

# MaskGIT: Masked Generative Image Transformer

Huiwen Chang Han Zhang Lu Jiang Ce Liu\* William T. Freeman  
 Google Research

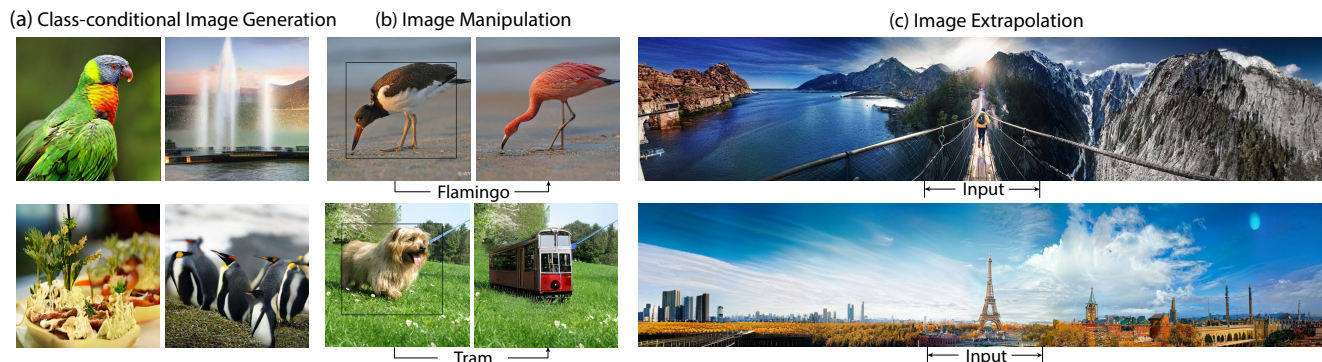


Figure 1. **Example generation by MaskGIT on image synthesis and manipulation tasks.** We show that MaskGIT is a flexible model that can generate high-quality samples on (a) class-conditional synthesis, (b) class-conditional image manipulation, *e.g.* replacing selected objects in the bounding box with ones from the given classes, and (c) image extrapolation. Examples shown here have resolutions  $512 \times 512$ ,  $512 \times 512$ , and  $512 \times 2560$  in the three columns, respectively. Zoom in to see the details.

## Abstract

Generative transformers have experienced rapid popularity growth in the computer vision community in synthesizing high-fidelity and high-resolution images. The best generative transformer models so far, however, still treat an image naively as a sequence of tokens, and decode an image sequentially following the raster scan ordering (*i.e.* line-by-line). We find this strategy neither optimal nor efficient. This paper proposes a novel image synthesis paradigm using a bidirectional transformer decoder, which we term MaskGIT. During training, MaskGIT learns to predict randomly masked tokens by attending to tokens in all directions. At inference time, the model begins with generating all tokens of an image simultaneously, and then refines the image iteratively conditioned on the previous generation. Our experiments demonstrate that MaskGIT significantly outperforms the state-of-the-art transformer model on the ImageNet dataset, and accelerates autoregressive decoding by up to 48x. Besides, we illustrate that MaskGIT can be easily extended to various image editing tasks, such as inpainting, extrapolation, and image manipulation. Project page: [masked-generative-image-transformer.github.io](https://masked-generative-image-transformer.github.io).

## 1. Introduction

Deep image synthesis as a field has seen a lot of progress in recent years. Currently holding state-of-the-art results are

Generative Adversarial Networks (GANs), which are capable of synthesizing high-fidelity images at blazing speeds. They suffer from, however, well known issues include training instability and mode collapse, which lead to a lack of sample diversity. Addressing these issues still remains open research problems.

Inspired by the success of Transformer [46] and GPT [5] in NLP, generative transformer models have received growing interests in image synthesis [7, 15, 37]. Generally, these approaches aim at modeling an image like a sequence and leveraging the existing autoregressive models to generate image. Images are generated in two stages; the first stage is to quantize an image to a sequence of discrete tokens (or visual words). In the second stage, an autoregressive model (*e.g.*, transformer) is learned to generate image tokens sequentially based on the previously generated result (*i.e.* autoregressive decoding). Unlike the subtle min-max optimization used in GANs, these models are learned by maximum likelihood estimation. Because of the design differences, existing works have demonstrated their advantages over GANs in offering stabilized training and improved distribution coverage or diversity.

Existing works on generative transformers mostly focus on the first stage, *i.e.* how to quantize images such that information loss is minimized, and share the same second stage borrowed from NLP. Consequently, even the state-of-the-art generative transformers [15, 35] still treat an image naively as a sequence, where an image is flattened into

\* Currently affiliated with Microsoft Azure AI.

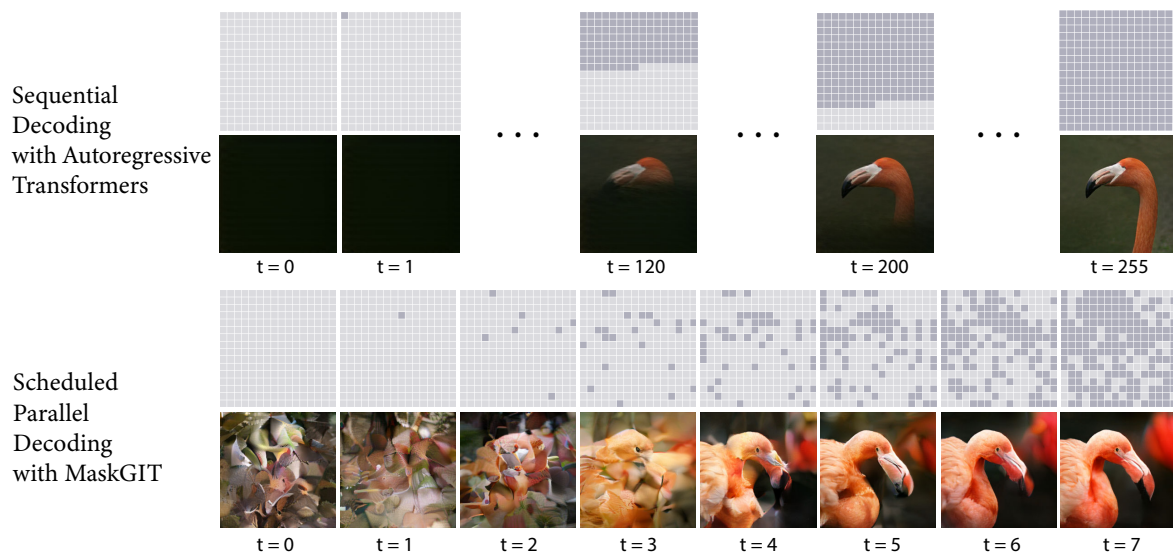


Figure 2. **Comparison between sequential decoding and MaskGIT’s scheduled parallel decoding.** Rows 1 and 3 are the input latent masks at each iteration, and rows 2 and 4 are samples generated by each model at that iteration. Our decoding starts with all unknown codes (marked in lighter gray), and gradually fills up the latent representation with more and more scattered predictions in parallel (marked in darker gray), where the number of predicted tokens increases sharply over iterations. MaskGIT finishes its decoding in 8 iterations compared to the 256 rounds the sequential method takes.

a 1D sequence of tokens following a raster scan ordering, *i.e.* from left to right line-by-line (*cf.* Figure 2). We find this representation neither optimal nor efficient for images. Unlike text, images are not sequential. Imagine how an artwork is created. A painter starts with a sketch and then progressively refines it by filling or tweaking the details, which is in clear contrast to the line-by-line printing used in previous work [7, 15]. Additionally, treating image as a flat sequence means that the autoregressive sequence length grows quadratically, easily forming an extremely long sequence—longer than any natural language sentence. This poses challenges for not only modeling long-term correlation but also renders the decoding intractable. For example, it takes a considerable 30 seconds to generate a single image on a GPU autoregressively with  $32 \times 32$  tokens.

This paper introduces a new bidirectional transformer for image synthesis called Masked Generative Image Transformer (MaskGIT). During training, MaskGIT is trained on a similar proxy task to the mask prediction in BERT [11]. At inference time, MaskGIT adopts a novel non-autoregressive decoding method to synthesize an image in constant number of steps. Specifically, at each iteration, the model predicts all tokens simultaneously in parallel but only keeps the most confident ones. The remaining tokens are masked out and will be re-predicted in the next iteration. The mask ratio is decreased until all tokens are generated with a few iterations of refinement. As illustrated in Figure 2, MaskGIT’s decoding is an order-of-magnitude faster than the autoregressive decoding as it only takes 8 steps, instead of 256 steps, to generate an image and the predictions within each step are parallelizable. Moreover, instead of conditioning only

on previous tokens in the order of raster scan, bidirectional self-attention allows the model to generate new tokens from generated tokens in all directions. We find that the mask scheduling (*i.e.* fraction of the image masked each iteration) significantly affects generation quality. We propose to use the cosine schedule and substantiate its efficacy in the ablation study.

On the ImageNet benchmark, we empirically demonstrate that MaskGIT is both significantly faster (by up to 48x) and capable of generating higher quality samples than the state-of-the-art autoregressive transformer, *i.e.* VQ-GAN, on class-conditional generation with  $256 \times 256$  and  $512 \times 512$  resolution. Even compared with the leading GAN model, *i.e.* BigGAN, and diffusion model, *i.e.* ADM [12], MaskGIT offers comparable sample quality while yielding more favourable diversity. Notably, our model establishes new state-of-the-arts on classification accuracy scores (CAS) [36] for synthesizing  $256 \times 256$  and  $512 \times 512$  images. To our knowledge, this paper provides the first evidence demonstrating the efficacy of the masked modeling for image generation on the common ImageNet benchmark.

Furthermore, MaskGIT’s multidirectional nature makes it readily extendable to image manipulation tasks that are otherwise difficult for autoregressive models. Fig. 1 shows a new application of class-conditional image editing in which MaskGIT re-generates content inside the bounding box based on the given class while keeping the context (outside of the box) unchanged. This task, which is either infeasible for autoregressive model or difficult for GAN models, is trivial for our model. Quantitatively, we demonstrate this flexibility by applying MaskGIT to image inpainting, and

image extrapolation in arbitrary directions. Even though our model is not designed for such tasks, it obtains comparable performance to the dedicated models on each task.

## 2. Related Work

### 2.1. Image Synthesis

Deep generative models [12, 17, 29, 34, 40, 43, 44, 51] have achieved lots of successes in image synthesis tasks. GAN based methods demonstrate amazing capability in yielding high-fidelity samples [4, 17, 27, 42, 51]. In contrast, likelihood-based methods, such as Variational Autoencoders (VAEs) [29, 43], Diffusion Models [12, 24, 40] and Autoregressive Models [34, 44], offer distribution coverage and hence can generate more diverse samples [40, 43, 44].

However, maximizing likelihood directly in pixel space can be challenging. So instead, VQVAE [37, 45] proposes to generate images in latent space in two stages. In the first stage, which is known as **tokenization**, it tries to compress images into discrete latent space, and primarily consists of three components:

- an encoder  $E$  that learns to tokenize images  $x \in \mathbb{R}^{H \times W \times 3}$  into latent embedding  $E(x)$ ,
- a codebook  $\mathbf{e}_k \in \mathbb{R}^D, k \in 1, 2, \dots, K$  which serves for a nearest neighbor look up used to quantize the embedding into visual tokens, and
- a decoder  $G$  which predicts the reconstructed image  $\hat{x}$  from the visual tokens  $\mathbf{e}$ .

In the second stage, it first predicts the latent priors of the visual tokens using deep autoregressive models, and then uses the decoder from the first stage to map the token sequences into image pixels. Several approaches have followed this paradigm due to the efficacy of the two-stage approach. DALL-E [35] uses Transformers [46] to improve token prediction in the second stage. VQGAN [15] adds adversarial loss and perceptual loss [26, 52] in the first stage to improve the image fidelity. A contemporary work to ours, VIM [49], proposes to use a ViT backbone [13] to further improve the tokenization stage. Since these approaches still employ an auto-regressive model, the decoding time in the second stage scales with the token sequence length.

### 2.2. Masked Modeling with Bi-directional Transformers

The transformer architecture [46], was first proposed in NLP, and has recently extended its reach to computer vision [6, 13]. Transformer consists of multiple self-attention layers, allowing interactions between all pairs of elements in the sequence to be captured. In particular, BERT [11] introduces the masked language modeling (MLM) task for language representation learning. The bi-directional self-attention used in BERT [11] allows the masked tokens in

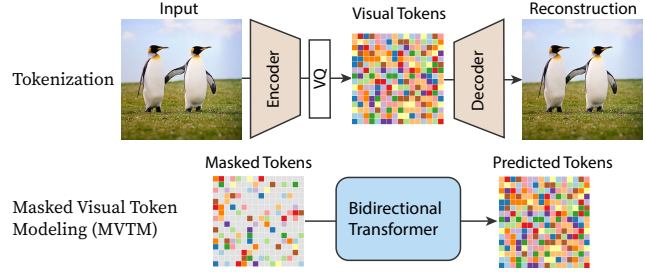


Figure 3. **Pipeline Overview.** MaskGIT follows a two-stage design, with 1) a tokenizer that tokenizes images into visual tokens, and 2) a bidirectional transformer model that performs MVTM, i.e. learns to predict visual tokens masked at random.

MLM to be predicted utilizing context from both directions. In vision, the masked modeling in BERT [11] has been extended to image representation learning [2, 21] with images quantized to discrete tokens. However, few works have successfully applied the same masked modeling to image generation [53] because of the difficulty in performing autoregressive decoding using bi-directional attentions. To our knowledge, this paper provides the first evidence demonstrating the efficacy of masked modeling for image generation on the common ImageNet benchmark. Our work is inspired by bi-directional machine translation [16, 19, 20] in NLP, and our novelty lies in the proposed new masking strategy and decoding algorithm which, as substantiated by our experiments, are essential for image generation.

## 3. Method

Our goal is to design a new image synthesis paradigm utilizing parallel decoding and bi-directional generation.

We follow the two-stage recipe discussed in 2.1, as illustrated in Figure 3. Since our goal is to improve the second stage, we employ the same setup for the first stage as in the VQGAN model [15], and leave potential improvements to the tokenization step to future work.

For the second stage, we propose to learn a bidirectional transformer by *Masked Visual Token Modeling* (MVTM). We introduce MVTM training in 3.1 and the sampling procedure in 3.2. We then discuss the key technique of masking design in 3.3.

### 3.1. MVTM in Training

Let  $\mathbf{Y} = [y_i]_{i=1}^N$  denote the latent tokens obtained by inputting the image to the VQ-encoder, where  $N$  is the length of the reshaped token matrix, and  $\mathbf{M} = [m_i]_{i=1}^N$  the corresponding binary mask. During training, we sample a subset of tokens and replace them with a special [MASK] token. The token  $y_i$  is replaced with [MASK] if  $m_i = 1$ , otherwise, when  $m_i = 0$ ,  $y_i$  will be left intact.

The sampling procedure is parameterized by a mask scheduling function  $\gamma(r) \in (0, 1]$ , and executes as follows:



we first sample a ratio from 0 to 1, then uniformly select  $\lceil \gamma(r) \cdot N \rceil$  tokens in  $\mathbf{Y}$  to place masks, where  $N$  is the length. The mask scheduling significantly affects the quality of image generation and will be discussed in 3.3.

Denote  $Y_{\overline{\mathbf{M}}}$  the result after applying mask  $\mathbf{M}$  to  $\mathbf{Y}$ . The training objective is to minimize the negative log-likelihood of the masked tokens:

$$\mathcal{L}_{\text{mask}} = - \mathbb{E}_{\mathbf{Y} \in \mathcal{D}} \left[ \sum_{\forall i \in [1, N], m_i = 1} \log p(y_i | Y_{\overline{\mathbf{M}}}) \right], \quad (1)$$

Concretely, we feed the masked  $Y_{\overline{\mathbf{M}}}$  into a multi-layer bidirectional transformer to predict the probabilities  $P(y_i | Y_{\overline{\mathbf{M}}})$  for each masked token, where the negative log-likelihood is computed as the cross-entropy between the ground-truth one-hot token and predicted token. Notice the key difference to autoregressive modeling: the conditional dependency in MVTM has two directions, which allows image generation to utilize richer contexts by attending to all tokens in the image.

### 3.2. Iterative Decoding

In autoregressive decoding, tokens are generated sequentially based on previously generated output. This process is not parallelizable and thus very slow for image because the image token length, *e.g.* 256 or 1024, is typically much larger than that of language. We introduce a novel decoding method where all tokens in the image are generated simultaneously in parallel. This is feasible due to the bi-directional self-attention of MTVM.

In theory, our model is able to infer all tokens and generate the entire image in a single pass. We find this challenging due to inconsistency with the training task. Below, the proposed iterative decoding is introduced. To generate an image at inference time, we start from a blank canvas with all the tokens masked out, *i.e.*  $Y_{\mathbf{M}}^{(0)}$ . For iteration  $t$ , our algorithm runs as follows:

1. **Predict.** Given the masked tokens  $Y_{\mathbf{M}}^{(t)}$  at the current iteration, our model predicts the probabilities, denoted as  $p^{(t)} \in \mathbb{R}^{N \times K}$ , for all the masked locations in parallel.
2. **Sample.** At each masked location  $i$ , we sample a token  $y_i^{(t)}$  based on its prediction probabilities  $p_i^{(t)} \in \mathbb{R}^K$  over all possible tokens in the codebook. After a token  $y_i^{(t)}$  is sampled, its corresponding prediction score is used as a “confidence” score indicating the model’s belief of this prediction. For the unmasked position in  $Y_{\mathbf{M}}^{(t)}$ , we simply set its confidence score to 1.0.
3. **Mask Schedule.** We compute the number of tokens to mask according to the mask scheduling function  $\gamma$  by  $n = \lceil \gamma(\frac{t}{T})N \rceil$ , where  $N$  is the input length and  $T$  is the total number of iterations.

4. **Mask.** We obtain  $Y_{\mathbf{M}}^{(t+1)}$  by masking  $n$  tokens in  $Y_{\mathbf{M}}^{(t)}$ . The mask  $\mathbf{M}^{(t+1)}$  for iteration  $t+1$  is calculated from:

$$m_i^{(t+1)} = \begin{cases} 1, & \text{if } c_i < \text{sorted}_j(c_j)[n], \\ 0, & \text{otherwise.} \end{cases}$$

where  $c_i$  is the confidence score for the  $i$ -th token.

The decoding algorithm synthesizes an image in  $T$  steps. At each iteration, the model predicts all tokens simultaneously but only keeps the most confident ones. The remaining tokens are masked out and re-predicted in the next iteration. The mask ratio is made decreasing until all tokens are generated within  $T$  iterations. In practice, the masking tokens are randomly sampled with temperature annealing to encourage more diversity, and we will discuss its effect in 4.4. Figure 2 illustrates an example of our decoding process. It generates an image in  $T = 8$  iterations, where the unmasked tokens at each iteration are highlighted in the grid, *e.g.* when  $t = 1$  we only keep 1 token and mask out the rest.

### 3.3. Masking Design

We find that the quality of image generation is significantly affected by the masking design. We model the masking procedure by a mask scheduling function  $\gamma(\cdot)$  that computes the mask ratio for the given latent tokens. As discussed, the function  $\gamma$  is used in both training and inference. During inference time, it takes the input of  $0/T, 1/T, \dots, (T-1)/T$  indicating the progress in decoding. In training, we randomly sample a ratio  $r$  in  $[0, 1)$  to simulate the various decoding scenarios.

BERT uses a fixed mask ratio of 15% [11], *i.e.*, it always masks 15% of the tokens, which is unsuitable for our task since our decoder needs to generate images from scratch. New masking scheduling is thus needed. Before discussing specific schemes, we first examine the property of the mask scheduling function. First,  $\gamma(r)$  needs to be a continuous function bounded between 0 and 1 for  $r \in [0, 1]$ . Second,  $\gamma(r)$  should be (monotonically) decreasing with respect to  $r$ , and it holds that  $\gamma(0) \rightarrow 1$  and  $\gamma(1) \rightarrow 0$ . The second property ensures the convergence of our decoding algorithm.

This paper considers common functions and makes simple transformations so that they satisfy the properties. Figure 8 visualizes these functions which are divided into three groups:

- **Linear function** is a straightforward solution, which masks an equal amount of tokens each time.
- **Concave function** captures the intuition that image generation follows a less-to-more information flow. In the beginning, most tokens are masked so the model only needs to make a few correct predictions for which the model feel confident. Towards the end, the mask ratio sharply drops, forcing the model to make a lot

more correct predictions. The effective information is increasing in this process. The concave family includes cosine, square, cubic, and exponential.

- **Convex function**, conversely, implements a more-to-less process. The model needs to finalize a vast majority of tokens within the first couple of iterations. This family includes square root and logarithmic.

We empirically compare the above mask scheduling functions in 4.4 and find the cosine function works the best in all of our experiments.

## 4. Experiments

In this section, we empirically evaluate MaskGIT on image generation in terms of quality, efficiency and flexibility. In 4.2, we evaluate MaskGIT on the standard class-conditional image generation tasks on ImageNet [10]  $256 \times 256$  and  $512 \times 512$ . In 4.3, we show MaskGIT’s versatility by demonstrating its performance on three image editing tasks, image inpainting, outpainting, and editing. In 4.4, we verify the necessity of our design of mask scheduling. We will release the code and model for reproducible research.

### 4.1. Experimental Setup

For each dataset, we only train a single autoencoder, decoder, and codebook with 1024 tokens on cropped  $256 \times 256$  images for all the experiments. The image is always compressed by a fixed factor of 16, *i.e.* from  $H \times W$  to a grid of tokens in the size of  $h \times w$ , where  $h=H/16$  and  $w=W/16$ . We find that this autoencoder, together with the codebook, can be reused to synthesize  $512 \times 512$  images.

All models in this work have the same configuration: 24 layers, 8 attention heads, 768 embedding dimensions and 3072 hidden dimensions. Our models use learnable positional embedding [46], LayerNorm [1], and truncated normal initialization (stddev=0.02). We employ the following training hyperparameters: label smoothing=0.1, dropout rate=0.1, Adam optimizer [28] with  $\beta_1=0.9$  and  $\beta_2=0.96$ . We use RandomResizeAndCrop for data augmentation and temperature=4.5 for decoding. All models are trained on 4x4 TPU devices with a batch size of 256. ImageNet models are trained for 300 epochs while the Places2 model is trained for 200 epochs.

### 4.2. Class-conditional Image Synthesis

We evaluate the performance of our model on class-conditional image synthesis on ImageNet  $256 \times 256$  and  $512 \times 512$ . Our main results are summarized in Table 1.

**Quality.** On ImageNet  $256 \times 256$ , without any special sampling strategies such as beam-search, top-k or nucleus sampling heuristics [25] or classifier guidance [37], we significantly outperform VQGAN [15] in both Fréchet Inception

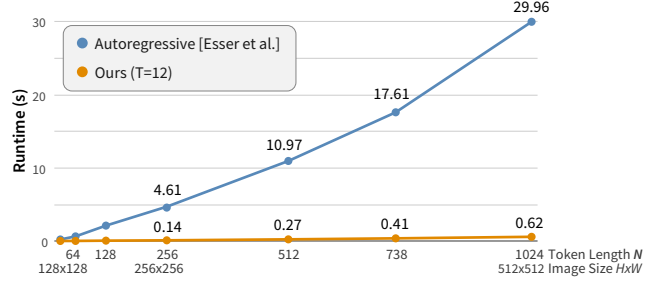


Figure 4. **Transformer wall-clock runtime comparison between VQGAN [15] and ours.** All results are run on a single GPU.

Distance (FID) [23] (6.18 vs 15.78) and Inception Score (IS) (182.1 vs 78.3). We also train a VQGAN baseline with the same tokenizer and hyperparameters as MaskGIT’s in order to further highlight the difference between bi-directional and uni-directional transformers, and find that on both resolutions, MaskGIT still outperforms our implemented baseline by a significant margin.

Furthermore, we report the results with classifier-based rejection sampling, following prior transformer-based methods [15, 37]. Specifically, we use a pre-trained ResNet classifier [22] to score output samples based on the predicted probability and keep samples with an acceptance rate of 0.05, as in VQGAN [15]. MaskGIT demonstrates consistent improvement over VQGAN, and is comparable with ADM with classifier guidance [12].

**Speed.** We evaluate model speed by assessing the number of steps, *i.e.* forward passes, each model requires to generate a sample. As shown in Table 1, MaskGIT requires the fewest steps among all non-GAN-based models on both resolutions.

To further substantiate the speed difference between MaskGIT and autoregressive models, we perform a runtime comparison between MaskGIT and VQGAN’s decoding processes. As illustrated in Figure 4, MaskGIT significantly accelerates VQGAN by 30-48x, with a speedup that gets more pronounced as the image resolution (and thus the input token length) grows.

**Diversity.** We consider Classification Accuracy Score (CAS) [36] and Precision/Recall [30] as two metrics for evaluating sample diversity, in addition to sample quality.

CAS involves first training a ResNet-50 classifier [22] solely on the samples generated by the candidate model, and then measuring the classifier’s classification accuracy on the ImageNet validation set. The last two columns in Table 1 present the CAS results, where the scores of the classifier trained on real ImageNet training data are included for reference (76.6% and 93.1% for the top-1 and top-5 accuracy). For image resolution  $256 \times 256$ , we follow the common practice of using data augmentation RandAugment [9], and report the scores trained without augmentation in the

Model	FID ↓	IS ↑	Prec ↑	Rec ↑	# params	# steps	CAS ×100 ↑	
ImageNet 256×256							Top-1 (76.6)	Top-5 (93.1)
DCTransformer [32] <sup>□</sup>	36.51	n/a	0.36	<b>0.67</b>	738M	>1024		
BigGAN-deep [4]	6.95	<b>198.2</b>	<b>0.87</b>	0.28	160M	1	43.99	67.89
Improved DDPM [33] <sup>□</sup>	12.26	n/a	0.70	0.62	280M	250		
ADM [12] <sup>□</sup>	10.94	101.0	0.69	0.63	554M	250		
VQVAE-2 [37] <sup>□</sup>	31.11	~45	0.36	0.57	13.5B <sup>†</sup>	5120	54.83	77.59
VQGAN [15] <sup>□</sup>	15.78	78.3	n/a	n/a	1.4B	256		
VQGAN*	18.65	80.4	0.78	0.26	227M	256	53.10	76.18
<b>MaskGIT</b>	<b>6.18</b>	182.1	0.80	0.51	227M	8	<b>63.14</b>	<b>84.45</b>
<hr/>								
ADM, 1.0 guidance [12] <sup>□</sup>	4.59	186.70	0.82	<b>0.52</b>				
VQGAN, 0.05 acceptance [15] <sup>□</sup>	5.88	304.8	n/a	n/a				
<b>MaskGIT, 0.05 acceptance</b>	<b>4.02</b>	<b>355.6</b>	<b>0.83</b>	0.50				
<hr/>								
ImageNet 512×512								
BigGAN-deep [4]	8.43	<b>232.5</b>	<b>0.88</b>	0.29	160M	1	44.02	68.22
ADM [12] <sup>□</sup>	23.24	58.06	0.73	<b>0.60</b>	559M	250		
VQGAN*	26.52	66.8	0.73	0.31	227M	1024	51.29	74.24
<b>MaskGIT</b>	<b>7.32</b>	156.0	0.78	0.50	227M	12	<b>63.43</b>	<b>84.79</b>
<hr/>								
ADM, 1.0 guidance [12] <sup>□</sup>	7.72	172.71	<b>0.87</b>	0.42				
<b>MaskGIT, 0.05 acceptance</b>	<b>4.46</b>	<b>342.0</b>	0.83	<b>0.44</b>				

Table 1. Quantitative comparison with state-of-the-art generative models on ImageNet 256×256 and 512×512. “# steps” refers to the number of neural network runs needed to generate a sample. \* denotes the model we train with the same architecture and setup with ours; <sup>□</sup> denotes values taken from prior publications; <sup>†</sup> estimated based on the pytorch implementation [39].



Figure 5. Sample Diversity Comparison between our proposed method MaskGIT and BigGAN-deep [4] on ImageNet 256×256. The class ids of the samples from left to right are 980, 009 and 993 respectively. Please refer to the supplementary for more comparisons.

supplementary. We find that MaskGIT significantly outperforms prior work VQVAE-2 and VQGAN, establishing a new state-of-the-art of CAS on the ImageNet benchmark on both resolutions.

The Precision/Recall results in Table 1 show that MaskGIT achieves better coverage (Recall) compared to BigGAN, and better sample quality (Precision) compared to likelihood-based models such as VQVAE-2 and diffusion

models. Compared to our baseline VQGAN, we improve the diversity as measured by recall while slightly boosting its precision.

In contrast to BigGAN’s samples, MaskGIT’s samples are more diverse with more varied lighting, poses, scales and context as shown in Figure 5. More comparisons are available in the supplementary.



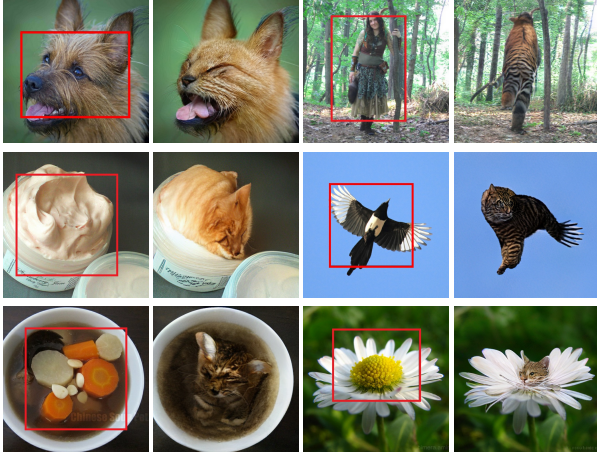
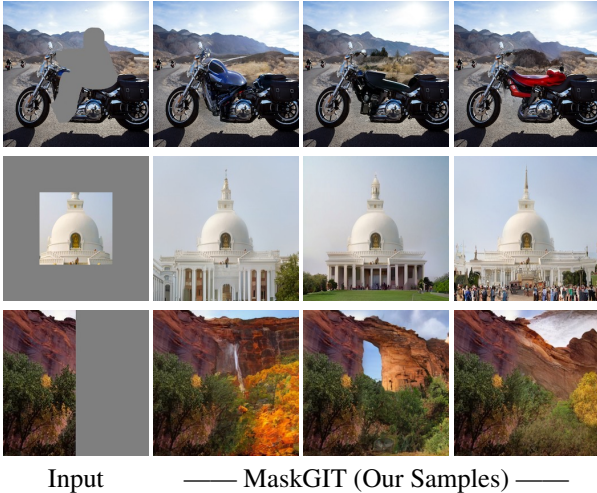


Figure 6. **Class-conditional image editing.** Given input images on the left of each pair, and a target class "tiger cat", MaskGIT replaces the bounding boxed regions with tiger cats, suggesting the composition ability of our model.



Input — MaskGIT (Our Samples) —

Figure 7. **Inpainting and outpainting.** Given a single input image, MaskGIT synthesizes diverse results for inpainting (first row) and outpainting in different directions (last two rows).

Task	Model	Resolution	FID ↓	IS ↑
<b>Outpainting</b> Right 50%	Boundless [41] <sup>o</sup>	256	35.02	6.15
	In&Out [8] <sup>o</sup>	256	23.57	7.18
	InfinityGAN [31]	256	10.60	5.57
	Boundless [41] TF *	256	7.80	5.99
	<b>MaskGIT (Ours)</b>	512	<b>6.78</b>	<b>11.69</b>
<b>Inpainting</b> Center 50% × 50%	DeepFill [50]	256	11.51	22.55
	ICT [47]	256	9.27	20.29
	HiFill [48]	512	16.60	19.93
	CoModGAN [54]	512	<b>7.13</b>	21.82
	<b>MaskGIT (Ours)</b>	512	7.92	<b>22.95</b>

Table 2. **Quantitative Comparisons for Inpainting and Outpainting on Places2.** The models are evaluated on samples with resolution consistent with their training. <sup>o</sup> taken from the prior work; \* evaluated using the TFHub model [18].

### 4.3. Image Editing Applications

In this subsection, we present direct applications of MaskGIT on three image editing tasks: class-conditional image editing, image inpainting, and outpainting. All three tasks can be almost trivially translated to ones that MaskGIT can handle if we consider the task as just a constraint on the initial binary mask  $M$  MaskGIT uses in its iterative decoding, as discussed in 3.2. We show that without modifications to the architecture or any task-specific training, MaskGIT is capable of generating very compelling results on all three applications. Furthermore, MaskGIT obtains comparable performance to dedicated models on both inpainting and outpainting, even though it is not designed specifically for either task.

**Class-conditional Image Editing.** We define a new class-conditional image editing task to showcase MaskGIT’s flexibility. In this task, the model regenerates content specified inside a bounding box on the given class while preserving the context, *i.e.* content outside of the box. It is infeasible for autoregressive methods due to the violation to their prediction orders.

For MaskGIT, however, it is a trivial task if we consider the bounding box region as the input of initial mask to the iterative decoding algorithm. Figure 6 shows a few example results. More can be found in the supplementary.

In these examples, we observe that MaskGIT can reasonably replace the selected object while preserving, or to some extent even completing, the context in the background. Furthermore, we find that MaskGIT seems to be capable of synthesizing unnatural yet plausible combinations unseen in the ImageNet training set, *e.g.* a flying cat, cat in a soup bowl, and cat in a flower. This suggests that MaskGIT has incidentally learned useful representations for composition, which may be further exploited in related tasks in future works.

**Image Inpainting.** Image inpainting is a fundamental image editing task to synthesize contents in missing regions so that the completion looks visually realistic. Traditional patch-based methods [3] work well on texture regions, while deep learning based methods [14, 38, 48, 50, 54] have been demonstrated to synthesize images requiring better semantic coherence. Both approaches have been extensively studied in computer vision.

We extend MaskGIT to this problem by tokenizing the masked image and interpreting the inpainting mask, dilated in latent space, as the initial mask in our iterative decoding. We then composite the output image by linearly blending it with the input based on the masking boundary following [8]. To match the training of our baselines, we train MaskGIT on the 512×512 center-cropped images from the Places2 [55] dataset. All hyperparameters are kept the same as the MaskGIT model trained on ImageNet.

$\gamma$	$T$	FID ↓	IS ↑	NLL
Exponential	8	7.89	156.3	4.83
Cubic	9	7.26	165.2	4.63
Square	10	6.35	179.9	4.38
<b>Cosine</b>	12	<b>6.07</b>	<b>211.6</b>	4.22
Linear	16	7.51	113.2	3.75
Square Root	32	12.33	99.0	3.34
Logarithmic	60	29.17	47.9	3.08

Table 3. **Ablation results on the mask scheduling functions.** We report the best FID, IS, and Negative Log-Likelihood loss for each candidate scheduling function.

We compare MaskGIT against common GAN-based baselines, including DeepFillv2 [50] and HiFill [48], on inpainting with a central  $50\% \times 50\%$  mask, which are evaluated on the Places2 validation set. Table 2 summarizes the quantitative comparisons. MaskGIT beats both DeepFill and HiFill in FID and IS by a significant margin, while achieving scores close to the state-of-the-art inpainting approaches such as CoModGAN [54] and ICT [47]. We show more qualitative comparisons in the supplementary.

**Image Outpainting.** Outpainting, or image extrapolation, is an image editing task that has received increased attention recently. Our adaptation of the problem and the model used in the following evaluation is the same as in inpainting.

We compare against common GAN-based baselines, including Boundless [41], In&Out [8], InfinityGAN [31], and CoModGAN [54] on extrapolating rightward with a 50% ratio. We evaluate on the image set generously provided by the authors of InfinityGAN [31] and In&Out [8].

Table 2 summarizes the quantitative comparisons. MaskGIT beats all baselines and achieves state-of-the-art FID and IS. As the examples in Figure 7 illustrate, MaskGIT is also capable of synthesizing diverse results given the same input with different seeds. We observe that MaskGIT completes objects and global structures particularly well, and hypothesize that this is thanks to the model learning useful representations with the global attentions in the transformer.

#### 4.4. Ablation Studies

We conduct ablation experiments using the default setting on ImageNet  $256 \times 256$ .

**Mask scheduling.** A key design of MaskGIT is the mask scheduling function used in both training and iterative decoding. We compare the functions discussed in 3.3, visualize them in Figure 8, and summarize the results in Table 3.

We observe that concave functions generally obtain better FID and IS than linear, followed by the convex functions. While cosine and square perform similarly relative to other functions, cosine slightly edges out square in all scores, making cosine the default in our model.

We hypothesize that concave functions perform favor-

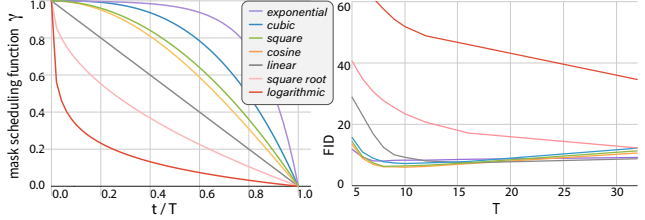


Figure 8. **Choices of Mask Scheduling Functions**  $\gamma(\frac{t}{T})$ , and **number of iterations**  $T$ . On the left, we visualize seven functions we consider for  $\gamma$ . On the right, we show line graphs of models’ FID scores against the number of decoding iterations  $T$ . Among the candidates, we find that cosine achieves the best FID.

ably because they 1) challenge training with more difficult cases (*i.e.* encouraging larger mask ratios), and 2) appropriately prioritize the less-to-more prediction throughout the decoding. That said, over-prioritization seems to be costly as well, as shown by the cubic function being worse than square, and exponential being much worse than all other concave functions.

**Iteration number.** We study the effect of the number of iterations ( $T$ ) on our model by running all candidate masking functions with different  $T$ s. As shown in Figure 8, under the same setting, more iterations are not necessarily better: as  $T$  increases, aside from the logarithmic function which performs poorly throughout, all other functions hit a “sweet spot” where the model’s performance peaks before it worsens again. The sweet spot also gets “delayed” as functions get less concave. Among functions that achieve strong FIDs (*i.e.* cosine, square, and linear), cosine not only has the strongest overall score, but also the earliest sweet spot at a total of 8 to 12 iterations. We hypothesize that such sweet spots exist because too many iterations may discourage the model from keeping less confident predictions, which worsens the token diversity. We think further study on the masking design would be interesting for future work.

## 5. Conclusion

In this paper, we propose MaskGIT, a novel image synthesis paradigm using a bidirectional transformer decoder. Trained on Masked Visual Token Modeling, MaskGIT learns to generate samples using an iterative decoding process within a constant number of iterations. Experimental results show that MaskGIT significantly outperforms the state-of-the-art transformer model on conditional image generation, and our model is readily extendable to various image manipulation tasks. As MaskGIT achieves competitive performance with GANs, applying it to other synthesis tasks is a promising direction for future work. Please see the supplementary for the limitations and more discussion.

**Acknowledgement** The authors would like to thank Xiang Kong for inspiring related works and anonymous reviewers for helpful comments.



## References

- [1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. [5](#)
- [2] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEit: BERT pre-training of image transformers. In *International Conference on Learning Representations*, 2022. [3](#)
- [3] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 28(3), Aug. 2009. [7](#)
- [4] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *ICLR*, 2019. [3](#), [6](#)
- [5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris 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. [1](#)
- [6] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021. [3](#)
- [7] 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. [1](#), [2](#)
- [8] Yen-Chi Cheng, Chieh Hubert Lin, Hsin-Ying Lee, Jian Ren, Sergey Tulyakov, and Ming-Hsuan Yang. In&out: Diverse image outpainting via gan inversion. *arXiv preprint arXiv:2104.00675*, 2021. [7](#), [8](#)
- [9] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space, 2019. [5](#)
- [10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. [5](#)
- [11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Tamar Solorio, editors, *NAACL-HLT*, 2019. [2](#), [3](#), [4](#)
- [12] Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat GANs on image synthesis. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. [2](#), [3](#), [5](#), [6](#)
- [13] 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. [3](#)
- [14] Patrick Esser, Robin Rombach, Andreas Blattmann, and Björn Ommer. Imagebart: Bidirectional context with multinomial diffusion for autoregressive image synthesis, 2021. [7](#)
- [15] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021. [1](#), [2](#), [3](#), [5](#), [6](#)
- [16] Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models, 2019. [3](#)
- [17] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014. [3](#)
- [18] Google. Tfhub model of boundless. <https://tfhub.dev/google/boundless/half/1>, 2021. [7](#)
- [19] Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. Non-autoregressive neural machine translation. In *ICLR*, 2018. [3](#)
- [20] Jiatao Gu and Xiang Kong. Fully non-autoregressive neural machine translation: Tricks of the trade, 2020. [3](#)
- [21] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners, 2021. [3](#)
- [22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. [5](#)
- [23] 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. [5](#)
- [24] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *arXiv preprint arXiv:2106.15282*, 2021. [3](#)

- [25] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration, 2019. 5
- [26] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016. 3
- [27] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *CVPR*, 2020. 3
- [28] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5
- [29] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014. 3
- [30] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. In *NeurIPS*, 2019. 5
- [31] Chieh Hubert Lin, Hsin-Ying Lee, Yen-Chi Cheng, Sergey Tulyakov, and Ming-Hsuan Yang. Infinitygan: Towards infinite-resolution image synthesis. *arXiv preprint arXiv:2104.03963*, 2021. 7, 8
- [32] Charlie Nash, Jacob Menick, Sander Dieleman, and Peter W. Battaglia. Generating images with sparse representations, 2021. 6
- [33] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. *arXiv preprint arXiv:2102.09672*, 2021. 6
- [34] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In Jennifer G. Dy and Andreas Krause, editors, *ICML*, 2018. 3
- [35] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In Marina Meila and Tong Zhang, editors, *ICML*, 2021. 1, 3
- [36] Suman V. Ravuri and Oriol Vinyals. Classification accuracy score for conditional generative models. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d’Alché Buc, Emily B. Fox, and Roman Garnett, editors, *NeurIPS*, pages 12247–12258, 2019. 2, 5
- [37] Ali Razavi, Aäron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with VQ-VAE-2. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d’Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *NeurIPS*, 2019. 1, 3, 5, 6
- [38] Chitwan Saharia, William Chan, Huiwen Chang, Chris A. Lee, Jonathan Ho, Tim Salimans, David J. Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models, 2021. 7
- [39] Kim Seonghyeon. Implementation of generating diverse high-fidelity images with vq-vae-2 in pytorch. <https://github.com/rosinality/vq-vae-2-pytorch>, 2020. 6
- [40] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d’Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *NeurIPS*, 2019. 3
- [41] Piotr Teterwak, Aaron Sarna, Dilip Krishnan, Aaron Maschinot, David Belanger, Ce Liu, and William T Freeman. Boundless: Generative adversarial networks for image extension. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10521–10530, 2019. 7, 8
- [42] Hung-Yu Tseng, Lu Jiang, Ce Liu, Ming-Hsuan Yang, and Weiling Yang. Regularizing generative adversarial networks under limited data. In *CVPR*, 2021. 3
- [43] Arash Vahdat and Jan Kautz. NVAE: A deep hierarchical variational autoencoder. In Hugo Larochelle, Marc’Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *NeurIPS*, 2020. 3
- [44] Aäron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Koray Kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional image generation with pixelcnn decoders. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *NeurIPS*, 2016. 3
- [45] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *NeurIPS*, 2017. 3
- [46] 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. 1, 3, 5
- [47] Ziyu Wan, Jingbo Zhang, Dongdong Chen, and Jing Liao. High-fidelity pluralistic image completion with transformers. *arXiv preprint arXiv:2103.14031*, 2021. 7, 8
- [48] Zili Yi, Qiang Tang, Shekoofeh Azizi, Daesik Jang, and Zhan Xu. Contextual residual aggregation for ultra high-resolution image inpainting. In *Proceedings*

of the *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7508–7517, 2020. [7](#), [8](#)

- [49] Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. Vector-quantized image modeling with improved VQGAN. *arXiv preprint arXiv:2110.04627*, 2021. [3](#)
- [50] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Free-form image inpainting with gated convolution. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4471–4480, 2019. [7](#), [8](#)
- [51] Han Zhang, Ian J. Goodfellow, Dimitris N. Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *ICML*, 2019. [3](#)
- [52] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. [3](#)
- [53] Zhu Zhang, Jianxin Ma, Chang Zhou, Rui Men, Zhikang Li, Ming Ding, Jie Tang, Jingren Zhou, and Hongxia Yang. UFC-BERT: Unifying multi-modal controls for conditional image synthesis. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. [3](#)
- [54] Shengyu Zhao, Jonathan Cui, Yilun Sheng, Yue Dong, Xiao Liang, Eric I Chang, and Yan Xu. Large scale image completion via co-modulated generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2021. [7](#), [8](#)
- [55] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. [7](#)