Transporting Quantiles to Codebook: Scalable Vector Quantization without Codebook Collapse

Yuwei Zeng[1] **Zekun Shi**[1,2]

[1] National University of Singapore [2] SEA AI Lab

Abstract

Effective discrete representation learning with VQ-VAE serves as a fundamental component of modern autoregressive generative models. However, it is widely known to suffer from codebook collapse and inefficient code utilization. To address this issue, we propose FlowVQ, a simple yet effective alternative to standard vector quantization by formulating neural quantization as a joint modeling and clustering process. Specifically, we learn the latent space using a lightweight flow matching model alongside training, corresponding to a probability flow ODE that transports an often fixed well-behaved source distribution while preserving probability mass. Based on this, we define the codebook $\mathcal{C}_1$ by transporting even-quantile partitioned prior codes $\mathcal{C}_0$ from the source space through the learned flow, and obtaining the Voronoi centroids under the induced distribution, leading to a structured discretization. We evaluated on image tokenization and generation, as well as on two tasks requiring high-precision tokenization, demonstrating scalability and improved expressivity of the codebook with full utilization. Code is available at: GitHub repository.

1 Introduction

Vector Quantized Variational AutoEncoder (VQ-VAE), proposed by Van Den Oord et al. [1], [2] demonstrates effective discrete latent representation learning by incorporating Vector Quantization (VQ) into reconstruction. It learns a finite set of codebook vectors and tokenizes continuous data into discrete sequences by mapping to the nearest vectors, enabling large autoregressive models for continuous data generation while leveraging the design, training, and scaling principles behind large language models. By balancing fidelity, expressiveness, and computational efficiency, VQ-VAE remains an essential building block of modern generative models that are adopted across diverse domains, including state-of-the-art models for image generation and video-based world modeling.

With codebook collapse and unscalable model capacity remaining unresolved, existing works have explored numerous ways to mitigate such issues. Early improvements [3–5] involved various training techniques such as random restart, random replacement, and k-means initialization. Williams et al. [6] and Takida et al. [7, 8] proposed stochastic quantization, Zhu et al. [9] and Chang et al. [10] proposed adding an additional projector layer or bridging network after a frozen codebook. With an altered latent space, model performance can be compromised, leading to degenerate reconstruction and representation learning. Recently, Lookup-Free Quantization (LFQ) [11–13] methods proposed an alternative to VQ based on direct scalar quantization with fixed hypercube geometry, rounding values in each dimension to the nearest point, and showed scalable codebook capacity with improved image generation using large vocabulary sizes beyond 2^{14}.

To circumvent codebook collapse, we propose that an ideal codebook should jointly model and cluster the latent space into even partitions alongside reconstruction training. The codebook vectors correspond to the Voronoi centers of these partitions, naturally allocating denser vectors to high-

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

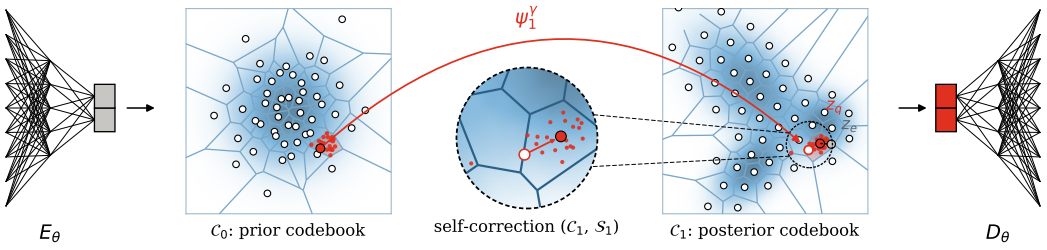

Figure 1: FlowVQ on a three-mode 2D latent distribution (K = 50): the learned flow ψ_1^γ transports the approximately equal-mass quantile prior codebook $\mathcal{C}_0 \in \mathcal{N}(\mathbf{0}, I)$, initialized using Sobol samples here and kept fixed thereafter, to the latent space. Self-correction refines each transported code using the pushed quantile samples $\mathcal{S}_1$, yielding the density-adaptive posterior codebook $\mathcal{C}_1$.

density regions and sparser vectors to low-density regions, regardless of the manifold geometry. This ensures full and balanced utilization, addressing expressiveness and scalability. Based on this principle, we propose a flow-based vector quantization approach without interfering with the representation learning objective.

A flow matching model [14] learns a continuous transport mapping samples from a prior distribution, such as a Gaussian, to an arbitrary data distribution. This transport corresponds to a Probability Flow ODE (PF-ODE) [15, 16], which preserves probability mass and transports prior quantiles to data quantiles. Based on this, we define the prior codebook uniformly in the well-behaved source space, and co-train the flow model with VQ-VAE. The codebook vectors are then obtained as the pushforward of the prior codebook and move toward the Voronoi centers under the estimated density. Experiments show that a lightweight flow model introduces minor overhead relative to VQ-VAE while maintaining full code utilization and improving performance.

In summary, our contributions are as follows:

1. We propose FlowVQ, a novel VQ technique that converts codebook learning into a quantile transport problem, where code utilization emerges as a geometric property of the latent space, enabling balanced usage and avoiding codebook collapse.
2. We show that FlowVQ consistently achieves full codebook utilization across a range of tasks, improving reconstruction and generation quality within a unified tokenizer framework.

2 Related Works

VQ-VAE & Generation in the Latent Space Proposed by [1], the VQ-VAE combines vector quantization with a reconstruction objective and achieves effective neural tokenization. It enables autoregressive generation on continuous data, such as high-fidelity image generation, by predicting the sequence of tokens in the latent space using a PixelCNN [1, 17] and, more recently transformer models. VQ-GAN [18] further couples the training with a discriminative loss, addressing the issue of blurry reconstruction from traditional VAE greatly improving generation fidelity. It has since been adopted as a standard framework. Recently, advancements on residual [19], multi-head [20], and multi-scale [21] tokenization were proposed and demonstrate solid improvements [19, 21, 22]. As a general tokenization technique, VQ-VAE based tokenizers are widely adopted in diverse generative tasks today, especially in tasks where direct generation can be prohibitively expensive such as image [12, 21, 23], audio [3, 24], and video [25, 26], or in fields where structural representations are often unavailable, such as AI for protein and material generation [27–29].

Mitigating Codebook Collapse To address the codebook collapse, which leads to unscalable codebook capacity, an active line of work has been proposed to mitigate it. Early work addressed it with various tricks such as random restart or random replacement [3, 5]. Another line of work explored differentiable relaxation [4, 30, 31], most notably using the Gumbel–Softmax [32] . However, these approaches generally underperform and produces lower-quality reconstruction [6]. Takida et

al. [7, 8] proposed improving the mode-dropping behavior by switching to stochastic quantization but suffer from similar unrealistic reconstructions. Zhu et al. [9] and Chang et al. [10] addressed the under-utilization issue by adding intermediate projection layer(s) between a frozen codebook and the decoder. In a parallel track, Mentzer et al. [11], Yu et al. [12], and Weber et al. [13] proposed LFQ, which rounds each dimension to the nearest point in a hypercube, and scales the codebook capacity with its size using direct scalar quantization. These methods demonstrate competitive or improved image generation performance, especially with large vocabulary sizes beyond 2^{14} to 2^{18}.

The majority of the existing works mitigate this issue without fundamentally solving it. To our knowledge, the closest work is OptVQ [22], which introduces global information via optimal transport between batch latents and the codebook. However, its cost matrix requires per-batch normalization for numerical stability, which distorts the transport cost inconsistently across batches. Moreover, the resulting doubly stochastic assignment is converted to a hard assignment, which does not guarantee that every code is selected, especially under stronger entropy regularization. Full utilization thus depends on the entropy weight, which trades off quantization accuracy against utilization and may be sub-optimal for a given task as observed in our experiments.

3 Method

3.1 Preliminary

We define a codebook $\mathcal{C} = \{\mathbf{C}_i\}_{i=1}^K$ where $\mathbf{C}_i \in \mathbb{R}^d$, K is the number of code vectors and d is the latent vector dimension. Given an input x, the original latent vector is $z_e = E_\theta(x)$, and the quantized vector z_q is defined as:

$$z_q = \mathbf{C}_i \quad \text{where} \quad i = \arg\min_i ||\mathbf{C}_i - z_e||_2 \tag{1}$$

With sg() indicating the stop gradient operator, the original VQ-VAE trains with the following objective function. β is a training dynamics dependent coefficient that controls the expansion rate of z_e driven by the reconstruction objective, while ensuring the effectiveness of the STE operator in the reconstruction loss:

$$\mathcal{L}_{\text{VQ-VAE}}(\theta, \mathcal{C}) = \underbrace{||D_\theta(\text{STE}(z_q)) - x||_2^2}_{\text{reconstruction loss}} + \underbrace{||\text{sg}(z_e) - z_q||_2^2}_{\text{codebook loss}} + \beta \underbrace{||z_e - \text{sg}(z_q)||_2^2}_{\text{commitment loss}} \tag{2}$$

$$\text{where} \quad \text{STE}(z_q) = z_e + \text{sg}(z_q - z_e)$$

3.2 Flow-based Codebook

In this work, we propose to define the codebook in the prior space, which we call the prior codebook $\mathcal{C}_0$. The prior space can be arbitrary, as long as it supports uniform sampling. To achieve uniform sampling, we partition the prior space into K equal probability mass quantiles using its cumulative distribution function (CDF) $\mathcal{F}$. $\{\mathbf{C}_0^i\}_{i=1}^K$ is then obtained by applying the inverse CDF $\mathcal{F}^{-1}$ to the centers of these quantiles:

$$\mathbf{C}_0^i = \mathcal{F}^{-1}\left(\frac{i - 0.5}{K}\right), \quad i = 1, ..., K$$

In higher dimensional latent spaces, as d increases, exact uniform quantile partitioning becomes intractable. We use Sobol sequences, a low-discrepancy quasi-random sequence designed to evenly fill multi-dimensional spaces to generate approximate uniformly distributed quantile centers. Specifically, we first generate K points in the CDF as d-dimensional unit hypercube $[0, 1]^d$:

$$\{\mathbf{u}_i\}_{i=1}^K = \text{Sobol}(K, d) \quad \text{where} \quad \mathbf{u}_i \in [0, 1]^d \tag{3}$$

Then obtain the prior codebook for each quantile similarly

$$\mathbf{C}_0^i = \mathcal{F}^{-1}(\mathbf{u}_i) \tag{4}$$

We define the learned quantizer flow velocity field as u_t^γ, and the associated flow map as ψ_t^γ. Let the data distribution be $x \sim p_{\text{data}}$. During VQ-VAE training, it is trained to match the current latent distribution over z_e with the conditional flow matching objective:

$$\mathcal{L}_{\text{FM}}(\gamma) = \mathbb{E}_{t \sim \mathcal{U}[0,1],\ z_0 \sim \mathcal{N}(0, I_d),\ x \sim p_{\text{data}}} ||u_t^\gamma(z) - u_t(z \mid z_1, z_0)||_2^2 \tag{5}$$

$$\text{where} \quad \frac{d}{dt}\psi_t^\gamma(z) = u_t^\gamma(\psi_t^\gamma(z)), \quad z_1 = E_\theta(x), \quad \psi_t(z \mid z_1, z_0) = (1-t)z_0 + tz_1$$

With latent distribution modeling and clustering handled by a separate flow model, the explicit codebook loss term in Eq (2) is no longer required. The flow matching objective is optimized in a decoupled manner and should introduce minimal interference to the reconstruction and quantization objectives. The overall training objective is then updated to:

$$\begin{aligned}\mathcal{L}_{\text{FlowVQ}}(\theta, \gamma) &= \mathcal{L}_{\text{VQ-VAE}}(\theta) \; + \; \mathcal{L}_{\text{FM}}(\gamma) \\ &= ||D_\theta(\text{STE}(z_q)) - x||^2 \; + \; \beta||z_e - \text{sg}(z_q)||^2 \; + \; \mathbb{E}_{t,z_0,z_1}||u_t^\gamma(z) - u_t(z \mid z_1, z_0)||^2\end{aligned} \tag{6}$$

The final codebook $\mathcal{C}_1$ used in vector quantization is periodically updated. Each code vector $\mathbf{C}_1^i$ is obtained as the pushforward of the prior codebook $\mathbf{C}_0^i$ through the trained flow ψ_t^γ:

$$\mathbf{C}_1^i = \psi_1^\gamma(\mathbf{C}_0^i), \quad i = 1, ..., K \tag{7}$$

The codebook structure closely follows the geometry of the latent manifold and adapts to its density, naturally providing sharper resolution in denser regions and coarser in sparser regions. This is in contrast to the original VQ, which is sensitive to both initialization and training dynamics due to its reliance on both codebook and commitment losses between only matched pairs, which are a root cause of codebook collapse. Similarly, fixed-geometry approaches such as LFQ do not adapt to the latent distribution, including empty regions beyond the latent boundaries and holes within it, which limits their expressivity.

3.3 Self-Correction toward Voronoi Centroids

Up to this point, a set of approximately uniform points in the latent space is obtained through Eq (7) as the codebook $\mathcal{C}_1$, which only fulfills the first two key properties of faithful latent modeling and approximately even partitioning. To minimize the approximation error arising from the quasi-random sequence and the non-linear flow mapping, we introduce a self-correction mechanism that updates $\mathcal{C}_1$ toward the Voronoi centroids as the Euclidean centers of the transported quantiles, aiming to achieve the final desired property of the codebook. *This process can be viewed as transporting each quantile cell through a set of its uniformly spread samples.*

The trained flow serves as a proxy for the latent data distribution. Denoting N as a sampling ratio, we uniformly draw NK points $\{\mathbf{S}_0^i\}_{i=1}^{NK}$ from the prior and push them through the flow to the latent space as dense samples $\mathcal{S}_1$ similarly, simulating the incoming latent data for vector quantization as $\{z_e^{'}\}$.

$$\begin{aligned}&\mathbf{S}_0^i = \mathcal{F}^{-1}(\mathbf{v}_i), \quad \text{where} \quad \{\mathbf{v}_i\}_{i=1}^{NK} = \text{Sobol}(NK, d), \; \mathbf{v}_i \in [0,1]^d \\ &\mathbf{S}_1^i \leftarrow \psi_1^\gamma(\mathbf{S}_0^i), \quad i = 1, ..., NK\end{aligned} \tag{8}$$

We then cluster $\{\mathbf{S}_1^i\}_{i=1}^{NK}$ with respect to the current codebook $\mathcal{C}_1$ and update each $\mathbf{C}_1^i$ toward the corresponding cluster centroid, implementing a self-correction mechanism that improves codebook alignment with the latent manifold. This process is analogous to k-means clustering initialized with well-chosen centers.

$$\mathcal{C}_1 = \arg\min_{\mathcal{C}_1} \sum_{j=1}^{K} \sum_{\mathbf{S}_1^i \in Q_j} ||\mathbf{S}_1^i - \mathbf{C}_1^j||_2^2 \tag{9}$$

Where $\{Q_j\}_{j=1}^K$ denotes the K quantile clusters in the latent space, and $\mathbf{C}_1^j$ corresponds to the current codebook vectors of the quantiles. This process is performed iteratively a few times and deterministic with minimal overhead.

Optionally, we can further correct the codebook $\mathcal{C}_1$ online towards the cluster centroids of real latent data $\{z_e\}$ during co-training. This could reduce the remaining approximation error due to finite samples and model error. Specifically, this online correction reinstates the codebook loss of $\mathcal{L}_{\text{VQ-VAE}}$ in Eq (2) in Eq (6), which can pose a risk of re-introducing codebook collapse between two codebook updates. However, due to the quality of the flow partition and frequent codebook update, we did not observe significant collapse in our experiments and found it can further improve the model slightly over the purely self-corrected version.

4 Experiments

In this section, We evaluate FlowVQ on image reconstruction and generation in terms of generation quality with common metrics such as FID, IS, and codebook quality, measured by its entropy and utilization rate. We first evaluate it on CIFAR-10 [33] in Sec 4.1.1, followed by ImageNet-1K [34] at 256×256 resolution in Sec 4.1.2.

To further examine the expressivity induced by FlowVQ and its general applicability, we evaluate FlowVQ on two high-precision tasks: crystalline material tokenization in Sec 4.2.1, and action tokenization for robotics in Sec 4.2.2

4.1 Image Tokenization

4.1.1 CIFAR-10

As various design choices affect the final model performance, to understand the sole effect of VQ techniques, we first conduct a set of controlled experiments on CIFAR-10. All models share the same encoder and decoder architecture and training scheme (47M parameters), differing only in the quantizer, i.e., quantization and codebook update. The image is compressed to the latent space by a fixed factor $f = 4$, i.e. from $H \times W$ to an $H/f \times W/f$ patch token grid. We evaluate reconstruction under different codebook sizes and dimensions to examine the codebook scalability.

For baselines, we compare against two most common quantization methods: VQ-GAN [18, 13], and hypercubic LFQ method [12, 13]. We also compare with OptVQ [22]. For a fair comparison with LFQ method , which ties the codebook size $|\mathcal{C}|$ to the latent vector dimension $d : |\mathcal{C}| = 2^d$, we adjust d accordingly for $|\mathcal{C}| = 1024$ and $|\mathcal{C}| = 16384$. We also explored the effect of auxiliary entropy loss, a technique widely adopted to improve codebook utilization in existing works [11–13, 35], which optimizes the mutual information between the latent inputs and codebook, as illustrated in [36, 37]:

$$\mathcal{L}_{ent} = \mathbb{E}[H(q(z))] - H[\mathbb{E}(q(z))] \tag{10}$$

	d	$\lvert\mathcal{C}\rvert$	rFID↓	IS ↑	SSIM↑	PSNR↑	LPIPS↓	MAE↓	Entropy↑	Util % ↑
VQ-GAN+	10	1024	2.40	10.77	0.78	23.62	0.130	0.050	8.81	45.6
	10	10240	2.65	10.70	0.75	23.27	0.143	0.052	8.07	2.7
w/o $\mathcal{L}_{ent}$	14	16384	1.93	10.88	0.81	24.79	0.115	0.044	8.71	2.6
	14	16384	1.65	10.87	0.81	24.82	0.114	0.044	8.57	2.8
LFQ	10	1024	3.00	10.65	0.74	23.01	0.149	0.058	8.46	50.0
w/o $\mathcal{L}_{ent}$	14	16384	2.95	10.83	0.73	22.57	0.161	0.056	5.91	0.8
	14	16384	2.03	10.83	0.83	24.96	0.104	0.043	13.83	100.0
OptVQ	10	1024	3.42	10.79	0.76	23.50	0.138	0.050	9.99	100.0
	10	10240	1.95	10.88	0.85	25.31	0.091	0.041	13.27	100.0
	14	16384	1.72	**10.93**	0.85	25.78	0.087	0.039	13.94	100.0
FlowVQ	10	1024	2.11	10.71	0.80	24.28	0.118	0.046	9.93	100.0
	10	10240	1.87	10.68	0.83	25.39	0.100	0.040	13.28	100.0
	14	16384	1.40	10.72	0.86	26.50	0.086	0.036	**13.97**	100.0
w/ $\mathcal{L}_{ent}$	14	16384	**1.39**	10.89	**0.87**	**26.72**	**0.081**	**0.035**	**13.97**	100.0

Table 1: Evaluation of image quality and codebook utilization on CIFAR-10. By default, all baselines optimize the auxiliary entropy loss while FlowVQ does not if not specified. As the entropy maximization is tightly coupled in the OptVQ optimization process, we did not explore removing it.

For FlowVQ, the velocity field is modeled using a lightweight 4-layer MLP with hidden dimension $h = 256$ (0.2M extra parameters). We observe this setting to be effective in producing an aligned latent space and fix it across all tasks. We set the source distribution to a normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I}_d)$. The dense prior samples are generated with a ratio N = 50. We start the initial codebook construction at the 1000th step, then update it every 1000 steps using batch size 128.

Result Analysis The results are summarized in Table 1. We can spot the codebook collapse of the original VQ immediately when $|\mathcal{C}|$ increases from 1024 to 10240 with a fixed latent dimension. The

number of active codes decreases from $1024 \times 45.6\% = 467$ to $10240 \times 2.73\% = 280$, leading to degenerate overall performance. The entropy regularization drastically improves the utilization rate of LFQ, but does not guarantee full utilization when balancing multiple objectives of minimizing quantization error and maximally filling the hypercube simultaneously. The same conflict applies to OptVQ, which was observed to learn fast in early steps, but plateaus much earlier than methods that are solely driven by the quantization objective, such as VQGAN and FlowVQ here. Nevertheless, we note that VQ-GAN serves as a strong baseline due to its flexibility, which leads to high-quality alignment between the codebook and an evolving latent space.

We observe that FlowVQ achieves significant improvements in both image quality, with rFID = 1.40, reconstruction error MAE of 0.036, and codebook quality with full utilization across all settings. The performance consistently improves with increasing $|\mathcal{C}|$. Explicit optimizing the mutual information further improves the model slightly, though we note the performance is very close between the two. Both variants achieve the highest entropy (13.97, near the maximum $log_2 16384 = 14$), indicating the most balanced usage across codes, which validates the effectiveness of the proposed method.

Codebook Visualization We next provide visualizations of the latent space on the evaluation set and codebook alignment between FlowVQ and baselines without global information in Fig 2. We can see the codebook collapse pattern in VQ-GAN, where a large number of codes at the bottom left remain unused despite the active codes aligning well with the latent space, while LFQ imposes a hypercube geometry, so holes in latent space inevitably lead to unused codes. FlowVQ, on the contrary, models the geometry flexibly similar to VQ-GAN and imposes no additional regularization on the latent shape, resulting in high code utilization that allows it to scale with the codebook size.

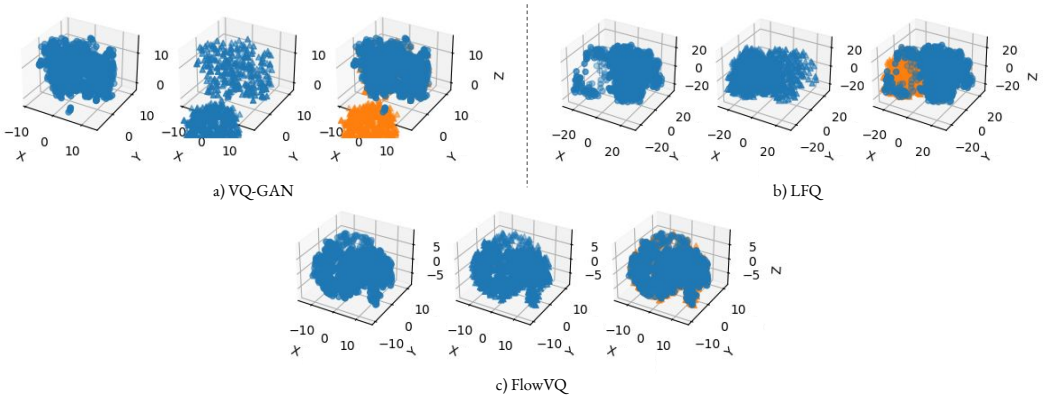

Figure 2: Visualization of ***(left)*** the latent vectors on the evaluation set ●; ***(middle)*** the codebook ▲; ***(right)*** the joint plot of latent vectors ● and codebook vectors ▲ in each plot. For each model, the latent vectors and code vectors are randomly down-sampled to the same number, then jointly reduced to 3D using t-SNE for visualization.

Time Analysis One potential concern for the proposed method is that the pushforward operation can be time-consuming. We measure the average training time per 1000 steps (one codebook update cycle) for all models in robotics VLA model Table 6 in Appendix B. FlowVQ takes on average 618s compared to 535s for VQ-GAN, increased by 15.6% with the difference mostly due to the codebook update, which takes 82.8s on average. This time is expected to further decrease with more GPU memory. Meanwhile, as the prior vectors are pre-defined in the fixed source space, the update is fully deterministic and easily supports distributed execution with minimal communication overhead. We implement the distributed update. It takes 3s per codebook update on 32 NVIDIA A100 GPUs for $|\mathcal{C}| = 16384$ for large scale training such as ImageNet, thus comparable speed to the original VQ-GAN.

4.1.2 ImageNet-1K

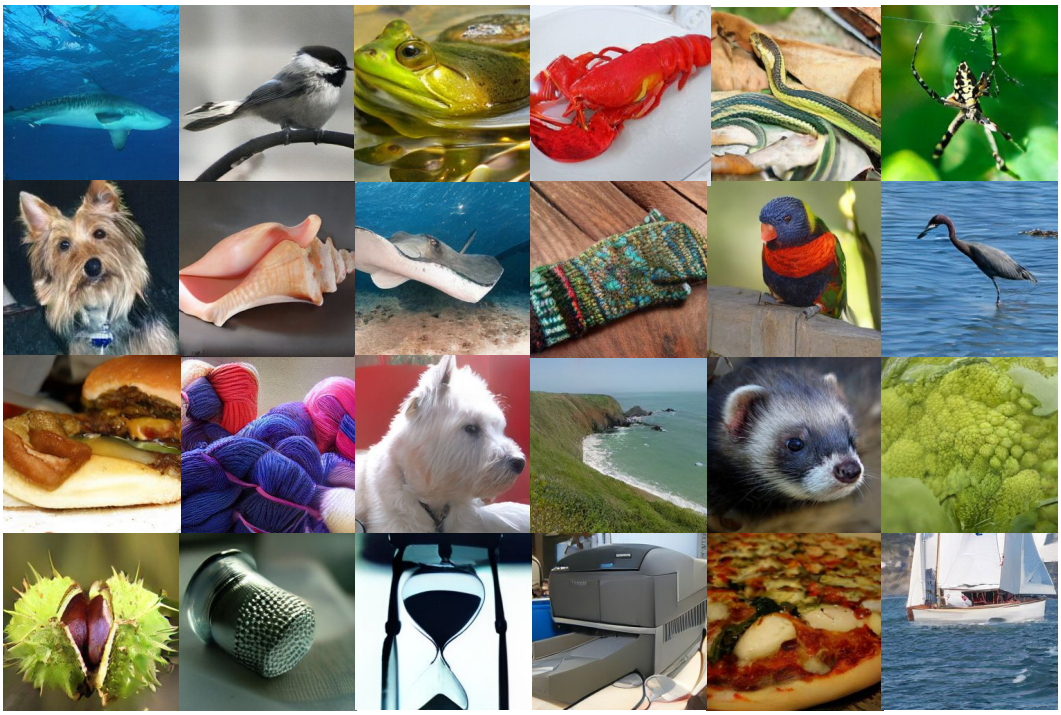

Figure 3: class-conditioned generation on ImageNet-1K 256×256 trained model. Each image corresponds to a different class. More samples can be found in Appendix C.3.

Model	Tokenizer	$\lvert\mathcal{C}\rvert$	#tokens	#params	rFID↓	gFID ↓	IS ↑
ImageNet-1K val. data [21]						1.78	236.9
LDM-4-G [23]	Continuous*	8192	64^2	400M	-	3.60	247.7
LDM-8-G [23]	Continuous*	16384	32^2	506M	-	7.76	209.5
MAR-L-407M [38]	Continuous	-	16^2	407M	1.22	1.97	291.2
FSQ [11]	LFQ	1000	16^2	155M	-	4.53	-
MAGVIT-v2 [12]	LFQ	262144	16^2	307M	-	**1.78**	319.4
Open-MAGVIT2-AR-B [39]	LFQ	262144	16^2	343M	1.67	3.08	258.3
BSQ [40]	LFQ	262144	16^2	174M	**1.14**	5.44	139.6
taming-VQGAN	VQ	1024	16^2	310M	7.94	5.88	304.8
taming-VQGAN	VQ	16384	16^2	310M	4.98	-	-
MaskGIT [35]	VQ	1024	16^2	227M	2.28	6.18	182.1
SQ-VAE-2 [7]	VQ	3072	16^2	-	4.51	-	-
LlamaGen [41]	VQ	16384	16^2	343M	2.19	4.21	200.1
VAR-d16 [21]	VQ	4096	16^2	310M	-	3.30	274.4
VAR-d20 [21]	VQ	4096	16^2	600M	-	2.57	302.6
FlowVQ	VQ	16384	16^2	307M	1.48	1.89	**351.4**

Table 2: Image reconstruction and generation on ImageNet-1K 256×256 with classifier free guidance. We compare FlowVQ with the existing works with similar model capacity and their guided or rejection sampling scores. * LDM reconstructs on continuous space, but generation discretizes input to the nearest token.

We next extend the evaluation on ImageNet-1K at a resolution of 256×256 for both reconstruction and generation. For stage one, the model shares a similar design to Sec 4.1.1 but is constructed deeper using channel multiplier as [1,1,2,2,4], leading to a compression ratio $f = 16$ and a 16×16 token

grid. We train the model for 1.35M steps using batch size 256, and a single-cycle cosine scheduler with a base learning rate 1e-4 using the AdamW optimizer [42].

For stage two, we train a Masked Language Model (MLM) to generate images similar to the existing line of works [35, 11, 43, 40, 39, 12]. We use a cosine mask schedule from MaskGIT [35] which demonstrated superior generative performance. The MLM contains 307M parameters, comparable to GPT2-medium capacity. It is trained with the same learning rate and steps as stage one. Following convention, we report FID [44] and IS [45] scores for guided generation. More hyperparameter details can be found in Appendix C.2.

From Table 2, under similar model capacity, FlowVQ $|\mathcal{C}| = 16384$ achieves a good balance between reconstruction and generation. It achieves the highest IS score and outperforms all models in terms of gFID with 1.89, except MAGVIT-v2 (1.78), where FlowVQ uses only 6.25% of its vocabulary size. The baselines include MAR-L-407M, which uses a slightly larger generator and autoregressively generates images in the original continuous latent space without tokenization, as well as models using advanced VQ techniques such as VAR and SQ-VAE-2, which use multi-scale and residual VQ, respectively. Like other VQ methods, FlowVQ is compatible with those techniques.

For qualitative measure, we plot the generated samples in Figure 3. We also validate generalization of the codebook by reconstructing unseen images from COCO-2017 validation set in Figure 4, where we found FlowVQ can produce more faithful and plausible reconstruction over details and particularly faces.



Figure 4: Reconstruction on unseen images from COCO-2017 validation data on ImageNet-1K 256×256 trained model.

4.2 High Precision Tokenization

To examine the potential of an adaptive and fully utilized codebook for FlowVQ as a general tokenizer, we would like to extend the evaluation beyond the image domain. One natural task to consider is high-precision tokenization, especially where structural representation is missing and generation can greatly benefit from the discretized representation learning. Data containing geometric, motion, and spatial state information usually resides in continuous space, and small numerical differences can lead to significantly different physical states or interaction outcomes. Tokenization needs to allocate sufficient resolution to ensure that distinct physical configurations do not collapse into indistinguishable tokens. The current common practice uses k-means initialized VQ-VAE [27–29]. We consider two representative tasks here: crystalline material tokenization in Sec 4.2.1 and robot action tokenization in Sec 4.2.2.

4.2.1 Crystalline Material Tokenization

We evaluate tokenization techniques for crystalline materials based on their ability to faithfully represent crystal structures using an atom-based encoder–decoder transformer. Crystals exhibit periodic atomic arrangements in 3D space and are commonly represented by a unit cell containing a set of atoms. Each atom is described by a tuple $(\mathbf{A}, \mathbf{X}, \mathcal{L})$, where $\mathbf{A}$ is its one-hot atom type vector, $\mathbf{X}$ is its 3D fractional coordinates, and $\mathcal{L}$ is the lattice information. These components are then concatenated into a single vector before being fed into the model.

We evaluate on MP-20 [46] with $|\mathcal{C}| = 8192$ and d = 8. As VQ-VAE and OptVQ benefit from an aligned initialization [27–29], and to isolate the effect of the quantizer, we initialize their codebooks with k-means centroids of the training-set latents from the same autoencoder at epoch W = 500, when FlowVQ quantization begins. All methods also start from the same encoder and decoder weights. For evaluation, we follow prior work [47] by comparing each reconstructed material with its ground-truth material using StructureMatcher from PyMatgen [48] and reporting the resulting match rate. We do not report codebook utilization rate for this task because MP-20 is a relatively small dataset, and the evaluation distribution does not fully cover the training distribution, which is reflected as only 7,408 of the 8,192 codes derived from the training data k-means clusters were actively used on the evaluation set. This number can further vary with the latent geometry evolving during training. We analyze the number of actively utilized codes instead.

	VQVAE	LFQ	OptVQ ($\epsilon = 10^*$)	KM-VQVAE	KM-OptVQ ($\epsilon = 10$)	KM-OptVQ ($\epsilon = 1$)	FlowVQ	FlowVQ (w/ oc)
Match Rate ↑	47.16 %	24.63%	10.81%	78.38%	17.85%	53.27%	**90.81** %	90.66 %

$^*\epsilon$ in Eq. 8 of the original paper, where $\epsilon = 10$ is the open-source configuration across all tasks and used for table 1 experiments.

Table 3: Match rate between the reconstructed and ground-truth material on MP-20.

FlowVQ significantly outperforms all baseline methods, including k-means initialized techniques. It achieves the highest match rate (90.81%) with 7178 actively utilized codes on average, compared to the second best method KM-VQVAE as 78.38% with the average actively used code number as 3192. LFQ and OptVQ generally perform poorly on this task. For k-means initialized OptVQ, reducing ϵ from 10 to 1, which strengthens the entropy regularization as its weight scales with $\frac{1}{\epsilon}$, improves the match rate from 17.85% to 53.27% but significantly reduces the number of active codes from 7110 to 1311, consistent with our analysis in Sec 2. More details on the experiment setting can be found in Appendix D.

4.2.2 Robotics Action Tokenization

Robot actions in foundation models [49–52] have recently been represented as action chunks [53, 54], as fixed-length sequences of actions. Each action is represented as a vector defining the next delta movement. We evaluate on the DROID dataset [55] and compare with common action tokenizers: 1) FAST+ [56], which applies the Discrete Cosine Transform (DCT) to each action dimension of a chunk, quantizes the coefficients by scaling and rounding, flattens them from low to high frequency, and compresses the resulting sequence with a pretrained Byte-Pair Encoding (BPE) [57, 58] vocabulary, 2) K-means initialized VQ-VAE. Similar to Sec 4.2.1, we use an encoder-decoder transformer with quantization applied in the latent space. $|\mathcal{C}|$ is set to 2048 for fair comparison with existing tokenizers, latent dimension d = 8, and more details can be found in Appendix E.

	FAST+	KM-VQVAE	FlowVQ
MSE(e-4) ↓	2.40	1.88	**1.71**
Utilization ↑	97.02%	10.89%	**100%**

Table 4: Trajectory reconstruction error of 50-step action chunks on DROID dataset.

From Table 4, we observe FlowVQ achieves full codebook utilization on validation trajectories throughout the training, and reaches the lowest reconstruction error of $1.71e^{-4}$. KM-VQVAE starts with full utilization, but gradually drops to 10.89% during training. Both FlowVQ and KM-VQVAE yield a more accurate reconstruction of action compared to FAST+.

4.3 Ablation Study

To better understand the contribution of the learned transport and of each codebook correction mechanism, we ablate FlowVQ on CIFAR-10 and MP-20 and report the best model performance in Table 5. On both tasks, the prior codebook transport can deliver more accurate tokenization compared to baseline methods reflected as lower FID score and higher material match rate (FID 1.49 vs 1.65 in Table 1; 88.15% vs 78.38% in Table 3), and adding self-correction stabilizes the training with less fluctuation observed. Adding online correction on top further improves image tokenization slightly, while performing on par on material tokenization.

(a) Image Tokenization: CIFAR-10

	FID ↓	**IS** ↑	**SSIM** ↑	**PSNR** ↑	**LPIPS** ↓	**MAE** ↓	**Entropy** ↑	**Util** ↑
$\mathcal{C}_0 \to \mathcal{C}_1$	1.49	**10.79**	0.856	26.11	0.086	0.038	13.92	100%
$\mathcal{C}_0 \to \mathcal{C}_1$ + sc	1.52	10.70	0.841	25.73	0.094	0.039	13.97	100%
$\mathcal{C}_0 \to \mathcal{C}_1$ + sc + oc	**1.40**	10.72	**0.861**	**26.50**	**0.086**	**0.036**	**13.97**	100%

(b) Crystalline Material Tokenization: MP-20

MP-20 Match Rate	$\mathcal{C}_0 \to \mathcal{C}_1$	$\mathcal{C}_0 \to \mathcal{C}_1$ + sc	$\mathcal{C}_0 \to \mathcal{C}_1$ + sc + oc
Match Rate ↑	88.15%	**90.81%**	90.66%

Table 5: Ablation study on FlowVQ components on CIFAR-10 image tokenization and MP-20 material tokenization tasks. With details in Sec 3.3, *sc* refers to self-correction mechanism, and *oc* refers to the optional online-correction mechanism.

5 Conclusion

To summarize, we introduced FlowVQ, a vector quantization technique that reformulates codebook learning in classic VQ-VAE as a quantile-based transport process, where balanced code utilization is governed by the geometry of the latent space rather than a byproduct of auxiliary optimization objectives. By constructing the codebook via deterministic pushforwards from a uniform prior and refining them through self-correction, FlowVQ achieves stable and balanced code usage while avoiding codebook collapse. We conducted experiments across a range of tokenization and generation tasks, demonstrating consistently improved reconstruction and generation quality. We hope the proposed method will serve as a useful tool, especially in fields where features are naturally continuous or hybrid, and benefit from learned representations.

Limitations and Future Work In addition to the slight increase in training time, FlowVQ primarily focuses on addressing codebook collapse, while another important limitation of VQ-VAE lies in the reliance on the biased STE for backpropagation through discrete quantization. Recent works have shown that this gradient mismatch can be alleviated through simple rotation tricks [59], but more effective alternatives remain largely unexplored. Another direction worth exploring is to investigate FlowVQ together with recent architectures on flexible-length [60, 61] or flexible-resolution [62] embedding learning, which may enable more adaptive token allocation to regions with higher semantic or structural complexity, leading to further improvements.

Acknowledgments We thank the anonymous reviewers for their constructive feedback and suggestions, which improved the paper. We thank Liu Nian for sharing his code for k-means initialized VQ-VAE for crystalline material tokenization, which enabled an insightful comparison during the development. Zekun Shi is partially supported by the Singapore Economic Development Board through the Industrial Postgraduate Programme.

References

[1] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[2] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019.

[3] 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.

[4] 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. *Advances in neural information processing systems*, 30, 2017.

[5] Chuanxia Zheng and Andrea Vedaldi. Online clustered codebook. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22798–22807, 2023.

[6] 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.

[7] 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*, 2022.

[8] Yuhta Takida, Yukara Ikemiya, Takashi Shibuya, Kazuki Shimada, Woosung Choi, Chieh-Hsin Lai, Naoki Murata, Toshimitsu Uesaka, Kengo Uchida, Wei-Hsiang Liao, et al. Hq-vae: Hierarchical discrete representation learning with variational bayes. *Trans. Mach. Learn. Res.*, 2024.

[9] 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%. *Advances in Neural Information Processing Systems*, 37: 12612–12635, 2024.

[10] Yifan Chang, Jie Qin, Limeng Qiao, Xiaofeng Wang, Zheng Zhu, Lin Ma, and Xingang Wang. Scalable training for vector-quantized networks with 100% codebook utilization. *International Conference on Learning Representations*, 2026.

[11] Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: Vq-vae made simple. In *The Twelfth International Conference on Learning Representations*, 2024.

[12] Lijun Yu, José Lezama, Nitesh Bharadwaj Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Agrim Gupta, Xiuye Gu, Alexander G Hauptmann, et al. Language model beats diffusion-tokenizer is key to visual generation. In *ICLR*, 2024.

[13] Mark Weber, Lijun Yu, Qihang Yu, Xueqing Deng, Xiaohui Shen, Daniel Cremers, and Liang-Chieh Chen. Maskbit: Embedding-free image generation via bit tokens. *Transactions on Machine Learning Research*, 2024.

[14] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[15] Dimitra Maoutsa, Sebastian Reich, and Manfred Opper. Interacting particle solutions of fokker–planck equations through gradient–log–density estimation. *Entropy*, 22(8):802, 2020.

[16] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[17] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. *Advances in neural information processing systems*, 29, 2016.

[18] 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.

[19] 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.

[20] 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.

[21] Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *Advances in neural information processing systems*, 37:84839–84865, 2024.

[22] Borui Zhang, Wenzhao Zheng, Jie Zhou, and Jiwen Lu. Preventing local pitfalls in vector quantization via optimal transport. *arXiv preprint arXiv:2412.15195*, 2024.

[23] 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.

[24] Sander Dieleman, Aaron Van Den Oord, and Karen Simonyan. The challenge of realistic music generation: modelling raw audio at scale. *Advances in neural information processing systems*, 31, 2018.

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

[26] Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *Forty-first International Conference on Machine Learning*, 2024.

[27] Mahdi Pourmirzaei, Alex Morehead, Farzaneh Esmaili, Jarett Ren, Mohammadreza Pourmirzaei, and Dong Xu. Gcp-vqvae: A geometry-complete language for protein 3d structure. *bioRxiv*, pages 2025–10, 2025.

[28] Shuqi Lu, Haowei Lin, Lin Yao, Zhifeng Gao, Xiaohong Ji, Linfeng Zhang, Guolin Ke, et al. Uni-3dar: Unified 3d generation and understanding via autoregression on compressed spatial tokens. *arXiv preprint arXiv:2503.16278*, 2025.

[29] ZiJie Qiu, Luozhijie Jin, Zijian Du, Hongyu Chen, Yan Cen, Siqi Sun, Yongfeng Mei, and Hao Zhang. Vqcrystal: Leveraging vector quantization for discovery of stable crystal structures. *arXiv preprint arXiv:2409.06191*, 2024.

[30] Alexei Baevski, Steffen Schneider, and Michael Auli. vq-wav2vec: Self-supervised learning of discrete speech representations. In *International Conference on Learning Representations*, 2020.

[31] Shiyin Jiang, Wei Long, Minghao Han, Zhenghao Chen, Ce Zhu, and Shuhang Gu. Differentiable vector quantization for rate-distortion optimization of generative image compression. *arXiv preprint arXiv:2604.10546*, 2026.

[32] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017.

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

[34] 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.

[35] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11315–11325, 2022.

[36] Andreas Krause, Pietro Perona, and Ryan Gomes. Discriminative clustering by regularized information maximization. *Advances in neural information processing systems*, 23, 2010.

[37] Aren Jansen, Daniel PW Ellis, Shawn Hershey, R Channing Moore, Manoj Plakal, Ashok C Popat, and Rif A Saurous. Coincidence, categorization, and consolidation: Learning to recognize sounds with minimal supervision. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 121–125. IEEE, 2020.

[38] Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. *Advances in Neural Information Processing Systems*, 37:56424–56445, 2024.

[39] Zhuoyan Luo, Fengyuan Shi, Yixiao Ge, Yujiu Yang, Limin Wang, and Ying Shan. Open-magvit2: An open-source project toward democratizing auto-regressive visual generation. *arXiv preprint arXiv:2409.04410*, 2024.

[40] Yue Zhao, Yuanjun Xiong, and Philipp Kraehenbuehl. Image and video tokenization with binary spherical quantization. In *The Thirteenth International Conference on Learning Representations*, 2025.

[41] Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. Autoregressive model beats diffusion: Llama for scalable image generation. *arXiv preprint arXiv:2406.06525*, 2024.

[42] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[43] Lijun Yu, Yong Cheng, Kihyuk Sohn, José Lezama, Han Zhang, Huiwen Chang, Alexander G Hauptmann, Ming-Hsuan Yang, Yuan Hao, Irfan Essa, et al. Magvit: Masked generative video transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10459–10469, 2023.

[44] 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.

[45] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

[46] Anubhav Jain, Shyue Ping Ong, Geoffroy Hautier, Wei Chen, William Davidson Richards, Stephen Dacek, Shreyas Cholia, Dan Gunter, David Skinner, Gerbrand Ceder, et al. Commentary: The materials project: A materials genome approach to accelerating materials innovation. *APL materials*, 1(1), 2013.

[47] Chaitanya K Joshi, Xiang Fu, Yi-Lun Liao, Vahe Gharakhanyan, Benjamin Kurt Miller, Anuroop Sriram, and Zachary Ward Ulissi. All-atom diffusion transformers: Unified generative modelling of molecules and materials. In *Forty-second International Conference on Machine Learning*, 2025.

[48] Shyue Ping Ong, William Davidson Richards, Anubhav Jain, Geoffroy Hautier, Michael Kocher, Shreyas Cholia, Dan Gunter, Vincent L Chevrier, Kristin A Persson, and Gerbrand Ceder. Python materials genomics (pymatgen): A robust, open-source python library for materials analysis. *Computational Materials Science*, 68:314–319, 2013.

[49] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, et al. Openvla: An open-source vision-language-action model. In *8th Annual Conference on Robot Learning*, 2024.

[50] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[51] Jose Barreiros, Andrew Beaulieu, Aditya Bhat, Rick Cory, Eric Cousineau, Hongkai Dai, Ching-Hsin Fang, Kunimatsu Hashimoto, Muhammad Zubair Irshad, Masha Itkina, et al. A careful examination of large behavior models for multitask dexterous manipulation. *Science Robotics*, 11(113):eaea6201, 2026.

[52] Han Xue, Jieji Ren, Wendi Chen, Gu Zhang, Yuan Fang, Guoying Gu, Huazhe Xu, and Cewu Lu. Reactive diffusion policy: Slow-fast visual-tactile policy learning for contact-rich manipulation. In *Proceedings of Robotics: Science and Systems (RSS)*, 2025.

[53] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[54] Lucy Lai, Ann ZX Huang, and Samuel J Gershman. Action chunking as conditional policy compression. *Cognition*, 264:106201, 2025.

[55] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[56] Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

[57] Philip Gage. A new algorithm for data compression. *C Users Journal*, 12(2):23–38, 1994.

[58] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 1715–1725, 2016.

[59] Christopher Fifty, Ronald Guenther Junkins, Dennis Duan, Aniketh Iyengar, Jerry Weihong Liu, Ehsan Amid, Sebastian Thrun, and Christopher Re. Restructuring vector quantization with the rotation trick. In *The Thirteenth International Conference on Learning Representations*, 2025.

[60] Keita Miwa, Kento Sasaki, Hidehisa Arai, Tsubasa Takahashi, and Yu Yamaguchi. One-d-piece: Image tokenizer meets quality-controllable compression. *arXiv preprint arXiv:2501.10064*, 2025.

[61] Roman Bachmann, Jesse Allardice, David Mizrahi, Enrico Fini, Oğuzhan Fatih Kar, Elmira Amirloo, Alaaeldin El-Nouby, Amir Zamir, and Afshin Dehghan. Flextok: Resampling images into 1d token sequences of flexible length. In *Forty-second International Conference on Machine Learning*, 2025.

[62] Yucheng Mao, Zeyuan Chen, Xiaojun Shan, Xiang Zhang, Divyansh Srivastava, Bingnan Li, and Zhuowen Tu. Quadtok: Quadtree visual tokenizer for autoregressive image generation. *arXiv preprint arXiv:2610.10497*, 2026.

[63] Nian Liu, Yuwei Zeng, Ryoji Kubo, Nikita Kazeev, Stephen Gregory Dale, Artem Maevskiy, Pengru Huang, Thomas Laurent, Kostya S Novoselov, and Xavier Bresson. Composable crystals: Controllable materials discovery via concept learning. *arXiv preprint arXiv:2605.14769*, 2026.

[64] Claudio Zeni, Robert Pinsler, Daniel Zügner, Andrew Fowler, Matthew Horton, Xiang Fu, Zilong Wang, Aliaksandra Shysheya, Jonathan Crabbé, Shoko Ueda, et al. A generative model for inorganic materials design. *Nature*, 639(8055):624–632, 2025.

[65] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. *IEEE Transactions on Big Data*, 2025.

Appendix

A Algorithm

Here we now present the pseudocode of the complete training scheme for FlowVQ in Algorithm 1.

Algorithm 1 Flow-based Vector Quantization (FlowVQ)

Require: Dataset $\mathcal{D}$, Encoder E_θ, Decoder D_θ, Quantizer ψ_γ
Require: T - total steps, K - codebook size, d - codebook dimension, N - dense sample ratio
W - starting steps, M - codebook update frequency , β - $\mathcal{L}_{\text{commitment}}$ coefficient

Initialize prior codebook and dense samples:

$$\{\mathbf{C}_0^i\}_{i=1}^K = \mathcal{F}^{-1}(\textit{Sobol}(K,d)), \quad \{\mathbf{S}_0^i\}_{i=1}^{NK} = \mathcal{F}^{-1}(\textit{Sobol}(NK,d))$$

for $i = 1$ **to** T **do**
  if $i < W$ **then**
    $\mathcal{L}_{\text{recon}}(\theta) = \|D_\theta(E_\theta(x)) - x\|_2^2$
    $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_{\text{recon}}$
  else *# Train tokenization with FlowVQ*
    if $i = W$ **then** *# Initialize flow ψ_γ at step W*
      while not converged **do**
        $\mathcal{L}_{\text{FM}}(\gamma) = \mathbb{E}\big[\|u_t^\gamma(z) - u_t(z|E_\theta(x), z_0)\|_2^2\big]$ for $x \in \mathcal{D}$
        $\gamma \leftarrow \gamma - \eta \nabla_\gamma \mathcal{L}_{\text{FM}}$
      end while
    end if
    if $(i - W) \bmod M = 0$ **then** *# Update $\mathcal{C}_1$ periodically afterwards*
      $\mathcal{C}_1 = \psi_1^\gamma(\mathcal{C}_0), \mathcal{S}_1 = \psi_1^\gamma(\mathcal{S}_0)$
      $\mathcal{C}_1 \leftarrow \arg\min_{\mathcal{C}_1} \sum_{j=1}^K \sum_{s_1^i \in Q_j} \|s_1^i - \mathbf{C}_1^j\|_2^2$
    end if
    $z_e = E_\theta(x)$
    $z_q = \text{VQ}(z_e, \mathcal{C}_1)$
    $\mathcal{L}_{\text{FlowVQ}}(\theta, \gamma) = \|D_\theta(\text{STE}(z_q)) - x\|_2^2 + \beta\|z_e - \text{sg}(z_q)\|_2^2 + \|u_t^\gamma(z) - u_t(z \mid z_e, z_0)\|_2^2$
    $(\theta, \gamma) \leftarrow (\theta, \gamma) - \eta \nabla_{(\theta,\gamma)} \mathcal{L}_{\text{FlowVQ}}$
  end if
end for

B Training Time Comparison

Here we report the mean training time per 1000 steps as one codebook update cycle using $|\mathcal{C}| = 16384$.

For training time, the relatively higher variance of FlowVQ is due to the usage of adaptive ODE solver Dopri45, where we did not investigate alternative solvers in this work. In particular, fixed-step Euler integration may offer further speedups without significantly compromising performance, a possibility that remains to be explored.

	VQ-GAN	LFQ	OptVQ	FlowVQ w/ sc
t (s)↓	535.2 ± 0.4	578.2 ± 0.4	596.4 ± 0.5	618.0 ± 13.5

Table 6: Runtime per 1,000 steps (one codebook update cycle) on a single NVIDIA A5000 GPU, with no other processes running and $|\mathcal{C}| = 16384$.

C Image Tokenization

C.1 Image Reconstruction

The hyper-parameters for image tokenization tasks for both CIFAR-10 and ImageNet-1K 256×256 are listed in Table 7:

Config	CIFAR-10 Reconstruction	ImageNet-1k Reconstruction
input size	$32 \times 32 \times 3$	$256 \times 256 \times 3$
latent size	$8 \times 8 \times d$	$16 \times 16 \times d$
batch size	128	256
Data Augmentation	False	True
random cropping		[0.8, 1.0]
random aspect ratio		$[\frac{3}{4}, \frac{4}{3}]$
resize to input size		True
random horizontal flip		True
Optimizer	AdamW	
base lr	$1e^{-4}$	
beta	[0.9, 0.999]	
weight decay	$1e^{-4}$	
warmup step	0	5000
scheduler	Cosine	
gradient clipping norm	1.0	
total steps	$\|C\|$ = 1024, 10240, 16384 - 165K, 165K, 325K	1.35M
Model		
hidden dimension	128	
channel multiplier	[1, 2, 4]	[1, 1, 2, 2, 4]
residual blocks	2	2
Loss Coefficients		
discriminator start step	20k	
β	0.25	
$\mathcal{L}_{reconstruction}$ weight	4.0	
$\mathcal{L}_{discriminator}$	0.02	
$\mathcal{L}_{perceptual}$	0.1	
$\mathcal{L}_{LeCam}$	0.001	
$\mathcal{L}_{Entropy}$	0.02	
$\mathcal{L}_{fm}$	1	
GPU	A5000	A100

Table 7: Hyperparameters for image reconstruction.

C.2 Masked Language Model

The hyper-parameters for image generation using masked language model is listed here for ImageNet-1K 256×256 are listed in Table 8 below:

Config	
#Tokens	16×16
batch size	1024
Data Augmentation	
random cropping	[0.8, 1.0]
resize to input size	True
random horizontal flip	True
Optimizer	AdamW
base lr	$1e^{-4}$
beta	[0.9, 0.96]
weight decay	0.045
warmup step	5000
scheduler	Cosine
gradient clipping norm	1.0
total steps	1.35M
Model	
hidden dimension	4096
attention heads	16
depth	24
dropout	0.1
label dropout	0.1
class label smoothing	0.1
mask schedule	arccos
classifier free sampling	
temperature	9.0
guidance	7.6
scale power	2.5
GPU	A100

Table 8: Hyperparameters for masked language model on image generation.

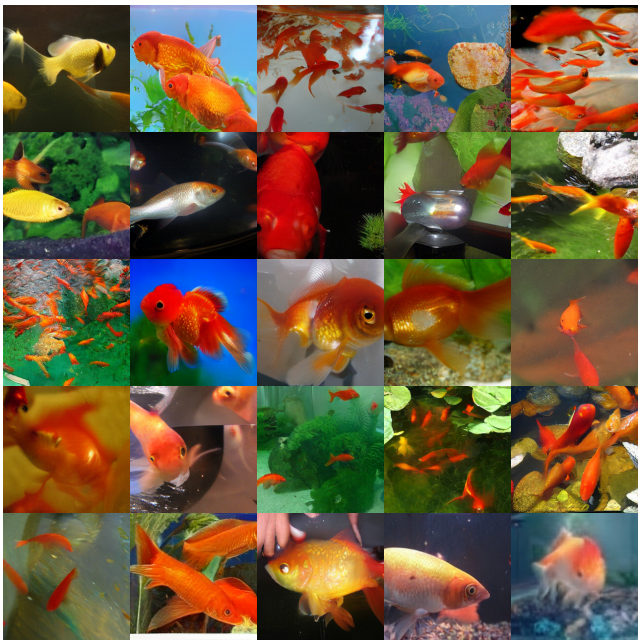

Figure 5: goldfish, Carassius auratus

C.3 More Generated Samples

In this section, we present more samples using classifier free guidance. The model is given the same label in the batch size 25. We reshape the output to $[5 \times 256, 5 \times 256]$ and plot here over different classes:



Figure 6: ostrich, Struthio camelus



Figure 7: tree frog, tree-frog



Figure 8: tusker



Figure 9: king penguin, Aptenodytes patagonica



Figure 10: lion, king of beasts, Panthera leo



Figure 11: mongoose



Figure 12: cab, hack, taxi, taxicab



Figure 13: carpenter's kit, tool kit

D Material Tokenization

We follow [63] for the data processing, model design and training for crystalline material tokenization using atom only features without using atom neighbor information. Each material is represented by its minimum repeating unit, containing the lattice information as 3 lattice base vectors $L = [\boldsymbol{l_1}, \boldsymbol{l_2}, \boldsymbol{l_3}]$, and a set of atoms within, represented using a one-hot atom type A and continuous 3D coordinate $X = [x, y, z]$ in the fractional coordinate. Following Mattergen [64], a normalized rotational invariant representation $\tilde{\boldsymbol{L}}$ is used, where $\boldsymbol{W}$ and $\boldsymbol{V}$ are the left and right singular vectors of $\boldsymbol{L}$, respectively, and $\boldsymbol{\Sigma}$ is the diagonal matrix of singular values.

$$\boldsymbol{L} = \boldsymbol{U}\tilde{\boldsymbol{L}}, \quad \boldsymbol{U} = \boldsymbol{W}\boldsymbol{V}^\top, \quad \tilde{\boldsymbol{L}} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{V}^\top,$$
$$\tilde{\boldsymbol{L}} = \frac{\tilde{\boldsymbol{L}}}{\sqrt[3]{N_{atom}}}.$$

We used a transformer-based encoder-decoder architecture, where the encoder contains 6 and decoder contains 2 transformer blocks. The hidden dimension is 512 and each transformer block uses 8 attention heads. We set the starting epoch W to 500, and update the codebook at every M = 5 epochs. The model is trained using Adam optimizer, with constant learning rate $1e-4$ and batch size 128. The codebook size K = 8192 and vector size d = 8. The loss function is defined as follows where $\mathcal{L}_{\text{codebook}}$ is enabled only when online correction is used, and $\alpha = 0.01$ is chosen for both VQ-VAE and FlowVQ to keep all loss terms on comparable scales:

$$\mathcal{L}(\theta, \gamma) = \mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{FM}} + \alpha(\mathcal{L}_{\text{codebook}} + \beta\mathcal{L}_{\text{commitment}})$$

For baseline models, we performed k-means clustering on the training data latent vectors of the autoencoder at epoch W using `FAISS` [65][1], and used the resulting cluster centroids to initialize VQ-VAE and Opt-VQ codebook vectors. To ensure consistency with the initial latent space, and fair comparison from the same starting point, we also initialized the encoder and decoder weights using those of the autoencoder. For LFQ, we used the implementation from `vector-quantize-pytorch` library[2].

E Action Tokenization

The action tokenization is evaluated on the DROID dataset [55], which contains 76k successful demonstration trajectories. We split the original trajectories into 80% train, 10% validation and 10% test set, then further processed them into non-overlapping chunks of 50 steps. The codebook size K is set to 2048 for fair comparison with existing tokenizers, and vector size d = 8. The codebook construction starts at step $W = 1000^{th}$, and is update at every 250th steps using batch size 512. The model contains 8 transformer blocks for both encoder and decoder, using 8 attention heads and hidden dimension as 512. Similarly, we train the model using a constant learning rate $1e^{-4}$ and Adam optimizer.

For the baseline FAST+ tokenizer, we use the weight downloaded from its original HuggingFace repository[3].

[1] `https://github.com/facebookresearch/faiss`

[2] `https://github.com/lucidrains/vector-quantize-pytorch`

[3] `https://huggingface.co/physical-intelligence/fast`

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our results are backed by experimental evidence provided in section 4 with 4 tasks across 3 domains.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations are discussed in section 5.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: This paper does not contain theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The experiment details are documented in section 4 and additional hyperparameter setting in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Code is available at: https://github.com/friolero/FlowVQ.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provided the details in section 4 under each experiment section.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We use 50000 images for FID and IS score measure, and report averaged time taken from 5 measures.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: we included the hardware specification in the experiment section.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have read the NeurIPS Code of Ethics and made sure that the paper conforms to it.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This work is not tied to particular applications but addresses an issue with classic neural tokenization method. The potential societal impacts of certain applications are not relevant to this work.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We did not use pretrained models or scraped dataset but only common benchmark datasets.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Citations to relevant models and works are given.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: This paper does not introduce new datasets. The details of the models are provided in section 4 and appendix.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This work does not involve crowdsourcing experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This work does not involve human participants.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core methodology of this work does not involve LLMs as an important, original, or non-standard component.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.