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

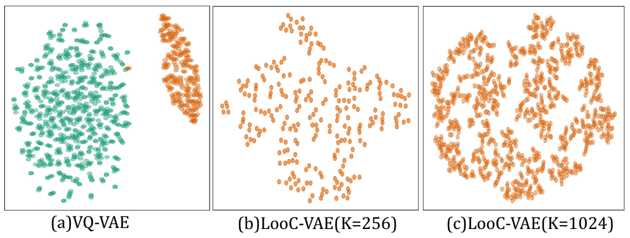

Figure 5. **Codebook visualization with t-SNE** for models trained on CIFAR10 and evaluated on the validation set. VQ-VAE has unused codevectors (green points) with only 24.12% useage. LooC achieves 100% usage at both $K = 256$ and $K = 1024$.

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.

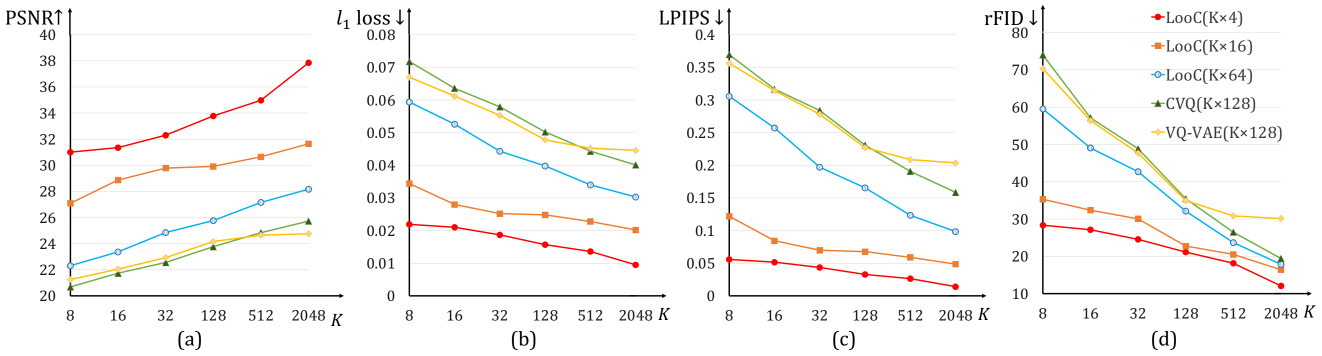

Figure 6. **Reconstruction results** of LooC and other SOTA methods on CIFAR10 [25] with various numbers of codevectors demonstrate LooC's superior performance with a smaller codebook, showcasing its flexibility and efficiency. Reducing the codevector dimension d^*, *i.e.*, increasing the value of $m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

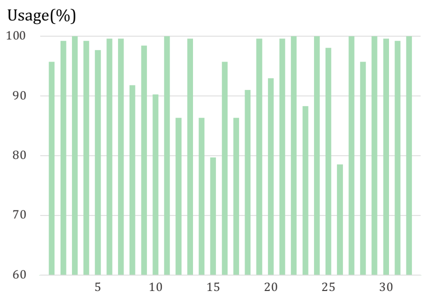

Figure 7. **Usage** of the codevectors for each segments. The experiment is conducted on the CIFAR10 dataset with $K = 256$ and $m = 32$. A high per-segment usage rate of codevectors also suggests a considerable need for codevector sharing between different segments.

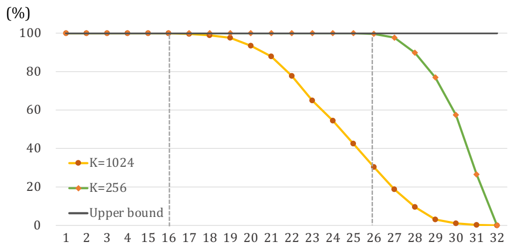

Figure 8. **(a)** With $K = 256$, codevectors are shared among at least 26 out of 32 segments. **(b)** With $K = 1024$, codevectors are shared among at least 16 segments.Smaller K values result in higher codevector sharing rates. Larger K values reduce the need for a high sharing rate.

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 accross 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 $n \in \{1, 2, \cdots, 32\}$ 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↓	SSIM↑	PSNR↑
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↓
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 Fig. 9 illustrate that our approach can produce intricate and high-quality reconstructions.

Method	$K \times d^* \downarrow$	rFID↓	SSIM↑	PSNR↑
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.

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.

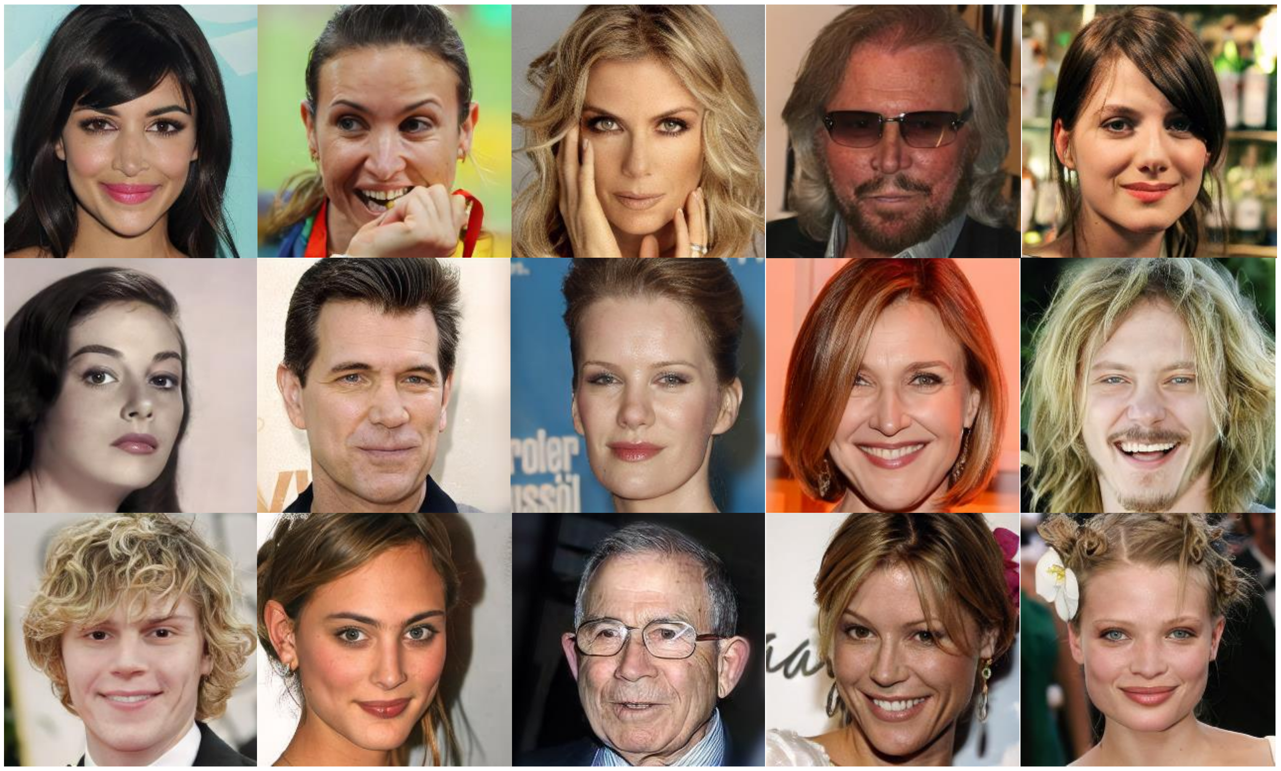
Figure 9. **Visualization of image reconstruction** of LooC, trained on FFHQ and tested on CelebA