Image Understanding Makes for A Good Tokenizer for Image Generation

Luting Wang[*†]   **Yang Zhao**[1†]   **Zijian Zhang**[1]   **Jiashi Feng**[1]   **Si Liu**[‡]   **Bingyi Kang**[1‡§]

[1]ByteDance

Abstract

Modern image generation (IG) models have been shown to capture rich semantics valuable for image understanding (IU) tasks. However, the potential of IU models to improve IG performance remains uncharted. We address this issue using a token-based IG framework, which relies on effective tokenizers to map images into token sequences. Currently, *pixel reconstruction* (*e.g.*, VQGAN) dominates the training objective for tokenizers. In contrast, our approach adopts the *feature reconstruction* objective, where tokenizers are trained by distilling knowledge from pretrained IU encoders. Comprehensive comparisons indicate that tokenizers with strong IU capabilities achieve superior IG performance across a variety of metrics, datasets, tasks, and proposal networks. Notably, VQ-KD$_{\text{CLIP}}$ achieves 4.10 FID on ImageNet-1k. Visualization suggests that the superiority of VQ-KD can be partly attributed to the rich semantics within the VQ-KD codebook. We further introduce a straightforward pipeline to directly transform IU encoders into tokenizers, demonstrating exceptional effectiveness for IG tasks. These discoveries may energize further exploration into image tokenizer research and inspire the community to reassess the relationship between IU and IG. The code is released at `https://github.com/magic-research/vector_quantization`.

1 Introduction

Image understanding (IU) and image generation (IG) have been the core pursuits of computer vision research for a long time. Thanks to the progress in generative models [15,35–37,48] and network architectures [9,43], IG has witnessed remarkable advancements in recent years. These advancements spurred extensive research on leveraging powerful IG models for IU tasks (Fig. 1). Studies have shown that IG models can benefit IU tasks in various ways, including data augmentation through synthetic data generation [45, 46, 51], improved representation learning [16, 28, 52], and utilizing intermediate features from IG models for solving perception tasks [26, 53]. However, the reciprocal question remains largely uncharted: *how might IU models aid IG tasks?*

The primary focus of this paper lies in the AutoRegressive (AR) IG framework, which is gaining considerable attention for its excellence in generating high-quality images and videos [20, 48, 50]. This framework operates in a two-stage process. The first stage learns a tokenizer to map images into sequences of discrete tokens. Subsequently, the second stage trains a proposal network to model the token sequences. As underlined by prior research [4, 49], the quality of the tokenizers significantly influences overall IG performance. Meanwhile, tokenizers and IU encoders adhere to a similar structure as they both aim to map images into latent representations, either discrete or continuous. As a result, the token-based IG framework provides an optimal environment for investigating the relationship between IU and IG. Through comprehensive studies, we demonstrate
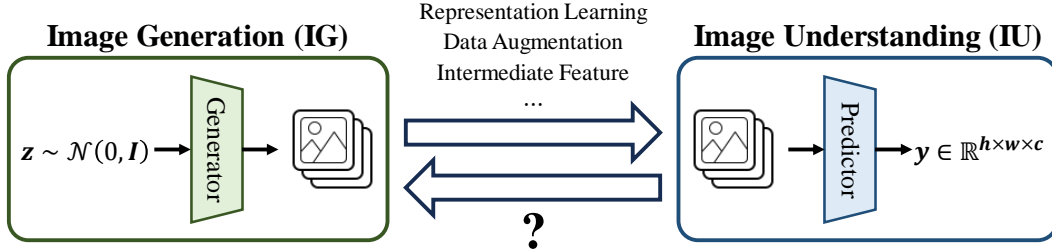

Figure 1: Extensive studies have tried to adopt IG models for IU. However, few attempts have been made to use IU models in IG.

that existing IU models from representation learning can be useful in generative models, even if they are not specifically designed for the IG task.

Our study involves training three components within the AR framework: tokenizer, decoder, and proposal network. Traditionally, *pixel reconstruction* has been the dominant objective for training tokenizers, such as VQGAN [10] and FSQ [25]. To the best of our knowledge, we are the first to systematically demonstrate that *feature reconstruction* (VQ-KD [27]) achieves better IG performance. [1] This approach distills knowledge from pretrained IU encoders to tokenizers. Therefore, the training strategy of the IU encoder is crucial for the performance of the tokenizer. In this regard, we investigate four representative IU encoders: ViT [9], CLIP [29], DINO [3], and MAE [13]. Following VQGAN [10], we train decoders to restore pixel values from discrete tokens, and proposal networks (AR or NAR) that can model the distribution of image tokens. The models are then evaluated using various metrics, including codebook usage, Fréchet Inception Distance (FID) [14], Inception Score (IS) [34], perplexity (PPL), *etc*.

Initially, we compare the above tokenizers on IN-1k for class-conditional IG. VQ-KD achieves 4.10 FID_{AR}, outperforming VQGAN (15.78 FID_{AR}) by a large margin. FSQ experiments confirm that the superiority of VQ-KD is not solely attributable to the specific quantization operation or high codebook usage. More generally, VQ-KD consistently outperforms across different proposal networks, datasets, and tasks.

We analyze VQ-KD from multiple perspectives. By visualizing the codebook, we discover that codes from VQ-KD carry more semantics than VQGAN, which makes them easier to model and subsequently improve the IG quality. Building upon this insight, we propose a straightforward pipeline to efficiently transform IU encoders into tokenizers, outperforming VQ-KD on the MS-COCO dataset. We also find that tokenizers with weaker IU capabilities require larger proposal networks for effective AR modeling and show less robustness to variations in the training images. Finally, we conduct qualitative analysis to present the visual results.

In sum, the key insights from our study include the following: 1) This research is the first to demonstrate that IU models can substantially enhance IG through VQ-KD; 2) Tokenizers with strong IU capabilities consistently outperform conventional VQGAN-based methods across various metrics, datasets, tasks, and network architectures; 3) The VQ-KD codebook encapsulates more semantics than VQGAN, contributing to the superiority of VQ-KD in IG.

We believe these findings can benefit future research on image tokenizers and provoke further discussion on the relationship between IU and IG.

2   Related Work

Image Tokenization.   Vector Quantization (VQ) [11] is originally developed for data compression. To circumvent posterior collapse in the VAE [19] framework, VQ-VAE [41] adopts VQ for image tokenization. Subsequently, VQGAN [10] introduces adversarial and perceptual losses to enhance the quality of the generated images.

Vanilla VQGAN suffers from limitations like low codebook usage, limited semantic representation ability, and the trade-off between modeling efficiency and image quality. To address these

---

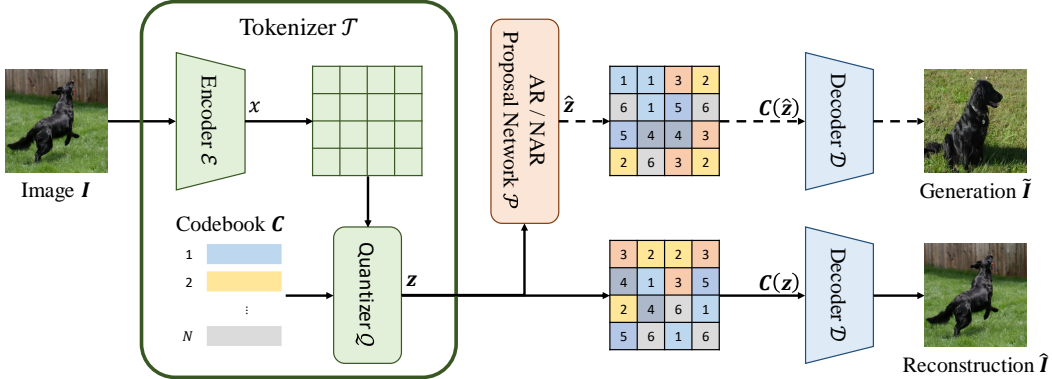[1]VQ-KD was initially proposed for image pretraining.

Figure 2: The token-based IG framework. Solid and dashed lines represent training and inference pipelines, respectively. During training, the tokenizer $\mathcal{T}$ tokenizes an image $\mathbf{I}$ into discrete codes $\mathbf{z}$. A proposal network $\mathcal{P}$ is trained to model the distribution $p(\mathbf{z})$, while a decoder $\mathcal{D}$ learns to reconstruct $\mathbf{I}$. During inference, we sample codes $\hat{\mathbf{z}}$ from $\mathcal{P}$, which guides $\mathcal{D}$ to perform generation.

challenges, researchers have focused on improving the codebook. ViT-VQGAN [47] adopts a ViT-based [9] autoencoder to create more expressive code vectors. SeQ-GAN [12] improves the perceptual loss and decoder to balance between semantic compression and detail preservation. SQ-VAE [40] improves VQ-VAE with stochastic quantization and a trainable posterior categorical distribution. VQ-WAE [44] builds upon SQ-VAE by encouraging the discrete representation to be a uniform distribution via a Wasserstein distance. HQ-VAE [39] employs random re-initialization of inactive code vectors. CVQ-VAE [55] selects encoded features as anchors to update dead codes. VQ-KD [27] adopts knowledge distillation instead of image reconstruction as the objective to train VQ-VAE. LFQ [49] and FSQ [25] adopt bounded scalar quantization techniques from neural compression to harness the potential of extra-large codebooks.

Furthermore, several works explore the potential of multiple codebooks. VQ-VAE-2 [32] extends VQ-VAE to a multi-scale hierarchical organization. RQ-VAE [21] and MoVQ [54] aim to represent each feature as a stack of tokens, where RQ-VAE adopts an iterative way to factorize features into a series of residuals and MoVQ models features across multiple channels via specialized modulation.

Token-based Image Generation. Inspired by the success of GPT [2, 30, 31], VQ-VAE and most of its derivative works [10, 54] adopt AR transformers to model the token sequence. This approach leverages techniques from text generation to enhance IG performance. However, the decoding time of AR models scales linearly with the length of the token sequence. To accelerate decoding, MaskGIT [5] introduces a bidirectional transformer, referred to as the NAR proposal network.

Given the versatility of token-based modeling, both AR and NAR proposal networks can be easily extended to conditional IG scenarios. For instance, VQGAN uses a class token as the condition in its AR proposal network for class-conditional IG. With an NAR proposal network, MUSE [4] adopts text embeddings to predict masked image tokens in Text-to-Image generation.

3 Token-Based Image Generation

We start with the two-stage IG framework in Sec. 3.1. Subsequently, Sec. 3.2 details the architecture and training protocol for the tokenizers under consideration. Sec. 3.3 explains the evaluation benchmark. Sec. 3.4 further outlines our main observations derived from the IN-1k experiments. Lastly, we validate the observations under different settings in Sec. 3.5.

3.1 Two-Stage Image Generation

We illustrate the two-stage IG framework in Fig. 2. Given an image $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$, the encoder $\mathcal{E}$ converts this image into a feature map $x \in \mathbb{R}^{h \times w \times d}$, where $(h, w) = (H/f, W/f)$ and f is a downsample factor. Let a codebook $\mathbf{C}$ be a set of N code vectors $\{c_i\}_{i=1}^{N} \in \mathbb{R}^{N \times d}$, where each code vector $c_i \in \mathbb{R}^d$ corresponds to a specific code i. A quantizer $\mathcal{Q}$ then maps x into a sequence
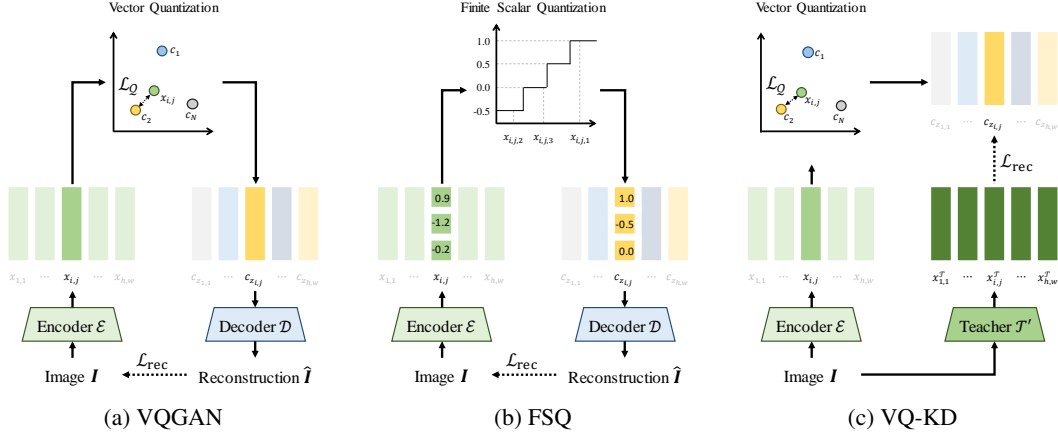
Figure 3: The architecture and training objective of different image tokenizers.

of codes $\mathbf{z} = \{z_i\}_{i=1}^{L}$, where $L = h \times w$ defines the sequence length and z_i is an integer that falls within the range $[1, N]$. Let c_{z_i} denote the code vector that corresponds to code z_i. Similarly, $\mathbf{C}(\mathbf{z}) = \{c_{z_i}\}_{i=1}^{L} \in \mathbb{R}^{L \times d}$ represents a sequence of code vectors associated with the code sequence $\mathbf{z}$. The encoder $\mathcal{E}$, quantizer $\mathcal{Q}$, and codebook $\mathbf{C}$ collectively form an image tokenizer $\mathcal{T}$.

The proposal network $\mathcal{P}$ models the distribution over $\mathbf{z}$, where the distribution is denoted as $p(\mathbf{z})$. Early proposal networks are implemented as an AR transformer, which sequentially models $p(z_i|z_{1:i-1})$ and formulates $p(\mathbf{z})$ as $\prod_{i=1}^{h \times w} p(z_i|z_{1:i-1})$. While the AR transformers can be trained in parallel, it has to sequentially decode $\mathbf{z}$ during inference, which renders it inefficient. Therefore, NAR proposal networks are being prevalent [5], which typically adopt bidirectional transformers to model $\mathbf{z}$. We denote the two types of proposal networks as $\mathcal{P}_{\text{AR}}$ and $\mathcal{P}_{\text{NAR}}$, respectively.

Finally, a decoder $\mathcal{D}$ maps the code vectors to the pixel space. In training, $\mathcal{D}$ takes $\mathbf{C}(\mathbf{z})$ as input and learns to reconstruct the original image $\mathbf{I}$ as $\hat{\mathbf{I}} \in \mathbb{R}^{H \times W \times 3}$. In inference, a sequence of codes $\hat{\mathbf{z}}$ is sampled from $p(\mathbf{z})$, translated to $\mathbf{C}(\hat{\mathbf{z}})$, and then fed into $\mathcal{D}$ to generate an image $\tilde{\mathbf{I}}$.

3.2 Image Tokenizers

In this paper, we focus on three types of image tokenizers: VQGAN [10], FSQ [25], and VQ-KD [27].

Let $x_{i,j} \in \mathbb{R}^d$ be a vector in the feature map x. As shown in Fig. 3a, to quantize $x_{i,j}$, the VQGAN tokenizer looks up the codebook $\mathbf{C}$ for the closest code vector in terms of Euclidean distance:

$$z_{i,j} = \arg \min_z \|x_{i,j} - c_z\|_2. \tag{1}$$

Since the quantization process is non-differentiable, VQGAN adopts the Straight-Through Estimator (STE) [1] to optimize the encoder $\mathcal{E}$, which copies gradients from $\mathbf{C}(\mathbf{z})$ to x. As a result, the codebook $\mathbf{C}$ receives no gradient. To optimize $\mathbf{C}$, VQGAN introduces a quantization loss $\mathcal{L}_{\mathcal{Q}}$:

$$\mathcal{L}_{\mathcal{Q}} = \|\text{sg}[x] - \mathbf{C}(\mathbf{z})\|_2^2 + \beta \|x - \text{sg}[\mathbf{C}(\mathbf{z})]\|_2^2, \tag{2}$$

where $\text{sg}[\cdot]$ denotes the stop-gradient operation and β is the loss weight. The first term is the codebook loss, which optimizes the codebook. The second term is the commitment loss to make sure the encoder $\mathcal{E}$ commits to a code vector [41]. Therefore, the overall loss for VQGAN is defined as:

$$\mathcal{L} = \mathcal{L}_{\mathcal{Q}} + \mathcal{L}_{\text{rec}}(\mathbf{I}, \hat{\mathbf{I}}), \tag{3}$$

where $\mathcal{L}_{\text{rec}}$ is the reconstruction loss between image $\mathbf{I}$ and reconstruction $\hat{\mathbf{I}}$, which includes ℓ_1 loss, perceptual loss, and adversarial loss.

Based on VQGAN, FSQ introduces a simpler image tokenizer, without the need for codebook lookup and quantization loss. As shown in Fig. 3b, FSQ adopts finite scalar quantization to quantize

4

Table 1: Comparison between image tokenizers on IN-1k.

Tokenizer $\mathcal{T}$	Codebook Usage (%)	rFID ↓	$\mathcal{P}_{AR}$			$\mathcal{P}_{NAR}$	
			PPL ↓	FID_{AR} ↓	IS_{AR}	FID_{NAR} ↓	IS_{NAR}
VQGAN	4.9	5.09	116.75	24.11	39.52	20.03	48.30
FSQ	**100.0**	4.96	791.56	40.17	26.40	29.78	33.63
VQ-KD$_{CLIP}$	**100.0**	4.96	**53.73**	11.78	**128.18**	9.51	**121.33**
VQ-KD$_{ViT}$	**100.0**	3.69	89.30	**11.40**	107.56	**8.45**	108.75
VQ-KD$_{DINO}$	**100.0**	**3.41**	74.07	13.15	80.89	10.21	91.39
VQ-KD$_{MAE}$	**100.0**	4.93	280.06	26.85	40.03	16.11	59.05

each channel of $x_{i,j}$ into a finite set of scalars. Since the quantization process involves no trainable parameter, FSQ can be trained with solely the reconstruction loss $\mathcal{L}_{rec}(\mathbf{I}, \hat{\mathbf{I}})$.

Unlike VQGAN and FSQ, which are designed for IG, VQ-KD was originally presented in BEiT v2 to provide supervision for IU models. As shown in Fig. 3c, VQ-KD is trained to reconstruct the feature map $x^{\mathcal{T}}$ encoded by a pretrained teacher $\mathcal{T}'$. Formally, the reconstruction loss is defined as:

$$\mathcal{L}_{rec} = -\cos\left(\sigma(\mathbf{C}(\mathbf{z})), x^{\mathcal{T}}\right), \tag{4}$$

where $\cos(\cdot, \cdot)$ represents cosine similarity and σ is a feature adapter. σ is implemented as a decoder, which maps $\mathbf{C}(\mathbf{z})$ to the same feature space as $x^{\mathcal{T}}$.

In this study, we examine VQ-KD using four types of pretrained teachers, including fully-supervised, text-supervised, contrastive, and Masked Image Modeling (MIM). We use VQ-KD$_{CLIP}$ and VQ-KD$_{DINO}$ to represent VQ-KD tokenizers trained with CLIP [29] and DINO [3] teachers, respectively. VQ-KD$_{MAE}$ and VQ-KD$_{ViT}$ represent tokenizers trained with MAE [13] and ViT [9] teachers. The latter two teachers are pretrained on IN-1k utilizing a ViT-B/16 architecture.

3.3 Benchmark

We detail how we fairly compare different tokenizers for token-based IG here.

For each tokenizer, we train a proposal network $\mathcal{P}$ and a decoder $\mathcal{D}$ to constitute an image generator. In training, the tokenizer is frozen to ensure fairness. Thus, $\mathcal{P}$ and $\mathcal{D}$ can be trained in parallel. We follow VQGAN [10] to train the AR proposal network and the decoder. The NAR proposal network is trained following MAGE [22]. Implementation details can be found in Appendix B.

Our benchmark adopts various metrics to comprehensively evaluate the image tokenizers. Given an image tokenizer, we assess the effectiveness of its encoding process by evaluating the codebook usage. To assess the generative capabilities of the image tokenizers, we evaluate IS [34] and FID [14] on the generated images $\tilde{\mathbf{I}}$. We assess the reconstruction capabilities of the image tokenizers by reporting the reconstruction FID (rFID). In addition, we present the PPL scores to appraise the AR modeling proficiency of the image tokenizers. A low PPL score implies that $\mathcal{P}_{AR}$ easily models $\mathbf{z}$. Details about the evaluation metrics can be found in Appendix C.

3.4 Main Observation

We evaluate the *class-conditional IG* performance of VQGAN, FSQ, and VQ-KD tokenizers on IN-1k. The results in Tab. 1 leads to the following observations.

VQ-KD significantly enhances generation quality over VQGAN. Equipped with either AR or NAR proposal networks, VQ-KD tokenizers consistently outperform VQGAN and FSQ, as evidenced by superior FID and IS metrics. In particular, VQ-KD$_{ViT}$ attains an FID_{AR} of 11.40 and an FID_{NAR} of 8.45, both less than half of those from VQGAN (24.11 FID_{AR} and 20.03 FID_{NAR}).

Tab. 2 presents a system-level comparison between VQ-KD$_{CLIP}$ and other class-conditional IG models on IN-1k at a resolution of 256×256. With a 1.4B AR proposal network, VQ-KD$_{CLIP}$ achieves an FID of 4.10, surpassing prior AR, NAR, and several VQ-based diffusion models.

Table 2: System level comparison on IN-1k.

Model	Architecture	#params	FID ↓
VQGAN [10]	AR	1.4B	15.78
RQ-VAE [21]	AR	1.4B	8.71
ViT-VQGAN [47]	AR	1.7B	5.30
MoVQ [54]	NAR	307M	8.78
MaskGIT [5]	NAR	227M	6.18
FSQ [25]	NAR	227M	4.53
LDM-8-G [33]	Diffusion	506M	7.76
CVQ-VAE [8]	Diffusion	400M	6.87
VQ-KD$_{\text{CLIP}}$ [27] (ours)	AR	1.4B	**4.10**

The superiority of VQ-KD is irrelevant to the quantization operation and codebook usage. Both VQ-KD$_{\text{CLIP}}$ and FSQ record 100.0% codebook usage and 4.96 rFID, but VQ-KD$_{\text{CLIP}}$ achieves lower FID$_{\text{AR}}$ and higher IS$_{\text{AR}}$ scores. Moreover, VQ-KD proves robustness to high codebook usage, with the PPL metric of most VQ-KD tokenizers surpassing VQGAN. In contrast, FSQ lags behind VQGAN in terms of PPL, suggesting that the high codebook usage of FSQ hinders $\mathcal{P}_{\text{AR}}$ from modeling the code sequence $\mathbf{z}$. As demonstrated in Sec. 4.1, this difference is likely due to the rich semantics in the VQ-KD feature map.

Tokenizers with stronger semantic understanding tend to deliver superior IG performance. Considering the FID and IS metrics, we find that VQ-KD tokenizers with supervised teachers (CLIP and ViT) consistently surpass those with unsupervised teachers (DINO and MAE). While VQ-KD$_{\text{DINO}}$ achieves the lowest rFID and PPL, its 13.15 FID$_{\text{AR}}$ is worse than VQ-KD$_{\text{CLIP}}$ (11.78) and VQ-KD$_{\text{ViT}}$ (11.40). This trend can be attributed to the superior capability of supervised models in capturing semantics compared to the unsupervised ones.

3.5 Further Verification

The superiority of VQ-KD holds across proposal networks. As seen in Tab. 1, all VQ-KD tokenizers surpass VQGAN and FSQ in the FID$_{\text{NAR}}$ and IS$_{\text{NAR}}$ metrics. In particular, VQ-KD$_{\text{ViT}}$ scores the lowest FID$_{\text{NAR}}$ at 8.45 and VQ-KD$_{\text{CLIP}}$ scores the highest IS$_{\text{NAR}}$ at 121.33. In contrast, VQGAN only achieves an FID$_{\text{NAR}}$ of 20.03 and an IS$_{\text{NAR}}$ of 48.30. It is also worth mentioning that VQ-KD$_{\text{CLIP}}$ and VQ-KD$_{\text{ViT}}$ show slightly better performance than VQ-KD$_{\text{DINO}}$ and VQ-KD$_{\text{MAE}}$, further supporting our conclusion that superior semantic understanding in supervised models plays a significant role in enhancing the quality of IG.

Table 3: Comparison between image tokenizers on MS-COCO. T2I experiments are conducted on the MS-COCO Captions dataset.

Tokenizer $\mathcal{T}$	Codebook Usage (%)	rFID ↓	$\mathcal{P}_{\text{AR}}$		FID$_{\text{T2I}}$ ↓
			PPL ↓	FID$_{\text{AR}}$ ↓	
VQGAN	2.4	16.21	**47.89**	38.43	24.11
FSQ	**100.0**	4.62	1040.02	44.64	23.36
VQ-KD$_{\text{CLIP}}$	82.2	5.48	72.31	29.80	**11.17**
VQ-KD$_{\text{ViT}}$	**100.0**	3.70	117.10	23.51	15.49
VQ-KD$_{\text{DINO}}$	**100.0**	**2.69**	129.93	**17.55**	11.50
VQ-KD$_{\text{MAE}}$	**100.0**	3.51	317.98	44.01	15.60

The superiority of VQ-KD holds across datasets. We conduct *unconditional IG* experiments on the MS-COCO dataset [23], which contains images of greater complexity than IN-1k. As demonstrated in Tab. 3, VQ-KD$_{\text{DINO}}$ achieves a FID$_{\text{AR}}$ metric of 17.55, significantly outperforming VQGAN (38.43) and FSQ (44.64). Since the ViT teacher in VQ-KD$_{\text{ViT}}$ is pretrained on IN-1k, the rFID and FID$_{\text{AR}}$ metrics of VQ-KD$_{\text{ViT}}$ are slightly inferior to VQ-KD$_{\text{DINO}}$. Note that the PPL metric of VQGAN is misleadingly favorable, due to its low codebook usage (2.4%).

The superiority of VQ-KD holds across tasks. *Text-to-Image* (T2I) experiments are conducted on the MS-COCO Captions dataset [6]. As shown in Tab. 3, the FID$_{\text{T2I}}$ metric of VQ-KD tokenizers range from 11.17 to 15.60, while VQGAN and FSQ only achieves 24.11 and 23.36, respectively.

4 Analysis

In this section, we analyze tokenizers based on feature reconstruction from various perspectives.

4.1 Codebook Visualization

We delve into the superiority of VQ-KD by visualizing its codebook. From IN-1k, we randomly choose four categories: golden retriever, pirate ship, valley, and sea anemone. For each image belonging to these categories, we deploy t-SNE [42] to project the feature map x and the code vectors $\mathbf{C}(\mathbf{z})$ into a two-dimensional space. x is colored according to the image category and $\mathbf{C}(\mathbf{z})$ is illustrated in black. As depicted in Fig. 4, the feature space of VQ-KD shows superior organization compared to VQGAN. In the VQ-KD feature space, x from the same category are grouped together. This implies that each code in the VQ-KD codebook conveys clear semantics. As codes with similar semantics are likely to present concurrently in an image, it becomes easier for the proposal network to model the code sequence $\mathbf{z}$. Conversely, each code in the VQGAN codebook is shared by multiple categories, resulting in semantics ambiguity. Hence, as illustrated in Tab. 1, the PPL metric for VQGAN is higher than VQ-KD, even though its codebook usage is considerably lower.
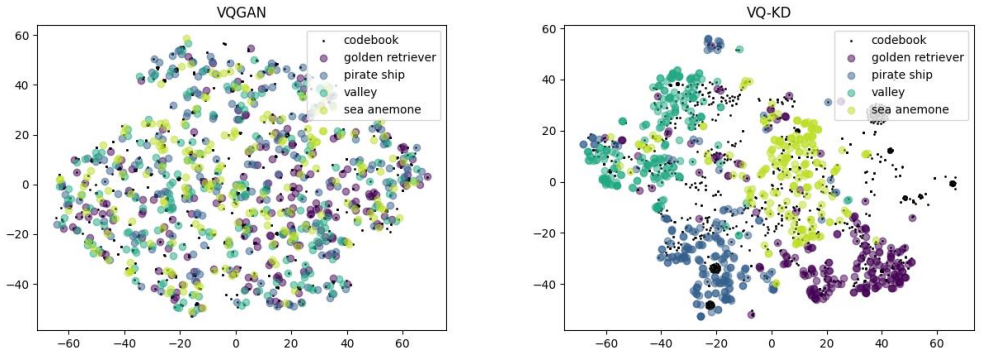

Figure 4: Codebook visualization of VQGAN and VQ-KD$_{\text{ViT}}$. Best viewed in color.

4.2 Clustering Pretrained Models as Tokenizers

To better harness the semantics in IU encoders, we propose a straightforward pipeline that transforms IU encoders into tokenizers via feature clustering. Given a pretrained IU model $\mathcal{T}'$, we employ it to encode the feature map $x^{\mathcal{T}}$ and subsequently utilize a clustering approach [55] to acquire N clusters. The cluster centroids constitute a codebook $\mathbf{C}$. $\mathcal{T}'$ remains frozen during training, which significantly accelerates the training process. As shown in Tab. 4, Cluster$_{\text{ViT}}$ presents 13.40 FID$_{\text{AR}}$, 10.58 FID$_{\text{T2I}}$, and 0.245 CLIP score on MS-COCO, outperforming all tokenizers in Tab. 3. This suggests that pretrained models with simple feature clustering can become good tokenizers. However, the cluster-based tokenizers behave worse in terms of rFID, since they encode little appearance detail in $x^{\mathcal{T}}$, which is essential for exact reconstruction. As a result, their FID and IS metrics on IN-1k are marginally weaker than those of their VQ-KD counterparts.

Table 4: Performance of cluster-based tokenizers.

Encoder $\mathcal{E}$	IN-1k			MS-COCO		
	rFID $\downarrow$	FID$_{\text{AR}}$ $\downarrow$	FID$_{\text{NAR}}$ $\downarrow$	rFID $\downarrow$	FID$_{\text{AR}}$ $\downarrow$	FID$_{\text{T2I}}$ $\downarrow$
CLIP	9.13	14.81	12.83	7.28	20.03	12.82
ViT	**4.78**	**11.87**	**8.59**	4.59	**13.40**	10.58
DINO	5.16	14.53	11.23	**4.02**	25.35	**7.66**
MAE	15.15	38.72	34.26	10.08	62.17	18.85

7

4.3 Scaling Up the Proposal Network

We examine the IG performance of tokenizers with a large-scale proposal network. Following VQGAN, we adopt GPT-2 XL as $\mathcal{P}_{AR}$, which comes with 1.4B parameters. In line with Tab. 1, VQ-KD$_{ViT}$ leads with 9.23 FID$_{AR}$, while VQ-KD$_{CLIP}$ achieves the highest IS metric at 150.63. Upon comparing with Tab. 1, tokenizers with stronger IU capabilities exhibit less improvement in the FID$_{AR}$ metric. For instance, VQ-KD$_{MAE}$ improves significantly from 26.85 to 17.11, while VQ-KD$_{CLIP}$ reveals a marginal enhancement from 11.78 to 11.27. This suggests that a small-scale $\mathcal{P}_{AR}$ is sufficient for tokenizers with strong IU capabilities, whereas those with weaker IU abilities benefit from a large-scale $\mathcal{P}_{AR}$.

Table 5: AR modeling with a large-scale proposal network or strong data augmentation.

Tokenizer $\mathcal{T}$	GPT-2 XL		Strong Aug.	
	FID$_{AR}$ ↓	IS$_{AR}$	FID$_{AR}$ ↓	IS$_{AR}$
VQGAN	17.13	59.19	32.76	26.76
FSQ	25.87	49.59	52.17	18.09
VQ-KD$_{CLIP}$	11.27	**150.63**	15.24	**90.41**
VQ-KD$_{ViT}$	**9.23**	146.00	**13.32**	81.44
VQ-KD$_{DINO}$	9.50	120.26	19.39	52.74
VQ-KD$_{MAE}$	17.11	69.03	36.63	28.49
Cluster$_{CLIP}$	14.00	110.26	17.22	83.72

4.4 Influence of Strong Data Augmentation

We investigate the impact of strong data augmentation on the AR modeling performance of tokenizers. Specifically, we employed a strong random crop, where the crop scale ranges from 0.08 to 1.0, introducing greater variability into the training data. As shown in Tab. 5, all tokenizers exhibit worse FID$_{AR}$ metrics than their counterparts in Tab. 1. Interestingly, tokenizers with stronger IU capabilities demonstrate greater robustness to the strong data augmentation. For instance, VQ-KD$_{ViT}$ experiences a minor increase in FID$_{AR}$ of just 1.92 (from 11.40 to 13.32), whilst VQ-KD$_{MAE}$ records a considerable leap of 9.78 (from 26.85 to 36.63).

4.5 Large Teacher Models in VQ-KD

We incorporate OpenCLIP [17] models of varying sizes as teacher models to train the VQ-KD tokenizers. As illustrated in Tab. 6, the FID$_{AR}$ metric sees a reduction from 10.31 to 8.70 when the size of the OpenCLIP model escalates from ViT-L/14 to ViT-G/14. Given that larger OpenCLIP models inherently possess stronger IU capabilities, these findings further corroborate the superiority of image tokenizers with more potent IU capabilities.

Table 6: Effect of different teachers in VQ-KD.

OpenCLIP	rFID ↓	PPL ↓	FID$_{AR}$ ↓	IS$_{AR}$
ViT-L/14	4.03	80.56	10.31	146.21
ViT-H/14	**3.60**	97.32	9.64	**161.13**
ViT-G/14	3.80	**77.79**	**8.70**	152.71

Table 7: Effect of codebook size and dimension. Experiments are conducted on VQ-KD$_{CLIP}$.

(a) Codebook size.

Codebook $\mathbf{C}$		rFID ↓	FID$_{AR}$ ↓	IS$_{AR}$
Size (log$_2$)	Usage (%)			
10	**100.0**	6.59	11.65	114.90
11	**100.0**	5.99	**10.98**	119.72
12	**100.0**	5.64	11.71	123.42
13	**100.0**	4.96	11.78	128.18
14	93.7	**4.53**	11.61	**131.51**

(b) Codebook dimension.

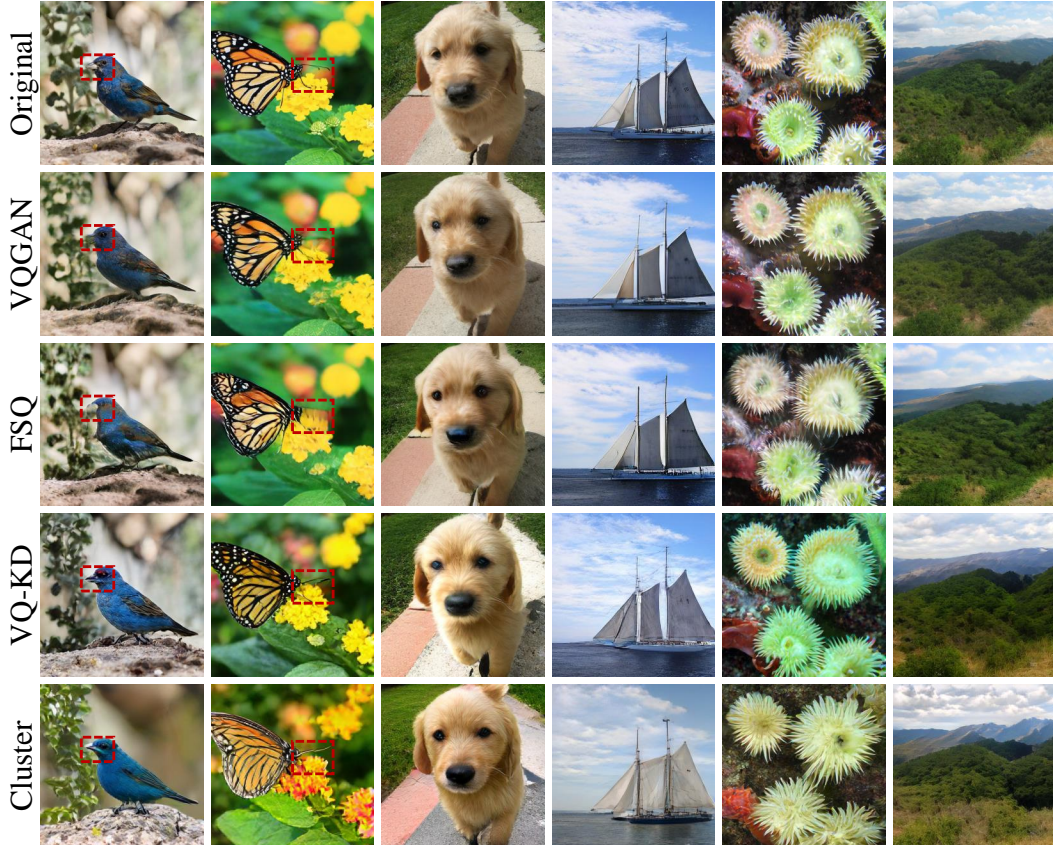| Codebook $\mathbf{C}$		rFID ↓	FID$_{AR}$ ↓	IS$_{AR}$
Dim	Usage (%)			
16	**100.0**	4.94	12.19	124.71
32	**100.0**	4.96	11.78	**128.18**
64	97.4	**4.64**	11.00	126.60
128	89.6	5.03	**10.50**	119.78
256	48.7	6.80	12.08	103.07

Figure 5: Reconstruction results of different image tokenizers.

4.6 Codebook Size and Dimension

The size and dimension of the codebook exert a significant influence on the IG performance of tokenizers [10, 47]. Tab. 7a showcases the performance of VQ-KD$_{\text{CLIP}}$ with varying codebook sizes. Large codebooks aid the tokenizers in representing fine-grained semantics, contributing to a consistent decrease in the rFID metric from 6.59 to 4.53. The IS metric also shows favor towards larger codebooks, with size 2^{14} leading to the highest IS metric of 131.51. However, choosing the correct code from a large codebook is harder than from a small codebook, hindering $\mathcal{P}_{\text{AR}}$ from achieving lower FID scores with larger codebooks.

Tab. 7b demonstrates the influence of codebook dimension. High-dimensional codes carry more information but lead to lower codebook usage. As a result, the rFID metric initially drops from 4.96 to 4.64, then increases drastically to 6.80. Similar to Tab. 7a, the FID and IS metrics favor different codebook dimensions. FID favors 128-dimensional codebooks, where codebook usage is relatively low. In contrast, IS favors 32-dimensional codebooks, possibly due to a superior diversity.

4.7 Qualitative Analysis

The reconstruction quality of various tokenizers is demonstrated in Fig. 5. Original images are displayed in the first row. Regions where VQGAN and FSQ fail to reconstruct are highlighted with red boxes. In contrast, VQ-KD reconstructions are visually more accurate. Since the IU encoder in Cluster emphasizes encoding semantics over visual details, Cluster fails to preserve all visual details during reconstruction. Nonetheless, the reconstruction results of Cluster still appear more natural than VQGANand FSQ, especially in the highlighted areas. Fig. 6 and Fig. 7 further illustrates the AR and NAR generation results, showcasing the superior visual performance of VQ-KD and Cluster.

9

Figure 6: Class-conditional AR generation results of different image tokenizers.
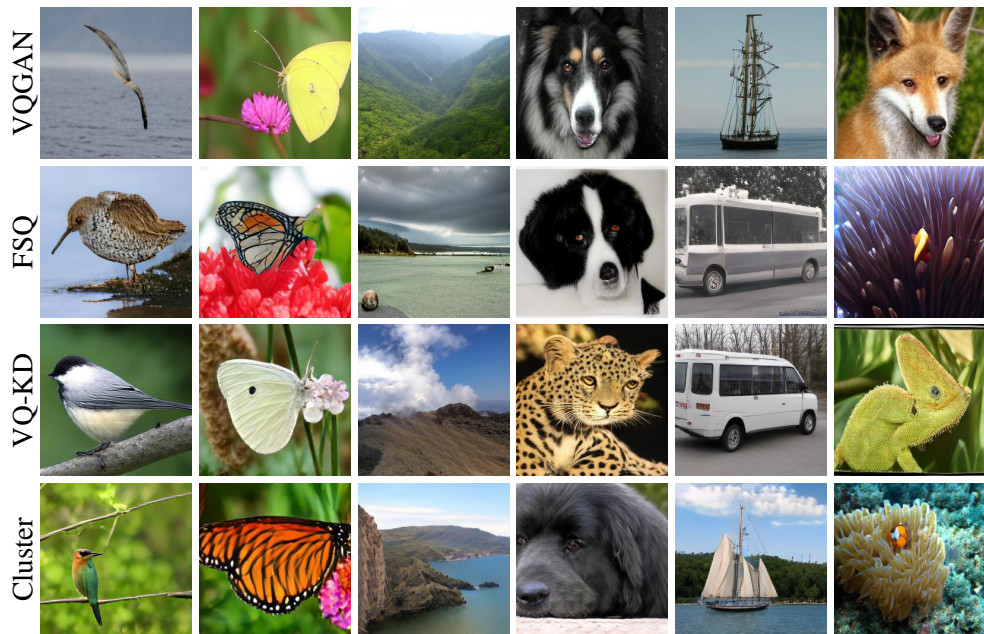

Figure 7: Unconditional NAR generation results of different image tokenizers.

5 Conclusion

In this paper, we show that image understanding (IU) models can be useful in image generation (IG). Through comprehensive studies, observe that the VQ-KD tokenizers significantly enhance generation quality over VQGAN, irrelevant of the quantization operation and codebook utilization. Within the VQ-KD tokenizers, stronger IU capabilities tends to deliver superior IG performance. Further verification shows that the superiority of VQ-KD holds across proposal networks, datasets, and tasks. Lastly, we analyze VQ-KD from multiple angles, including clustering pretrained models as tokenizers, scaling up the proposal network, influence of strong data augmentation, large teacher models in VQ-KD, and codebook size and dimensions.

Acknowledgement

References

[1] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *arXiv preprint arXiv:1308.3432*, 2013.

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. In *NeurIPS*, 2020.

[3] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jegou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging Properties in Self-Supervised Vision Transformers. In *ICCV*, 2021.

[4] Huiwen Chang, Han Zhang, Jarred Barber, AJ Maschinot, Jose Lezama, Lu Jiang, Ming-Hsuan Yang, Kevin Murphy, William T. Freeman, Michael Rubinstein, Yuanzhen Li, and Dilip Krishnan. Muse: Text-To-Image Generation via Masked Generative Transformers. In *ICML*, 2023.

[5] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. MaskGIT: Masked Generative Image Transformer. In *CVPR*, 2022.

[6] Xinlei Chen, Hao Fang, Tsung-yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollar, and C Lawrence Zitnick. Microsoft COCO Captions: Data Collection and Evaluation Server. *arXiv preprint arXiv:1504.00325*, 2015.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009.

[8] Prafulla Dhariwal and Alex Nichol. Diffusion Models Beat GANs on Image Synthesis. In *NeurIPS*, volume 11, pages 8780–8794, 2021.

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*, 2021.

[10] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming Transformers for High-Resolution Image Synthesis. In *CVPR*, 2021.

[11] R. Gray. Vector quantization. *IEEE ASSP Mag.*, 1(2), 1984.

[12] Yuchao Gu, Xintao Wang, Yixiao Ge, Ying Shan, Xiaohu Qie, and Mike Zheng Shou. Rethinking the Objectives of Vector-Quantized Tokenizers for Image Synthesis. *arXiv preprint arXiv:2212.03185*, 2022.

[13] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked Autoencoders Are Scalable Vision Learners. In *CVPR*, 2022.

[14] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *NeurIPS*, 2017.

[15] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. In *NeurIPS*, 2020.

[16] Drew A. Hudson, Daniel Zoran, Mateusz Malinowski, Andrew K. Lampinen, Andrew Jaegle, James L. McClelland, Loic Matthey, Felix Hill, and Alexander Lerchner. SODA: Bottleneck Diffusion Models for Representation Learning. *arXiv preprint arXiv:2311.17901*, 2023.

[17] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip. `https://doi.org/10.5281/zenodo.5143773`, 2021.

[18] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[19] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *ICLR*, 2014.

[20] Dan Kondratyuk, Lijun Yu, Xiuye Gu, José Lezama, Jonathan Huang, Rachel Hornung, Hartwig Adam, Hassan Akbari, Yair Alon, Vighnesh Birodkar, Yong Cheng, Ming-Chang Chiu, Josh Dillon, Irfan Essa, Agrim Gupta, Meera Hahn, Anja Hauth, David Hendon, Alonso Martinez, David Minnen, David Ross, Grant Schindler, Mikhail Sirotenko, Kihyuk Sohn, Krishna Somandepalli, Huisheng Wang, Jimmy Yan, Ming-Hsuan Yang, Xuan Yang, Bryan Seybold, and Lu Jiang. VideoPoet: A Large Language Model for Zero-Shot Video Generation. *arXiv preprint arXiv:2312.14125*, 2023.

[21] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook Shin Han. Autoregressive Image Generation using Residual Quantization. In *CVPR*, 2022.

[22] Tianhong Li, Huiwen Chang, Shlok Kumar Mishra, Han Zhang, Dina Katabi, and Dilip Krishnan. MAGE: MAsked Generative Encoder to Unify Representation Learning and Image Synthesis. In *CVPR*, 2023.

[23] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common Objects in Context. In *ECCV*, 2014.

[24] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

[25] Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite Scalar Quantization: VQ-VAE Made Simple. In *ICLR*, 2024.

[26] Soumik Mukhopadhyay, Matthew Gwilliam, Vatsal Agarwal, Namitha Padmanabhan, Archana Swaminathan, Srinidhi Hegde, Tianyi Zhou, and Abhinav Shrivastava. Diffusion Models Beat GANs on Image Classification. *arXiv preprint arXiv:2307.08702*, 2023.

[27] Zhiliang Peng, Li Dong, Hangbo Bao, Qixiang Ye, and Furu Wei. BEiT v2: Masked Image Modeling with Vector-Quantized Visual Tokenizers. *arXiv preprint arXiv:2208.06366*, 2022.

[28] Konpat Preechakul, Nattanat Chatthee, Suttisak Wizadwongsa, and Supasorn Suwajanakorn. Diffusion Autoencoders: Toward a Meaningful and Decodable Representation. In *CVPR*, 2022.

[29] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. In *ICML*, volume 139, 2021.

[30] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving Language Understanding by Generative Pre-Training. 2018.

[31] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. *arXiv preprint arXiv:2007.07582*, 2019.

[32] Ali Razavi, Aäron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with VQ-VAE-2. In *NeurIPS*, 2019.

[33] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjorn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models. In *CVPR*, 2022.

[34] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved Techniques for Training GANs. In *NeurIPS*, 2016.

[35] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In *ICML*, 2015.

[36] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising Diffusion Implicit Models. In *ICLR*, 2020.

[37] Yang Song and Stefano Ermon. Generative Modeling by Estimating Gradients of the Data Distribution. In *NeurIPS*, 2019.

[38] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. In *CVPR*, 2016.

[39] Yuhta Takida, Yukara Ikemiya, Takashi Shibuya, Kazuki Shimada, Woosung Choi, Chieh-Hsin Lai, Naoki Murata, Toshimitsu Uesaka, Kengo Uchida, Wei-Hsiang Liao, and Yuki Mitsufuji. HQ-VAE: Hierarchical Discrete Representation Learning with Variational Bayes. *arXiv preprint arXiv:2401.00365*, 2023.

[40] Yuhta Takida, Takashi Shibuya, Wei Hsiang 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. *Proc. Mach. Learn. Res.*, 162, 2022.

[41] Aaron Van Den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *NeurIPS*, 2017.

[42] Laurens Van Der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *JMLR*, 9, 2008.

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *NeurIPS*, 2017.

[44] Tung-Long Vuong, Trung Le, He Zhao, Chuanxia Zheng, Mehrtash Harandi, Jianfei Cai, and Dinh Phung. Vector Quantized Wasserstein Auto-Encoder. *ICML*, 2023.

[45] Weijia Wu, Yuzhong Zhao, Hao Chen, Yuchao Gu, Rui Zhao, Yefei He, Hong Zhou, Mike Zheng Shou, and Chunhua Shen. DatasetDM: Synthesizing Data with Perception Annotations Using Diffusion Models. In *NeurIPS*, 2023.

[46] Lihe Yang, Xiaogang Xu, Bingyi Kang, Yinghuan Shi, and Hengshuang Zhao. FreeMask: Synthetic Images with Dense Annotations Make Stronger Segmentation Models. In *NeurIPS*, 2023.

[47] Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. Vector-Quantized Image Modeling With Improved Vqgan. In *ICLR*, 2022.

[48] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, Ben Hutchinson, Wei Han, Zarana Parekh, Xin Li, Han Zhang, Jason Baldridge, and Yonghui Wu. Scaling Autoregressive Models for Content-Rich Text-to-Image Generation. *TMLR*, 2022.

[49] Lijun Yu, José Lezama, Nitesh B. Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Agrim Gupta, Xiuye Gu, Alexander G. Hauptmann, Boqing Gong, Ming-Hsuan Yang, Irfan Essa, David A. Ross, and Lu Jiang. Language Model Beats Diffusion – Tokenizer is Key to Visual Generation. *arXiv preprint arXiv:2310.05737*, 2023.

[50] Lili Yu, Bowen Shi, Ramakanth Pasunuru, Benjamin Muller, Olga Golovneva, Tianlu Wang, Arun Babu, Binh Tang, Brian Karrer, Shelly Sheynin, Candace Ross, Adam Polyak, Russell Howes, Vasu Sharma, Puxin Xu, Hovhannes Tamoyan, Oron Ashual, Uriel Singer, Shang-Wen Li, Susan Zhang, Gargi Ghosh, Yaniv Taigman, Maryam Fazel-Zarandi, Asli Celikyilmaz, Luke Zettlemoyer, and Armen Aghajanyan. Scaling Autoregressive Multi-Modal Models: Pretraining and Instruction Tuning. *arXiv preprint arXiv:2309.02591*, 2023.

[51] Manlin Zhang, Jie Wu, Yuxi Ren, Ming Li, Jie Qin, Xuefeng Xiao, Wei Liu, Rui Wang, Min Zheng, and Andy J. Ma. DiffusionEngine: Diffusion Model is Scalable Data Engine for Object Detection. *arXiv preprint arXiv:2309.03893*, 2023.

[52] Zijian Zhang, Zhou Zhao, and Zhijie Lin. Unsupervised Representation Learning from Pretrained Diffusion Probabilistic Models. In *NeurIPS*, volume 35, 2022.

[53] Wenliang Zhao, Yongming Rao, Zuyan Liu, Benlin Liu, Jie Zhou, and Jiwen Lu. Unleashing Text-to-Image Diffusion Models for Visual Perception. In *ICCV*, 2023.

[54] Chuanxia Zheng, Jianfei Cai, Long Tung Vuong, and Dinh Phung. MoVQ: Modulating Quantized Vectors for High-Fidelity Image Generation. In *NeurIPS*, 2022.

[55] Chuanxia Zheng and Andrea Vedaldi. Online Clustered Codebook. In *CVPR*, 2023.

A Datasets

The experiments are conducted on two image datasets: ImageNet-1k (IN-1k) [7] and MS-COCO [23]. The IN-1k dataset contains approximately 1.28 million training images and $50,000$ validation images across $1,000$ diverse categories. The MS-COCO dataset comprises $82,783$ images for training and $40,504$ for validation. Each image is annotated with several captions.

For a given image, we first resize its shorter side to s pixels, where s symbolizes the input size. Subsequently, a central crop is performed to derive an image fragment sized $s \times s$ pixels. Our default augmentation strategy incorporates a random crop (between 0.8 and 1.0) partnered with random horizontal flipping.

B Implementation Details

The image tokenizers under consideration generate token sequences of length 256 upon a 256×256 input image. All tokenizers remain frozen throughout the training of the decoder and the proposal networks. Experiments are performed using 8 A100 80GB GPUs. The approximate training times for the VQ-KD tokenizer is around 30 hours, the decoder requires roughly 68 hours, while the AR proposal network and NAR proposal network necessitate about 29 hours and 72 hours, respectively. In total, a single experiment takes approximately 200 hours training.

The CNN-based VQGAN tokenizers, with 27.9M parameters, are trained following the identical procedure employed for decoders. The codebook dimension of VQGAN is 256. FSQ levels $\mathcal{L}$ are set to $(8, 8, 5, 5, 5)$, equivalent to codebook size $8,000$.

As per BEiT v2 [27], an AdamW optimizer is utilized to train the VQ-KD tokenizers. The learning rate warms up linearly to 10^{-4} for $25,000$ steps, subsequently decaying to 10^{-5} under a cosine schedule. Unless specifically stated, VQ-KD tokenizer is trained with an input size of 224×224 and codebook dimension of 32.

VQGAN [10] is adopted for training both the decoder and the AR proposal networks. Both $\mathcal{D}$ and $\mathcal{P}_{AR}$ training span $260,000$ steps with a collective batch size of 96 for IN-1k and 24 for MS-COCO. The decoder is a CNN-based VQGAN decoder, consisting of 40.5M parameters. The decoders utilize the Adam [18] optimizer with learning rates set at 5.4×10^{-5}, $\beta_1 = 0.5$, and $\beta_2 = 0.9$. Their discriminators are also trained via Adam optimizer, employing learning rates of 4.32×10^{-4}, while keeping the β constants identical. Subsequent training of AR proposal networks relies on the AdamW [24] optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$, and a 0.2 weight decay. An initial learning rate of 10^{-4} is set, after which it decays to 0 on a cosine schedule. The AR proposal network is a GPT-2 Medium [31], with 335M parameters.

We follow MAGE [22] for training NAR proposal networks. $\mathcal{P}_{NAR}$ is trained for 300 epochs with a collective batch size of 512 on ImageNet-1k. NAR proposal networks are trained with the AdamW optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and a 0.05 weight decay. The learning rate warms up linearly to 3×10^{-4} throughout 10 epochs before decaying to 0 following a cosine schedule. The encoder of $\mathcal{P}_{NAR}$ is a ViT-B/16, with 86M parameters.

C Evaluation

Codebook usage is defined as the proportion of codes from the codebook that have been used at least once when encoding the dataset. A low value for codebook usage might be an indication of the 'codebook collapse' issue.

IS provides a measure of both the fidelity and diversity of $\tilde{\mathbf{I}}$. However, IS significantly relies on the classification capabilities of a pretrained Inception-v3 model [38]. Complex images are likely to be misinterpreted as lacking fidelity by IS. Therefore, we limit the use of IS to IN-1k experiments only.

To circumvent the limitations of IS, FID computes the statistical distance in the Inception-v3 feature space between the real images $\mathbf{I}$ and the generated images $\tilde{\mathbf{I}}$. A lower FID score indicates that $\tilde{\mathbf{I}}$ is statistically similar to $\mathbf{I}$.

rFID is defined as the FID between $\mathbf{I}$ and their reconstructed counterparts $\hat{\mathbf{I}}$. Obtaining a low rFID score requires that the image tokenizer encode sufficient visual details within the codes $\mathbf{C}(\mathbf{z})$ to enable accurate reconstruction by the decoder.

The PPL score is defined as:

$$\text{PPL} = \exp\left(-\frac{1}{L}\sum_{i=1}^{L}\log p(z_i|z_{1:i-1})\right),\tag{5}$$

where $\mathbf{z}$ denotes a sequence of codes offered by the tokenizer, L represents the length of $\mathbf{z}$, and $p(\mathbf{z})$ embodies the distribution modeled by the AR proposal network $\mathcal{P}_{\text{AR}}$.

Both reconstruction and AR modeling serve as two pivotal capabilities in an image generator. We anticipate that these metrics will lead to a more thorough insight into the generative capacities of image tokenizers.

D   Limitations

The VQ-KD tokenizers are designed to mimic the IU encoders, yielding superior quantitative results compared to traditional tokenizers like the VQGAN and FSQ. Nonetheless, qualitative analysis suggest that the VQ-KD may modify visual details during the pixel reformation process, thereby posing challenges for tasks such as image editing.

E   Broader Impacts

This paper explores the question *how might image understanding (IU) models aid image generation (IG) tasks*. We envision that our findings will motivate research on image tokenizers and prompt the community to reconsider the correlation between IU and IG.

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: The contributions and scope of the paper are summarized in the last paragraph of sec. 1.

   Guidelines:

   - The answer NA 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 NA 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 of the work is discussed in appendix D.

   Guidelines:

   - The answer NA 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: [NA]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer NA 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 model architecture is demonstrated in sec. 3.1. The implementation details are illustrated in appendix B. The code is released at `https://github.com/magic-research/vector_quantization`.

Guidelines:

- The answer NA 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: The code is released at `https://github.com/magic-research/vector_quantization`. The data used are publicly available.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.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://nips.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, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Implementation details are illustrated in appendix B.

Guidelines:

- The answer NA 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: [No]

Justification: Given the massive amount of experiments conducted in this paper, providing error bars would be computationally prohibitive.

Guidelines:

- The answer NA 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: Compute resources are described in appendix B.

   Guidelines:
   - The answer NA 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: The paper conforms with the NeurIPS Code of Ethics.

   Guidelines:
   - The answer NA 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: [Yes]

    Justification: Potential societal impacts of the work are discussed in appendix E.

    Guidelines:
    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA 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., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA 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: The paper mainly uses the BEiT v2 repository, which is cited in the paper and mentioned in the code.

    Guidelines:

    - The answer NA 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: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA 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: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA 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: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA 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.