0.1302	3.88	0.6700	22.99
MoVQ [54]		1024×64	56%	0.0585	2.26	0.8212	26.72
SeQ-GAN [13]		1024×256	100%	-	3.12	-	-
CVQ-VAE [53]		1024×256	100%	0.0533	2.03	0.8398	26.87
LooC-VAE		256×4	100%	0.0501	1.97	0.8499	27.73
LooC-VAE		1024×4	100%	0.0346	1.37	0.9276	32.44
VQGAN [7]	ImageNet	1024×256	44%	0.2011	7.94	0.5183	19.07
ViT-VQGAN [49]		8192×32	96%	-	1.28	-	-
RQ-VAE [28]		16384×256	-	-	1.83	-	-
MoVQ [54]		1024×64	63%	0.1132	1.12	0.6731	22.42
SeQ-GAN [13]		1024×256	100%	-	1.99	-	-
CVQ-VAE [53]		1024×256	100%	0.1099	1.57	0.7115	23.37
LooC-VAE		256×4	100%	0.0916	1.68	0.7233	23.64
LooC-VAE		1024×4	100%	0.7160	1.01	0.7160	29.15

Table 9. **Reconstruction results** on FFHQ [22] and ImageNet [4]. Our approach surpasses other SOTA methods and delivers comparable results even with significant reductions in the codebook size.

generate our results at a resolution of 256×256 and utilize the same training parameters. We also adopt the same $16 \times$ downsampling scales in the latent representations as LDM, except for utilizing different quantizers.

4. More Experimental Results

4.1. Quantitative Results of Reconstruction

In Fig. 6, we compare our results with the various latest quantizers on CIFAR10 [25]. We vary the number of codevectors, denoted as K , and extend Fig. 1-Right by including evaluation scores for four additional metrics: l_1 loss, LPIPS, rFID, and PSNR. This experiment shows that our method effectively utilizes the minimum number of codevectors to fully exploit the advantages of compositional VQ, thereby

improving the effectiveness of VQ. In contrast, other state-of-the-art (SOTA) methods rely heavily on large codebooks.

Tab. 8 presents a comprehensive comparison between our method and the SOTA methods on three datasets: MNIST [27], CIFAR10 [25], and FASHION-MNIST [46]. This table serves as an extension of Tab. 1. Our method showcases remarkable performance by achieving similar effects on rFID using a smaller codebook size of 32×4 , compared to the SOTA method that requires a larger codebook of 1024×128 . Notably, our method outperforms the SOTA method in terms of l_1 loss, LPIPS, SSIM and PSNR, indicating superior performance. Furthermore, when our method utilizes a codebook size of 1024×4 , we observe even more impressive results across various indicators.

Method	dataset	$K \times d^* \downarrow$	rFID \downarrow	SSIM \uparrow	PSNR \uparrow
PQ [19]	FFHQ	$16 \times 4 \times \#64$	2.43	0.8054	25.55
LooC-VAE		1024×4	1.37	0.9276	32.44
PQ [19]	ImageNet	$16 \times 4 \times \#64$	1.96	0.6701	21.73
LooC-VAE		1024×4	1.01	0.7160	29.15

Table 10. **Image reconstruction** results of PQ [19] and LooC on high-resolution datasets of FFHQ [22] and ImageNet [4].

According to the information provided, Tab. 9 serves as an extension of Tab. 2, with added results for the LPIPS metric on FFHQ and ImageNet datasets. The results show that our method outperforms the previous SOTA method in all indicators. Additionally, our method utilizes a smaller codebook size, specifically only one 256th of that of CVQ-VAE.

In Tab. 10, we provide quantitative evaluation results of PQ [19] on the FFHQ and ImageNet datasets. LooC exhibits clear superiority over PQ with regards to processing high-resolution images. From Tab. 10 and the results of other comparison methods in Tab. 9, we can see that PQ outperforms many other methods, while LooC shows clearly superior performance than PQ. Furthermore, we believe that a unified codebook is a more succinct and plausible solution.

Method	dataset	$K \times d^* \downarrow$	FID \downarrow
LooC-VAE	ImageNet-cls-avg	1024×4	46.78
LooC-VAE	LSUN-churches	1024×4	15.17
LooC-VAE	LSUN-bedroom	1024×4	17.52

Table 11. **Comparison of FID scores for class-conditional synthesis** on ImageNet [4] and LSUN [48]. The FID score for each class in ImageNet is computed individually and then averaged for all classes.

4.2. Quantitative Results of Generation

Apart from the visualization results in Fig. 4, we provide the comparison of FID metrics for class-conditional synthesis on ImageNet and LSUN in Tab. 11. The FID on ImageNet is 46.78, and the FIDs on LSUN-churches and LSUN-bedroom are 15.17 and 17.52 respectively. Note that the FID score for each class in ImageNet is computed individually and then averaged for all classes.

5. Generalization Ability

To further investigate the generalization ability of our method, we conduct experiments by training the reconstruction model with our LooC plugged in on FFHQ and testing it on the CelebA dataset. Tab. 12 and Fig. 9 showcase the quantitative and qualitative results respectively. The results demonstrate the effectiveness and strong generalization ability of our method. Despite being trained only on the FFHQ dataset, our method has achieved an rFID of 5.66 with $K = 256$ on the CelebA validation set. This is a notable improvement over VQGAN [7], which has an rFID of 10.2 with $K = 400$. Furthermore, when LooC uses $K = 1024$, the rFID is further improved to 3.86. The visualizations in

Method	$K \times d^* \downarrow$	rFID \downarrow	SSIM \uparrow	PSNR \uparrow
LooC-VAE	256×4	5.66	0.8141	26.36
LooC-VAE		3.86	0.8234	26.73

Table 12. **Generalization ability.** Image reconstruction results of LooC which is trained on FFHQ [22] and tested on CelebA.

Fig. 9 illustrate that our approach can produce intricate and high-quality reconstructions.

6. Further Discussion

Research Vision Current codebooks in VQ are dataset-specific, the same for LooC. A more efficient solution is a universally applicable codebook shared across different datasets and different types of data. Exploring a cross-dataset universal codebook is a promising and valuable future research direction. As the diversity of cross-dataset data increases, it also imposes higher requirements on the capacity of the codebook. Therefore, a more compact codebook design with even higher capacity remains an intriguing topic to study, LooC serves as a cornerstone for future exploration in this direction.

Other Impact The aim of this paper is to investigate a more efficient and compact method for representing visual data. Our approach helps to lower storage and transmission expenses, facilitating data exchange and sharing in our daily life. Meanwhile, it can expedite scientific research and foster technological innovation. Additionally, our approach involves dividing features into sub-segments, rather than treating them as separate and complete features. This compositional nature not only strengthens data security and privacy protection but also reduces the risk of data leakage, ultimately safeguarding the data assets of individuals and organizations.



Figure 9. Visualization of image reconstruction of LooC, trained on FFHQ and tested on CelebA