Not All Image Regions Matter: Masked Vector Quantization for Autoregressive Image Generation

Mengqi Huang[1], Zhendong Mao[1, 3]*, Quan Wang[2], Yongdong Zhang[1, 3]
[1]University of Science and Technology of China, Hefei, China; [2]MOE Key Laboratory of Trustworthy, Distributed Computing and Service, Beijing University of Posts and Telecommunications, Beijing, China;
[3]Institute of Artificial intelligence, Hefei Comprehensive National Science Center, Hefei, China
huangmq@mail.ustc.edu.cn, {zdmao, zhyd73}@ustc.edu.cn, wangquan@bupt.edu.cn

Abstract

Existing autoregressive models follow the two-stage generation paradigm that first learns a codebook in the latent space for image reconstruction and then completes the image generation autoregressively based on the learned codebook. However, existing codebook learning simply models all local region information of images without distinguishing their different perceptual importance, which brings redundancy in the learned codebook that not only limits the next stage's autoregressive model's ability to model important structure but also results in high training cost and slow generation speed. In this study, we borrow the idea of importance perception from classical image coding theory and propose a novel two-stage framework, which consists of Masked Quantization VAE (MQ-VAE) and Stackformer, to relieve the model from modeling redundancy. Specifically, MQ-VAE incorporates an adaptive mask module for masking redundant region features before quantization and an adaptive de-mask module for recovering the original grid image feature map to faithfully reconstruct the original images after quantization. Then, Stackformer learns to predict the combination of the next code and its position in the feature map. Comprehensive experiments on various image generation validate our effectiveness and efficiency. Code will be released at https://github.com/CrossmodalGroup/MaskedVectorQuantization.

1. Introduction

Deep generative models of images have received significant improvements over the past few years and broadly fall into two categories: likelihood-based models, which include VAEs [24], flow-based [36], diffusion models [17] and autoregressive models [40], and generative adversarial networks (GANs) [14], which use discriminator networks to distinguish samples from generator networks and real examples. Compared with GANs, likelihood-based models' training objective, *i.e.*, the negative log-likelihood (NLL) or its upper bound, incentives learning the full data distribution and allows for detecting overfitting.

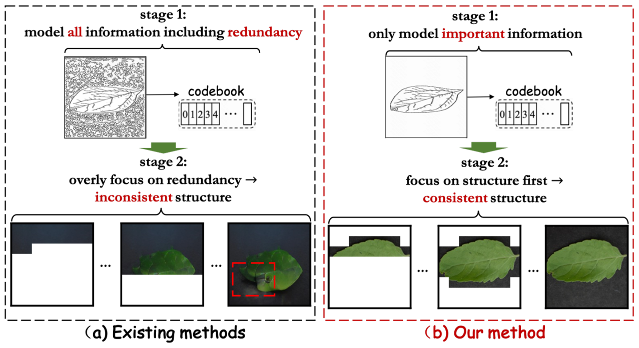

Figure 1. Illustration of our motivation. (a) Existing works model all local regions without distinguishing their perceptual importance in stage 1, which not only brings redundancy (*e.g.*, the textural regions like the background) in the learned codebook but also make the autoregressive models overly focus on modeling this redundancy and hinder other important structural regions modeling. (b) The codebook learning in our method only includes the important regions, *e.g.*, the structural regions like corners and edges, since other unimportant ones can be restored even if missing, and thus autoregressive model could focus on modeling these important regions in stage 2 and results in better generation quality.

Among the likelihood-based models, autoregressive models have recently attracted increasing attention for their impressive modeling ability and scalability. Recent autoregressive image generation [10, 12, 13, 28, 28, 34, 35, 37, 39] follows the two-stage generation paradigm, *i.e.*, the first stage learns a codebook in the latent space for image reconstruction and the second stage completes the image generation in the raster-scan [13] order by autoregressive models

*Zhendong Mao is the corresponding author.

based on the learned codebook. Since codebook learning in the first stage defines the discrete image representation for the next autoregressive modeling, a high-quality codebook is the key to generate high-quality images. Several recent works focus on improving the codebook learning in the first stage, *e.g.*, VQGAN [13] introduces adversarial loss and perceptual loss. ViT-VQGAN [42] introduces a more expressive transformer backbone. RQ-VAE [28] introduces the residual quantization to reduce the resolution of the latent space. In general, the essence of existing codebook learning is the modeling of all local region information (*i.e.*, an 8×8 or 16×16 patch) of images in the dataset, without distinguishing their different perceptual importance.

In this study, we point out that existing codebook learning exists gaps with classical image coding theory [20, 25, 26], the basic idea of which is to remove redundant information by perceiving the importance of different regions in images. The image coding theory reveals that an ideal image coding method should only encode images' perceptually important regions (*i.e.*, which cannot be restored if missing) while discarding the unimportant ones (*i.e.*, which can be restored by other image regions even if missing). The neglect of considering such perceptual importance in existing works poses problems in two aspects, as illustrated in Figure 1(a): (1) the existence of this large amount of repetitive and redundant information brings redundancy to the learned codebook, which further makes the autoregressive model in the next stage overly focus on modeling this redundancy while overlooking other important regions and finally degrades generation quality. (2) the redundancy makes the autoregressive model need to predict more (redundant) quantized codes to generate images, which significantly increases the training cost and decreases the generating speed. Although the effectiveness and efficiency of image coding theory have been widely validated, how to introduce this idea into codebook learning remains unexplored.

The key of applying image coding theory to codebook learning is to distinguish important image parts from unimportant ones correctly. Considering that the essential difference between these two sets lies in whether they can be restored if missing, we found that this distinction can be realized through the mask mechanism, *i.e.*, the masked part is important if it cannot be faithfully restored, and otherwise unimportant. Based on the above observation, we thereby propose a novel two-stage generation paradigm upon the mask mechanism to relieve the model from modeling redundant information. Specifically, we first propose a *Masked Quantization VAE (MQ-VAE)* with two novel modules, *i.e.*, an *adaptive mask module* for adaptively masking redundant region features before quantization, and an *adaptive de-mask module* for adaptively recovering the original grid image feature map to faithfully reconstruct original images after quantization. As for the *adaptive mask module*, it incorporates a lightweight content-aware scoring network that learns to measure the importance of each image region feature. The features are then ranked by the importance scores and only a subset of high-scored features will be quantized further. As for the *adaptive de-mask module*, we design a direction-constrained self-attention to encourage the information flow from the unmasked regions to the masked regions while blocking the reverse, which aims to infer the original masked region information based on unmasked ones. Thanks to the adaptive mask and de-mask mechanism, our MQ-VAE removes the negative effects of redundant image regions and also shortens the sequence length to achieve both effectiveness and efficiency.

Moreover, since different images have different important regions, the position of quantized codes in the feature map also dynamically changed. Therefore, we further propose *Stackformer* for learning to predict the combination of both codes and their corresponding positions. Concretely, the proposed *Stackformer* stacks a Code-Transformer and a Position-Transformer, where the Code-Transformer learns to predict the next code based on all previous codes and their positions, and the Position-Transformer learns to predict the next code's position based on all previous codes' positions and *current* code.

With our method, as shown in Figure 1(b), the codebook learning only includes the important regions, *e.g.*, the structural regions, since unimportant ones like the background can be restored even if missing. And therefore the autoregressive model in the second stage could focus on modeling these important regions and brings better generation quality.

In a nutshell, we summarize our main contributions as:

Conceptually, we point out that existing codebook learning ignores distinguishing the perceptual importance of different image regions, which brings redundancy that degrades generation quality and decreases generation speed.

Technically, (i) we propose MQ-VAE with a novel *adaptive mask module* to mask redundant region features before quantization and a novel *adaptive de-mask module* to recover the original feature map after quantization; (ii) we propose a novel *Stackformer* to predict the combination of both codes and their corresponding positions.

Experimentally, comprehensive experiments on various generations validate our effectiveness and efficiency, *i.e.*, we achieve 8.1%, 2.3%, and 18.6% FID improvement on un-, class-, and text-conditional state-of-the-art at million-level parameters, and faster generation speed compared to existing autoregressive models.

2. Related Work

2.1. Autoregressive Modeling for Image Generation

Autoregressive models for image generation have recently attracted increasing research attention and have

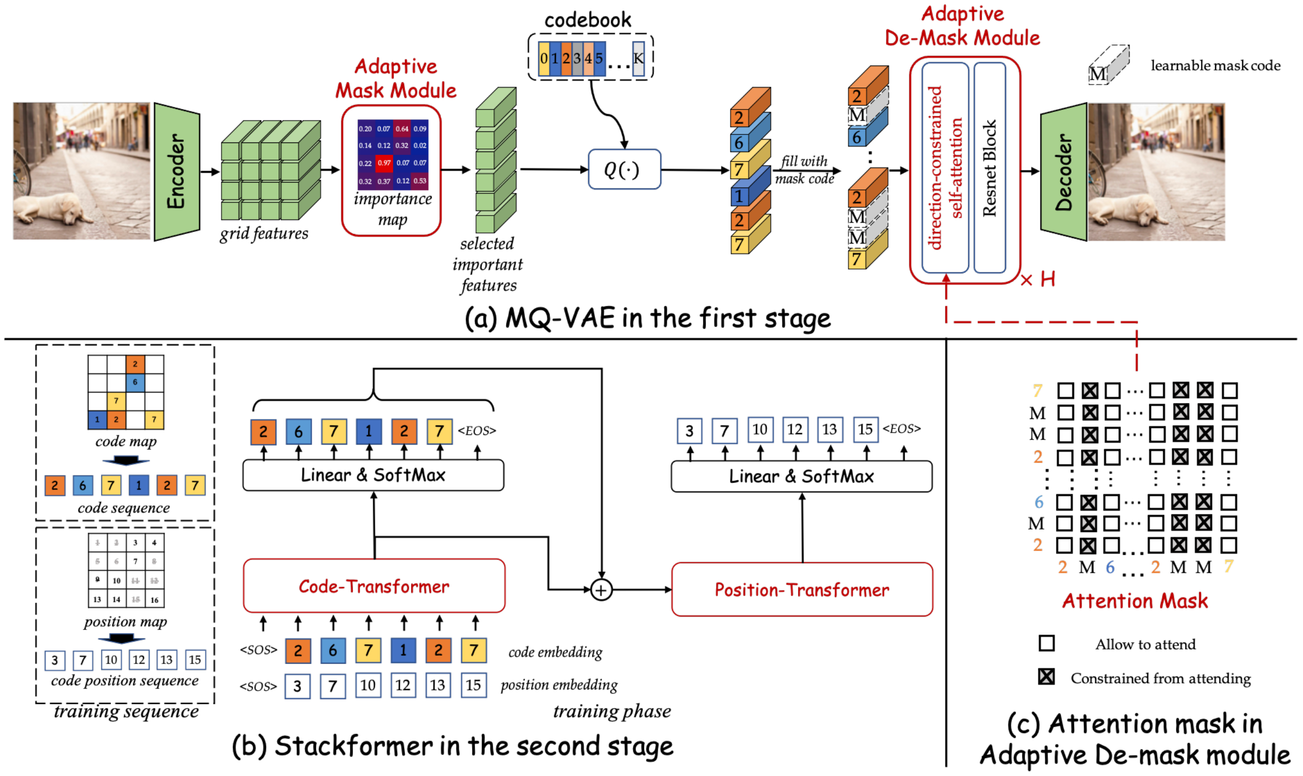

Figure 2. The Illustration of our proposed two-stage generation framework. (a) In the first stage, MQ-VAE adaptively masks the redundant region features to prevent redundant codes while keeping important ones, which ensures that the original images can still be faithfully recovered. Here, $\frac{10}{16}$ regions are masked and $\frac{6}{16}$ regions are kept. (b) In the second stage, Stackformer stacks a Code-Transformer and a Position-Transformer to autoregressively predict the next code and its position in the original 2D feature map, respectively. (c) The attention mask of the proposed direction-constrained self-attention in the adaptive de-mask module for inferring masked regions features.

shown impressive results [10, 12, 13, 32, 34, 35, 37, 39, 42] among various generation tasks. Early autoregressive models [5, 33, 40] directly optimizing the likelihood of raw image pixels, *e.g.*, Image-GPT [5] trains a transformer [41] to autoregressively predict pixels' cluster centroids, which could only generate images with a maximum resolution of 64×64. [39] presents the Vector Quantized Variational Autoencoder (VQVAE), which learns images' low-dimension discrete representation and models their distribution autoregressively. VQ-VAE2 [35] extends this approach using a hierarchy of discrete representations. VQ-GAN [13] further improves the perceptual quality of reconstructed images using adversarial [14, 19] and perceptual loss [27]. ViT-VQGAN [42] introduces a more expressive transformer backbone. RQ-VAE [28] uses Residual Quantization [21, 31] to iteratively quantizes a vector and its residuals and represent the vector as a stack of tokens. Although vector quantization has become the fundamental technique for modern visual autoregressive models, the critical removing redundancy in codebook learning has not been explored yet, which becomes a critical bottleneck.

2.2. Masked Modeling

Masked modeling is popular among both natural language processing and computer vision. BERT [8] randomly masks a portion of the input sequence and trains models to predict the missing content. In the computer vision domain, the ViT [11] studies masked patch prediction for self-supervised learning. BEiT [1] proposes to predict discrete tokens. Most recently, MaskGIT [4] also used the masking strategy for VQ-based image generation. However, our proposed method differs from MaskGIT in two aspects: (1) Our primary motivation for the masking strategy applied in the proposed MQ-VAE in the first stage aims to learn a more compact and effective vector quantization (VQ) itself by masking perceptual unimportant regions, while MaskGIT uses masking strategy in the second stage to better use a learned VQ. (2) The mask in our proposed MQ-VAE is learned and adaptively changed according to different image content, while the mask in MaskGIT is randomly sampled for the mask-and-predict training. In conclusion, to the best of our knowledge, this is the first time that masked modeling has been applied for vector quantization.

3. Methodology

We propose a novel two-stage framework with MQ-VAE and Stackformer for autoregressive image generation, as illustrated in Figure 2. MQ-VAE only masks redundant region features to prevent redundant codes and Stackformer stacks two transformers to autoregressively predict the next code and its position. In the following, we will first briefly revisit the formulation of vector quantization and then describe our proposed method in detail.

3.1. Preliminary

We follow the definition and notations of previous works [13, 28]. Specifically, the *codebook* $\mathcal{C} := \{(k, \boldsymbol{e}(k))\}_{k\in[K]}$ is defined as the set of finite pairs of code k and its code embedding $\boldsymbol{e}(k) \in \mathbb{R}^{n_z}$. Here K is the codebook size and n_z is the code dimension. An image $\mathbf{X} \in \mathbb{R}^{H_0\times W_0\times 3}$ is first encoded into grid features $\mathbf{Z} = E(\mathbf{X}) \in \mathbb{R}^{H\times W\times n_z}$ by the encoder E, where $(H, W) = (H_0/f, W_0/f)$ and f is the corresponding downsampling factor. For each vector $\boldsymbol{z} \in \mathbb{R}^{n_z}$ in $\mathbf{Z}$, it is replaced with the code embedding that has the closest euclidean distance with it in the codebook $\mathcal{C}$ through the vector quantization operation $\mathcal{Q}(\cdot)$:

$$\mathcal{Q}(\boldsymbol{z};\mathcal{C}) = \arg\min_{k\in[K]} ||\boldsymbol{z} - \boldsymbol{e}_k||_2^2. \tag{1}$$

Here, $\mathcal{Q}(\boldsymbol{z};\mathcal{C})$ is the quantized code. $\boldsymbol{z^q} = \boldsymbol{e}(\mathcal{Q}(\boldsymbol{z};\mathcal{C}))$ is the quantized vector. By applying $\mathcal{Q}(\cdot)$ to each feature vector, we could get the quantized code map $\mathbf{M} \in [K]^{H\times W}$ and the quantized features $\mathbf{Z^q} \in \mathbb{R}^{H\times W\times n_z}$. The original image is reconstructed by the decoder D as $\tilde{\mathbf{X}} = D(\mathbf{Z^q})$.

3.2. Stage 1: MQ-VAE

Existing methods quantize each feature vector of $\mathbf{Z}$ without distinguishing their different perceptual importance and thus bring redundancy in the learned codebook, which not only degrades the generation quality but also decreases the generation speed. To relieve the model from this redundancy, we propose MQ-VAE with two novel modules, *i.e.*, the *adaptive mask module* for adaptively masking redundant region features before vector quantization and *adaptive de-mask module* for adaptively recovering the original grid image feature map after vector quantization.

Adaptive Mask Module. The encoded grid feature map $\mathbf{Z} \in \mathbb{R}^{H\times W\times n_e}$ is first flattened into $\mathbf{Z} \in \mathbb{R}^{L\times n_e}$, where $L = H\times W$. The proposed adaptive mask module then uses a lightweight scoring network f_s to measure the importance of each region feature $\boldsymbol{z}_l$ in $\mathbf{Z}$, which is implemented as a two-layer MLP:

$$s_l = f_s(\boldsymbol{z}_l), l = 1, ..., L. \tag{2}$$

The larger score s_l is, the more important the region feature $\boldsymbol{z}_l$ is. Then the region features are sorted in descending order according to the predicted scores. The sorted region features and their scores are denoted as $\{\boldsymbol{z}_l^{'}\}$ and $\{s_l^{'}\}$ respectively, where $l = 1, ..., L$. To enable the learning of f_s, the predicted scores are further multiplied with the normalized region features as modulating factors. We select the top N scoring vectors as the important region features,

$$\hat{\mathbf{Z}} = \{\boldsymbol{z}_l^{''} | \boldsymbol{z}_l^{''} = \text{LayerNorm}(\boldsymbol{z}_l^{'}) * s_l^{'}\}, l = 1, ..., N. \tag{3}$$

$$\hat{\mathbf{P}} = \{p_{z_l^{''}} | p_{z_l^{''}} \in \{0, ..., L\}\}, l = 1, ..., N. \tag{4}$$

Here, $\hat{\mathbf{Z}}$ denotes the selected important region features set, and $\hat{\mathbf{P}}$ denotes the corresponding position set that represents the position of each selected region feature in the original 2D feature map. The selected number $N = \alpha \times L$, where α is a constant fractional value. The mask ratio is defined as $1 - \alpha$. This design also enables a flexible trade-off between the image generation speed and image generation quality, which we will discuss in experiments. After obtaining $\hat{\mathbf{Z}}$, we further apply the quantization function $\mathcal{Q}$ to each of them and obtain the quantized important region features set $\hat{\mathbf{Z}}^{\boldsymbol{q}}$ as well as its code matrix $\hat{\mathbf{M}}$.

Adaptive De-mask Module. After quantization, we fill the quantized features $\hat{\mathbf{Z}}^{\boldsymbol{q}}$ back into the original 2D feature map according to $\hat{\mathbf{P}}$, while other masked positions are filled with a uniformly initialized learnable mask code embedding, as shown in Figure 2(a). Directly inputting filled grid features to the decoder D could bring sub-optimal reconstruction results since the mask code embedding here only serves as the placeholders that contain little information. Therefore, we further propose the adaptive de-mask module, which applies a novel *direction-constrained self-attention* to encourage the information flow from unmasked regions to the masked ones while blocking the reverse. Such a design allows the model to utilize the unmasked region features to infer the masked ones while also preventing the masked regions to have negative impacts on the unmasked ones since they are less informative.

Our adaptive de-mask module is implemented as H identical sub-modules, where each consists of a direction-constrained self-attention block and a Resnet block. The direction-constrained self-attention is mathematically formed as (Resnet block is omitted for simplicity):

$$\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v} = \boldsymbol{W}_q \hat{\mathbf{Z}}^{\boldsymbol{q},h}, \boldsymbol{W}_k \hat{\mathbf{Z}}^{\boldsymbol{q},h}, \boldsymbol{W}_v \hat{\mathbf{Z}}^{\boldsymbol{q},h} \tag{5}$$

$$\boldsymbol{A} = (\text{SoftMax}(\frac{\boldsymbol{q}\boldsymbol{k}^T}{\sqrt{n_e}})) \odot \boldsymbol{B_h} \tag{6}$$

$$\hat{\mathbf{Z}}^{\boldsymbol{q},h+1} = \boldsymbol{A}\boldsymbol{v}. \tag{7}$$

Here $h \in \{1, .., H\}$ and $\hat{\mathbf{Z}}^{\boldsymbol{q},1}$ is the filled quantized grid features. $\boldsymbol{W}_q, \boldsymbol{W}_k, \boldsymbol{W}_v \in \mathbb{R}^{n_e\times n_e}$ are the learnable parameters. $\boldsymbol{B}_h$ is the attention mask at h sub-module. Specifically, since the initial mask code contains little information, we define $\boldsymbol{B}_1 = [b_l \in \{0,1\}, |l = 1, ..., L] \in \mathbb{R}^{1\times L}$ to forbid it from attending to other unmasked codes to avoid negative impact, where 0 for the mask position and 1 for the unmasked. Considering that the mask code is updated with more and more information after each sub-module, we propose to synchronously amplify its interaction with other codes step by step through a mask updating mechanism:

$$\boldsymbol{B}_{h+1} = \sqrt{\boldsymbol{B}_h}, \tag{8}$$

where the initial 0 in $\boldsymbol{B}_1$ is replaced with a small number 0.02, since $\sqrt{0}$ will always result in 0. Finally, the original image is recovered by the decoder as $\tilde{\mathbf{X}} = D(\hat{\mathbf{Z}}^{\boldsymbol{q},H})$.

3.3. Stage 2: Stackformer

The perceptual important regions of different images vary. Therefore the positions of quantized codes in the feature map also dynamically change along with the image content. As a result, our proposed MQ-VAE formulates an image as both the code sequence $\hat{\mathbf{M}}$ and the code position sequence $\hat{\mathbf{P}}$. To simultaneously learn the combination of the codes and their positions, we propose a novel Stackformer, which stacks a Code-Transformer and a Position-Transformer. The Code-Transformer learns to predict the next code based on all previous steps' codes and their positions, while the Position-Transformer learns to predict the next code's position based on all previous steps' positions and current code. Directly treating the importance descending order sequence $\hat{\mathbf{M}}$ and $\hat{\mathbf{P}}$ as the inputs are natural, but the dramatic position changes of adjacent code could make the network hard to converge. For example, the position of the first code may be in the upper left corner of the image, while the position of the second code may be in the lower right corner of the image. Therefore, we further propose to use the raster-scan order [13] to rearrange both sequences to deal with the converge problem.

Mathematically, taking the raster-scan code and code position sequence $(\overline{\mathbf{M}}, \overline{\mathbf{P}}) = \text{rearrange}(\hat{\mathbf{M}}, \hat{\mathbf{P}})$, Stackformer learns $p(\overline{\mathbf{M}}, \overline{\mathbf{P}})$, which is autoregressively factorized as:

$$p(\overline{\mathbf{M}}, \overline{\mathbf{P}}) = \prod_{l=1}^{N} p(\overline{\mathbf{M}}_l | \overline{\mathbf{M}}_{<l}, \overline{\mathbf{P}}_{<l}) p(\overline{\mathbf{P}}_l | \overline{\mathbf{M}}_{\leq l}, \overline{\mathbf{P}}_{<l}) \tag{9}$$

Code-Transformer takes the sum of code embeddings $\boldsymbol{e}_c(\cdot)$ and code position embedding $\boldsymbol{e}_p(\cdot)$ as inputs:

$$\mathbf{U}_c = \boldsymbol{e}_c(\overline{\mathbf{M}}_{[1:N_c+N]}) + \boldsymbol{e}_{p1}(\overline{\mathbf{P}}_{[1:N_c+N]}), \tag{10}$$

where N_c is the condition length. For the unconditional generation, we add a <sos> code at the start of the code and code position sequence. For conditioning, we append class or text codes to the start of the code sequence and the same length of <sos> code to the code position sequence. We further add an extra learned absolute position embedding to $\mathbf{U}_c$ to form the final input, which makes the network aware of the absolute position of the sequence. After processing by Code-Transformer, the output hidden vector $\mathbf{H}_c$ encodes both code and their position information and is used for the next code prediction. The negative log-likelihood (NLL) loss for code autoregressive training is:

$$\mathcal{L}_{\text{code}} = \mathbb{E}[-\log p(\overline{\mathbf{M}}_l | \overline{\mathbf{M}}_{<l}, \overline{\mathbf{P}}_{<l}]). \tag{11}$$

Position-Transformer takes the sum of Code-Transformer's output hidden vector $\mathbf{H}_c$ and an extra code embedding as input:

$$\mathbf{U}_p = \mathbf{H}_c[N_c : N_c + N - 1] + \boldsymbol{e}_c(\overline{\mathbf{M}}_{[N_c+1:N_c+N]}). \tag{12}$$

Here $\mathbf{U}_p$ is the input for Position-Transformer and the information of current code is included in $\boldsymbol{e}_c(\overline{\mathbf{M}}_{[N_c+1:N_c+N]})$. The design idea behind this is that when predicting the next code's position, the model should not only be aware of previous steps' codes and their position information but also should be aware of *current* code information. The negative log-likelihood (NLL) for position autoregressive training is:

$$\mathcal{L}_{\text{position}} = \mathbb{E}[-\log p(\overline{\mathbf{P}}_l | \overline{\mathbf{M}}_{\leq l}, \overline{\mathbf{P}}_{<l})]. \tag{13}$$

Training & **Inference.** The total loss for training Stackformer is defined as:

$$\mathcal{L} = \mathcal{L}_{\text{code}} + \mathcal{L}_{\text{position}}. \tag{14}$$

The inference procedure is illustrated in Algorithm 1, where we take the unconditional generation as an example and conditional generation can be derived accordingly.

Algorithm 1 Unconditional sampling of Stackformer.

```
Input: The sample step $N$;
    The code sequence $M_{sample}$ of a single <sos>;
    The code position sequence $P_{sample}$ of a single <sos>.
Output: The generated image $\mathcal{I}$.
1: for each $n \in [1, N]$ do
2:     $H_c$ = Code-Transformer$(M_{sample}, P_{sample})$;
3:     Sample next code $M_n$ by $H_c$;
4:     $M_{sample}$ = concat$(M_{sample}, M_n)$;
5:     $H_p$ = Position-Transformer$(H_c, M_{sample}[2:])$;
6:     Mask already sampled positions to avoid conflicts;
7:     Sample next code position $P_n$ by $H_p$;
8:     $P_{sample}$ = concat$(P_{sample}, P_n)$;
9: end for
10: Re-map $M_{sample}$ to 2D code map according to $P_{sample}$
    and the rest are filled with the mask code;
11: Decode the code map to the image $\mathcal{I}$
12: return The generated image $\mathcal{I}$
```

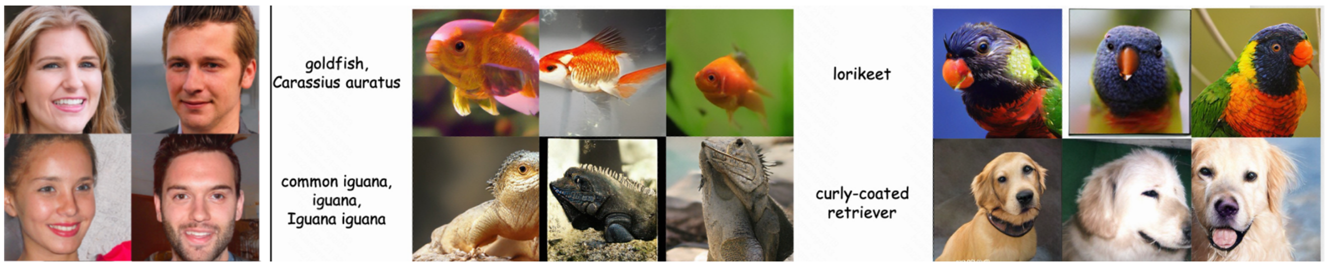
goldfish, Carassius auratus
common iguana, iguana, Iguana iguana
lorikeet
curly-coated retriever

Figure 3. Left: Our unconditional generated images on FFHQ benchmark. Right: Our class-conditional generated images on ImageNet.

4. Experiment

4.1. Experimental Settings

Benchmarks. We validate our model for unconditional, class-conditional, and text-conditional image generation tasks on FFHQ [22], ImageNet [7], and MS-COCO [30] benchmarks respectively, with 256 × 256 image resolution.

Metrics. Following previous works [13,28,42], the standard Frechet Inception Distance (FID) [16] is adopted for evaluating the generation and reconstruction quality (denoted as rFID). Inception Score (IS) [2] is also used for class-conditional generation on the ImageNet benchmark. FID and IS are calculated by sampling 50k images. rFID is calculated over the entire test set.

Implemented details. The architecture of MQ-VAE exactly follows [13] except for the proposed mask and de-mask modules, with the codebook size of $K = 1024$. For the de-mask module, the sub-module number $H = 8$. For the Stackformer, we implement two settings: a small version uses 18 transformer encoder blocks for the Code-Transformer and another 6 transformer encoder blocks for the Position-Transformer with a total of 307M parameters, and a base version uses 36 transformer encoder blocks for the Code-Transformer and another 12 transformer encoder blocks for the Position-Transformer with a total of 607M parameters to further demonstrate our scalability. The generation results are reported with a 25% mask ratio at 32×32 resolution feature map using eight RTX-3090 GPUs. Top-k and top-p sampling are used to report the best performance. More details could be found in the supplementary.

4.2. Comparison with state-of-the-art methods

Unconditional generation. We first compare with million-level state-of-the-art autoregressive models in Table 1. Our model significantly outperforms other autoregressive models with the same parameters (307M). With more parameters, we further increase the FID from 6.84 to 5.67, which demonstrates our scalability. We also compare with other types of unconditional state-of-the-art and large-scale big models in Table 3, where we also achieve top-level performance. Our qualitative unconditional generation results are shown on the left of Figure 3.

Methods	#Params	FID↓
DCT [32]	738M	13.06
VQGAN [13]	(72.1+307)M	11.4
RQ-Transformer [28]	(100+355)M	10.38
Mo-VQGAN [44]	(82.7+307)M	8.52
Stackformer	(44.4+307)M	**6.84**
Stackformer	(44.4+607)M	**5.67**

Table 1. Comparison of autoregressive unconditional generation at million-level parameters on FFHQ [22] benchmark. #Params splits in (VAE + autoregressive model).

Model Type	Methods	#Params	FID↓	IS↑
Diffusion	ADM [9]	554M	10.94	101.0
GAN	BigGAN [3]	164M	7.53	168.6
GAN	BigGAN-deep [3]	112M	6.84	203.6
Bidirection	MaskGIT [4]	227M	6.18	182.1
Autoregressive	DCT [32]	738M	36.5	n/a
Autoregressive	VQ-GAN† [13]	679M	17.03	76.85
Autoregressive	RQ-Transformer [28]	821M	13.11	104.3
Autoregressive	Mo-VQGAN [44]	383M	7.13	138.3
Autoregressive	**Stackformer**	651M	**6.04**	**172.6**

Table 2. Comparison of class-conditional image generation at million-level parameters without rejection sampling on ImageNet [7]. † denotes the model we train with the same setup with ours.

Model Type	Methods	#Params	FID↓
VAE	VDVAE [6]	115M	28.5
Diffusion	ImageBART [12]	3.5B	9.57
GAN	StyleGAN2 [23]	-	3.8
GAN	BigGAN [3]	164M	12.4
Autoregressive	ViT-VQGAN [42]	2.2B	5.3
Autoregressive	**Stackformer**	651M	**5.67**

Table 3. Comparison with other types of state-of-the-art generative models and large-scale *billion-level* parameters autoregressive models on unconditional FFHQ [22] benchmark.

Class-conditional generation. We first compare with million-level state-of-the-art in Table 2. We achieve the best FID score compared to all types of models including the re-

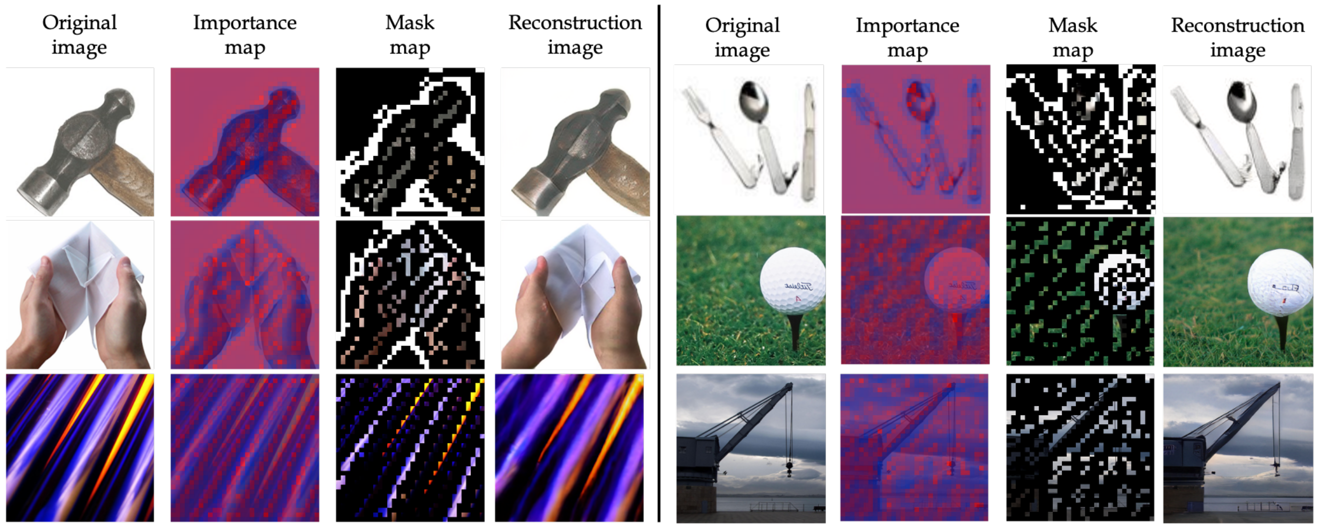

Figure 4. The visualization of our adaptive mask module which learns to mask unimportant regions on ImageNet [7]. In the importance map, red denotes high scores while blue denotes low scores.

Model Type	Methods	#Params	FID↓	IS↑
Diffusion	ImageBART [12]	3.5B	21.19	61.6
Autoregressive	VQVAE2 [35]	13.5B	31	45
Autoregressive	VQ-GAN [13]	1.4B	15.78	78.3
Autoregressive	ViT-VQGAN [42]	2.2B	4.17	175.1
Autoregressive	RQ-Transformer [28]	3.8B	7.55	134
Autoregressive	**Stackformer**	651M	**6.04**	**172.6**

Table 4. Comparison between our model and large-scale *billion-level* parameters models of class-conditional generation without rejection sampling on ImageNet [7] benchmark.

Model Type	Method	FID↓
GAN	DMGAN [45]	32.64
GAN	XMCGAN† [43]	50.08
GAN	DFGAN [38]	21.42
GAN	SSA-GAN [29]	19.37
GAN	DSE-GAN [18]	15.30
Diffusion	VQ-Diffusion [15]	19.75
Autoregressive	VQ-GAN† [13]	22.28
Autoregressive	**Stackformer**	**10.08**

Table 5. Comparison of text-conditional generation on MS-COCO [30] without using extra web-scale data or pre-trained models. † denotes reproduced results under our same experimental setting.

mask ratio	f	mask type	rFID↓	FID↓	usage↑ (%)
0%	32	-	8.1	13.5	70.02
0%	16	-	4.46	11.4	63.89
10%	16	adaptive	4.55	7.81	72.34
25%	16	adaptive	4.79	7.67	78.22
25%	16	random	6.13	12.21	67.48
50%	16	adaptive	5.31	8.36	84.29
50%	16	random	7.855	15.67	69.04
75%	16	adaptive	7.62	11.71	87.60
75%	16	random	10.58	17.62	70.41

Table 6. Ablations of adaptive mask module on FFHQ. Here f is the downsampling factor. The codebook usage is calculated as the percentage of used codes over the entire test dataset.

cent Mo-VQGAN [44] and RQ-Transformer [28]. We also compare our *million-level* model with existing *billion-level* big models in Table 4, where we also achieve top performance with fewer parameters and is only inferior to ViT-VQGAN big model. Our qualitative class-conditional generation results are shown on the right of Figure 3.

Text-conditional generation. We compare with existing text-conditional state-of-the-art without extra web-scale data or pretrained models on MS-COCO [30] for fair comparison in Table.5. We achieve 18.6% FID improvement.

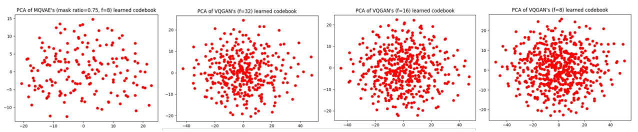

Figure 5. The PCA of learned codebook (1024 codebook size).

4.3. Ablations

We conduct ablations on 16 $\times$ 16 resolution feature map using four RTX-3090 GPUs for saving computation resources and the experimental trends are all consistent with 32 $\times$ 32 resolution feature map of the main results.

Ablations of adaptive mask module. As shown in Table 6, our proposed learned adaptive mask mechanism sig-

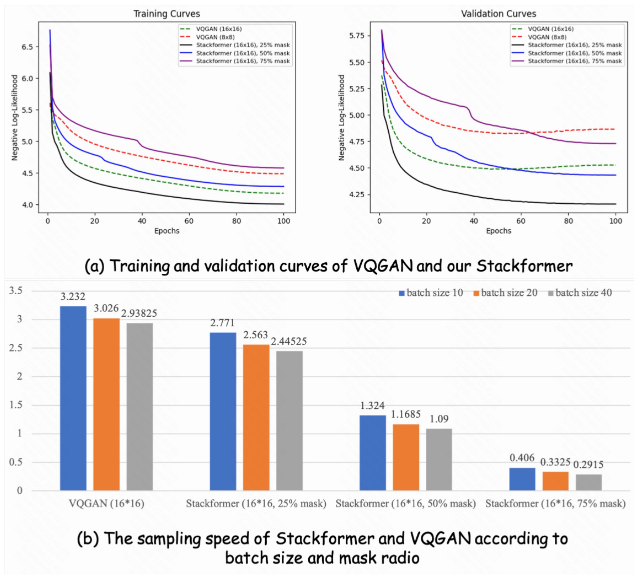

Figure 6. Comparison of training & validation curves and sample speed between VQGAN [13] and Stackformer.

nificantly outperforms the random one, which quantitatively validates that the adaptive mask module enables the model to learn perceptually important regions.

Our model with 10% and 25% mask radio has only slightly inferior reconstruction compared with VQGAN [13], but achieves a significant improvement in generation quality, which indicates that the effectiveness of focusing autoregressive models on modeling important regions. When we further increase the mask radio to 50% and 75%, the final generation quality drops, we believe the reason lies that an improper high mask radio will inevitably mask some important regions that greatly decrease the reconstruction results and hinder autoregressive modeling.

The redundancy of the existing learned codebook can be verified from two aspects: i) the PCA of the learned codebook in Figure 5, where each point is a code and closer codes have more similar semantics. We show many codes in VQGAN's codebook overlap, which indicates these codes have nearly the same semantics and are thus redundant. The redundancy increase (more overlaps) when VQGAN uses more code to represent images (smaller downsampling factor f). The redundancy is largely alleviated in our MQ-VAE. ii) in Table 6, a higher codebook usage indicates more "useful" codes in the codebook and thus less redundant. VQGAN has a lower usage compared with our MQ-VAE.

We visualize the training and validation curves of VQGAN and Stackformer in Figure 6(a). Previous autoregressive models [13, 28, 42] always suffer from the overfitting problem while our Stackformer successfully gets rid of it, which indicates the better generalization of our masked discrete representation and our model.

Model setting	rFID↓	FID↓
VQGAN	4.46	11.4
VQGAN*	4.17	11.02
MQ-VAE w/o de-mask	7.02	10.74
MQ-VAE w/ de-mask (SA)	6.56	9.8
MQ-VAE w/ de-mask (DC-SA w/o mask update)	5.84	8.92
MQ-VAE w/ de-mask (DC-SA w/ mask update)	5.31	8.36

Table 7. Ablations of adaptive de-mask module on FFHQ. SA for self-attention and DC-SA for direction-constrained self-attention. "VQGAN*" is the stronger baseline, where the same numbers of SA and Resnet blocks as MQ-VAE's de-mask module are added.

We compare the sampling speed on a single RTX-1080Ti in Figure 6(b). Compared with VQGAN, our 25% mask radio model achieves 32.72% quality improvement and 15.45% speed improvement, while Our 50% mask radio model achieves 26.67% quality improvement and 61.1% speed improvement. Therefore, our design enables a flexible trade-off between speed and quality.

Finally, We visualize the learned mask in Figure 4, with a 75% mask ratio on 32 × 32 resolution feature map, which validates that our proposed adaptive mask mechanism successfully learns to preserve the perceptual important image regions, *i.e.*, the structural and edge regions of objects.

Ablations of adaptive de-mask module. In Table 7, we show that MQ-VAE outperforms VQGAN and the stronger baseline ("VQGAN*"), which validates our effectiveness. We could conclude that the proposed direction-constrained self-attention and the mask updating mechanism both improve the reconstruction and generation quality.

5. Conclusion

In this study, we point out that the existing two-stage autoregressive generation paradigm ignores distinguishing the perceptual importance of different image regions, which brings redundancy that not only degrades generation quality but also decreases generation speed. We propose a novel two-stage generation paradigm with MQ-VAE and Stackformer to relieve the model from redundancy. MQ-VAE incorporates the *adaptive mask module* to mask redundant region features before quantization and the *adaptive de-mask module* to recover the original feature map after quantization. Stackformer then efficiently predict the combination of both codes and their positions. Comprehensive experiments on various types of image generation tasks validate the effectiveness and efficiency of our method.

6. Acknowledgments

This work is supported by National Natural Science Foundation of China under Grants 62222212 and U19A2057, Science Fund for Creative Research Groups under Grant 62121002.

References

[1] Hangbo Bao, Li Dong, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021. 3

[2] Shane Barratt and Rishi Sharma. A note on the inception score. *arXiv preprint arXiv:1801.01973*, 2018. 6

[3] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018. 6

[4] 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. 3, 6

[5] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *International Conference on Machine Learning*, pages 1691–1703. PMLR, 2020. 3

[6] Rewon Child. Very deep vaes generalize autoregressive models and can outperform them on images. *arXiv preprint arXiv:2011.10650*, 2020. 6

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 6, 7

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 3

[9] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34, 2021. 6

[10] Ming Ding, Zhuoyi Yang, Wenyi Hong, Wendi Zheng, Chang Zhou, Da Yin, Junyang Lin, Xu Zou, Zhou Shao, Hongxia Yang, et al. Cogview: Mastering text-to-image generation via transformers. *Advances in Neural Information Processing Systems*, 34, 2021. 1, 3

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 3

[12] Patrick Esser, Robin Rombach, Andreas Blattmann, and Bjorn Ommer. Imagebart: Bidirectional context with multinomial diffusion for autoregressive image synthesis. *Advances in Neural Information Processing Systems*, 34, 2021. 1, 3, 6, 7

[13] 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. 1, 2, 3, 5, 6, 7, 8

[14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. 1, 3

[15] Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. Vector quantized diffusion model for text-to-image synthesis. *arXiv preprint arXiv:2111.14822*, 2021. 7

[16] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 6

[17] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 1

[18] Mengqi Huang, Zhendong Mao, Penghui Wang, Quan Wang, and Yongdong Zhang. Dse-gan: Dynamic semantic evolution generative adversarial network for text-to-image generation. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 4345–4354, 2022. 7

[19] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017. 3

[20] Nikil Jayant, James Johnston, and Robert Safranek. Signal compression based on models of human perception. *Proceedings of the IEEE*, 81(10):1385–1422, 1993. 2

[21] Biing-Hwang Juang and A Gray. Multiple stage vector quantization for speech coding. In *ICASSP'82. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 7, pages 597–600. IEEE, 1982. 3

[22] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019. 6

[23] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020. 6

[24] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 1

[25] M Kocher and M Kunt. A contour-texture approach to picture coding. In *ICASSP'82. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 7, pages 436–439. IEEE, 1982. 2

[26] Murat Kunt, Athanassios Ikonomopoulos, and Michel Kocher. Second-generation image-coding techniques. *Proceedings of the IEEE*, 73(4):549–574, 1985. 2

[27] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017. 3

[28] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. *arXiv preprint arXiv:2203.01941*, 2022. 1, 2, 3, 4, 6, 7, 8

[29] Wentong Liao, Kai Hu, Michael Ying Yang, and Bodo Rosenhahn. Text to image generation with semantic-spatial aware gan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18187–18196, 2022. 7

[30] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 6, 7

[31] Julieta Martinez, Holger H Hoos, and James J Little. Stacked quantizers for compositional vector compression. *arXiv preprint arXiv:1411.2173*, 2014. 3

[32] Charlie Nash, Jacob Menick, Sander Dieleman, and Peter W Battaglia. Generating images with sparse representations. *arXiv preprint arXiv:2103.03841*, 2021. 3, 6

[33] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International Conference on Machine Learning*, pages 4055–4064. PMLR, 2018. 3

[34] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021. 1, 3

[35] 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. 1, 3, 7

[36] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015. 1

[37] Woncheol Shin, Gyubok Lee, Jiyoung Lee, Joonseok Lee, and Edward Choi. Translation-equivariant image quantizer for bi-directional image-text generation. *arXiv preprint arXiv:2112.00384*, 2021. 1, 3

[38] Ming Tao, Hao Tang, Songsong Wu, Nicu Sebe, Xiao-Yuan Jing, Fei Wu, and Bingkun Bao. Df-gan: Deep fusion generative adversarial networks for text-to-image synthesis. *arXiv preprint arXiv:2008.05865*, 2020. 7

[39] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017. 1, 3

[40] Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International conference on machine learning*, pages 1747–1756. PMLR, 2016. 1, 3

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 3

[42] 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. *arXiv preprint arXiv:2110.04627*, 2021. 2, 3, 6, 7, 8

[43] Han Zhang, Jing Yu Koh, Jason Baldridge, Honglak Lee, and Yinfei Yang. Cross-modal contrastive learning for text-to-image generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 833–842, 2021. 7

[44] Chuanxia Zheng, Long Tung Vuong, Jianfei Cai, and Dinh Phung. Movq: Modulating quantized vectors for high-fidelity image generation. *arXiv preprint arXiv:2209.09002*, 2022. 6, 7

[45] Minfeng Zhu, Pingbo Pan, Wei Chen, and Yi Yang. Dm-gan: Dynamic memory generative adversarial networks for text-to-image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5802–5810, 2019. 7