# RandAR: Decoder-only Autoregressive Visual Generation in Random Orders

Ziqi Pang[1*] Tianyuan Zhang[2*] Fujun Luan[3] Yunze Man[1] Hao Tan[3] Kai Zhang[3]
William T. Freeman[2] Yu-Xiong Wang[1]
[1]UIUC [2]MIT [3]Adobe Research

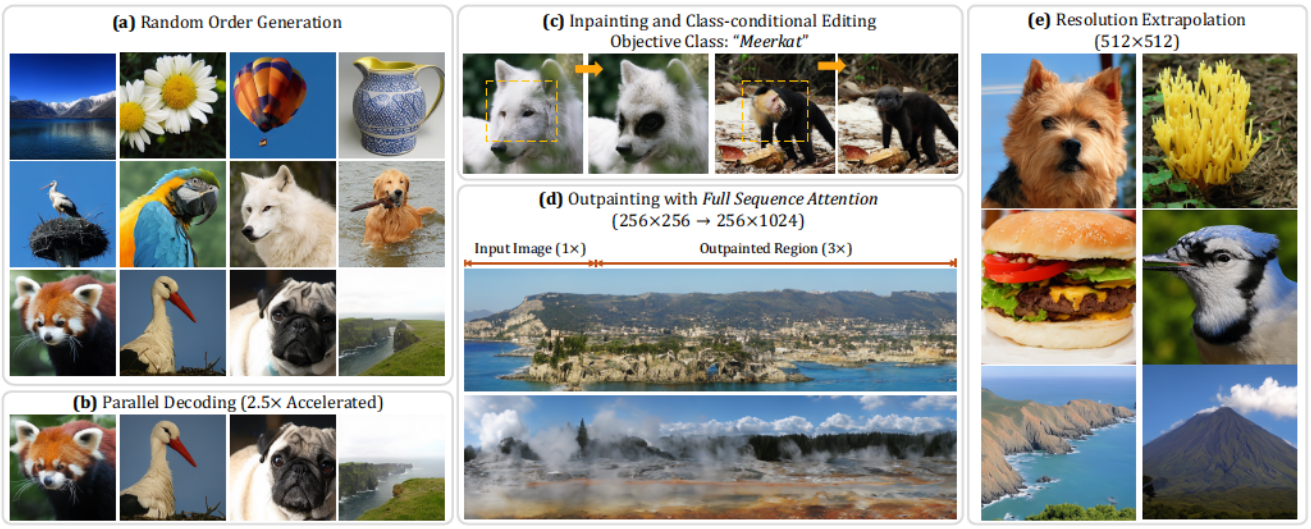



Figure 1. Our **RandAR** enables GPT-style causal decoder-only transformers to generate images via *random-order next-token prediction*, which entirely removes the raster-order sequencing inductive bias of previous decoder-only models. RandAR not only (a) generates images of comparable quality, but also shows multiple zero-shot capabilities, including (b) parallel decoding for acceleration, (c) inpainting, (d) outpainting, and (e) zero-shot generalization from a 256×256 model to synthesize high-resolution images. Zoom in for image details.

## Abstract

*We introduce RandAR, a decoder-only visual autoregressive (AR) model capable of generating images in arbitrary token orders. Unlike previous decoder-only AR models that rely on a predefined generation order, RandAR removes this inductive bias, unlocking new capabilities in decoder-only generation. Our essential design enables random order by inserting a "position instruction token" before each image token to be predicted, representing the spatial location of the next image token. Trained on randomly permuted token sequences – a more challenging task than fixed-order generation, RandAR achieves comparable performance to its conventional raster-order counterpart. More importantly, decoder-only transformers trained from random orders acquire new capabilities. For the efficiency bottleneck of AR models, RandAR adopts parallel decoding with KV-Cache at inference time, enjoying 2.5× acceleration without sacrificing generation quality. Additionally, RandAR supports inpainting, outpainting and resolution extrapolation in a zero-shot manner. We hope RandAR inspires new directions for decoder-only visual generation models and broadens their applications across diverse scenarios. Our project page is at https://rand-ar.github.io/.*

## 1. Introduction

Inspired by the success of "next-token prediction" in language modeling, computer vision researchers have explored using GPT-style *uni-directional decoder-only* transformers for image generation. The typical approach tokenizes an image into discrete 2D tokens, arranges them into 1D sequences in a row-major (raster) order from top-left to bottom-right, and applies a decoder-only transformer for sequential next visual token prediction. This design has shown promising results in uni-modal and multi-modal image generation [44, 46, 51, 58, 62]. However, enforcing a uni-directional raster order limits the decoder-only transformers from modeling the *bi-directional* context in 2D images – a constraint that their encoder-decoder counterparts, *e.g.*, MaskGIT [5] and MAR [25], do not face. Fundamental questions thus remain: Is pre-defined raster-order sequencing truly a necessary and useful inductive bias for decoder-only image generators? If not, how can we equip these models with bi-directional modeling capabilities?

*Equal Contribution.

To this end, we propose *random-order next-token prediction*, termed "**RandAR**", which enables *fully randomized generation order* during *both training and inference* time. This approach brings the encoder-decoder models' advantage of bi-directional context modeling to decoder-only models, while preserving the plain transformer architecture, simple next-token prediction mechanism, and KV-Cache acceleration.

Concretely, RandAR arranges 2D image tokens in a random-order 1D sequence, with specially designed *positional instruction tokens* inserted before each image token to indicate their spatial locations. Then, we apply a standard uni-directional decoder-only transformer for next-token prediction. Such random ordering encourages the model to learn non-local correlations. Although this setup is more challenging – introducing 256! possible sequences permutations for 256-token 256×256 images – we demonstrate that RandAR achieves comparable generation quality to its raster-order counterpart on the ImageNet benchmark [7] (examples in Fig. 1(a)).

More importantly, introducing random order to uni-directional decoder-only models unleashes their critical new *zero-shot* capabilities. As a direct advantage, RandAR inherently supports *parallel decoding* with no post-training required, accelerating sampling speed by 2.5× without compromising quality (Fig. 1(b)). Furthermore, random-order prediction provides the decoder-only model with a level of flexibility exceeding that of raster-order models, thereby unlocking new applications. Beyond inpainting [5] (Fig. 1(c)), RandAR can conduct zero-shot outpainting with a single round of *full-sequence attention* on an extrapolated number of tokens, leading to highly consistent details and patterns (Fig. 1(d)). *Surprisingly*, we demonstrate that RandAR, trained on 256×256 resolution, can synthesize 512×512 images by leveraging specially designed generation orders with full-sequence attention (Fig. 1(e)). Unlike conventional sliding-window outpainting, our high-resolution images exhibit unified objects with richer details. Finally, we show that RandAR's causal transformer can directly extract bi-directional features by processing image tokens twice, while raster-order models struggle with such generalization.

To summarize, our contributions are:

1. We introduce **RandAR**, a framework enabling causal decoder-only models to conduct random-order next-token prediction.
2. We validate our design on the ImageNet benchmark, demonstrating comparable generation quality to raster-order counterparts while reducing the inference latency by 2.5 times through parallel decoding.
3. RandAR unlocks a wide range of zero-shot capabilities: inpainting, bi-directional feature extraction, and full attention on extrapolated sequence lengths for outpainting and resolution extrapolation.

We hope RandAR removes a significant barrier to modeling 2D images with uni-directional decoder-only transformers and inspires further exploration of its broader capabilities.

## 2. Related Work

**AR Language Generation.** Current large language models (LLMs) generate text as 1D sequence autoregressively. Since the initial efforts of scaling up, two distinct architectures have emerged: bi-directional BERT-like models [19, 28, 36, 55] and unidirectional GPT-like models [3, 14, 35, 48, 49]. BERT-like architectures follow an encoder-only or encoder-decoder design and usually use mask tokens as placeholders for language generation. In comparison, GPT-like architectures are plain decoder-only transformers in causal attention, which learn to conduct "next-token prediction". Decoder-only architectures have recently become the dominant choice for language generation due to simplicity, scalability, and zero-shot generalization across various tasks. Inspired by the versatility of GPT models, we aim to build on the current decoder-only *image* model, reducing its inductive bias and expanding its zero-shot capabilities by adopting random 2D generation orders.

**Decoder-only AR Image Generation.** Models [37, 46, 51, 52, 54, 58, 62] represented by VQGAN [10], RQ-Tran [23], and LLaMAGen [44] directly transfer the GPT-style decoder-only language models for visual generation. These models turn 2D images into 1D sequences following a pre-defined factorization, typically raster order or coarse-to-fine resolutions modeled by bi-directional attention [47]. Instead, our RandAR provides a simple strategy empowering decoder-only transformers for arbitrary generation orders, which greatly extends their capabilities.

**Masked AR Image Generation.** Masked AR methods [5, 6, 11, 13, 24, 25, 27, 30, 53, 56, 59] employ bi-directional attention commonly implemented with an encoder-decoder design learning to decode place holding mask tokens. While these architectures lack KV-Cache support and direct compatibility with large language models (LLMs), *e.g.*, MaskGIT [5] and MAR [25], they offer greater flexibility and versatility than raster-order decoder-only models, such as parallel decoding and image inpainting. Therefore, the major objective of our paper is to introduce such random order and bi-directional ability into decoder-only models via our RandAR, bridging the conceptual gap between unidirectional decoder-only image generation and masked image generation.

## 3. Method

### 3.1. Preliminaries

**Decoder-only Autoregressive Models** generate sequences by predicting each token sequentially, using only past information. Formally, given a 1D sequence of $N$ variables, denoted as $\mathbf{x} = [x_1, x_2, ..., x_N]$, an autoregressive model is

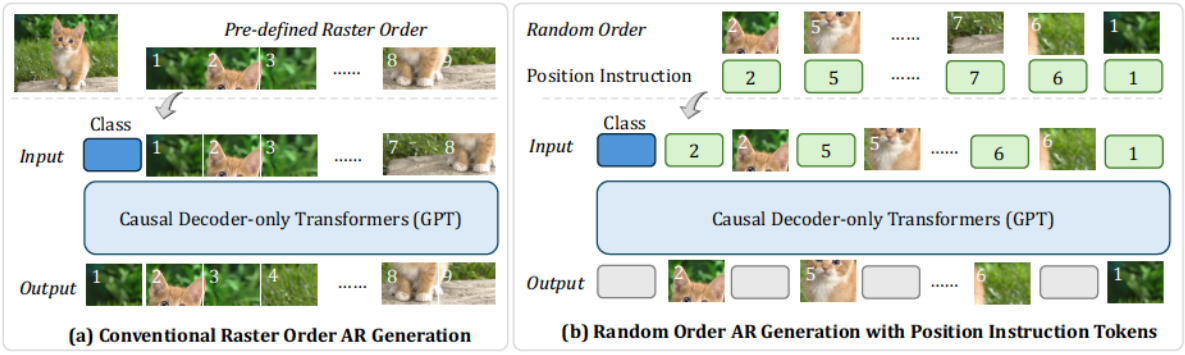



Figure 2. Overview of our **RandAR**. **(a)** Conventional autoregressive generation typically enforces a fixed order, *e.g.,* raster order, allowing the model to memorize token orders. **(b)** Our RandAR enables random order generation by inserting a *position instruction token* before each image token to be predicted. This design seamlessly integrates with the next-token prediction framework using decoder-only transformer.

trained to model the probability distribution of each variable $x_n$ based on its precedents $[x_1, .., x_{n-1}]$:

$$p_\theta(\mathbf{x}) = \prod_{n=1}^{N} p_\theta(x_n|x_1, ..., x_{n-1}), \tag{1}$$

where $p_\theta(\mathbf{x})$ may be implemented using a multinomial distribution for discrete tokens or a diffusion model for continuous tokens [25]. Currently, one of the most scalable implementations of autoregressive models employs a stack of unidirectional transformer layers with causal attention, i.e., decoder-only transformer [3].

To apply this unidirectional approach to image generation, 2D images must be converted to 1D sequences. Existing works [10, 25, 44, 50] enforce a predefined generation order, such as the raster-line order. This design introduces an inductive bias, focusing the network on predicting adjacent patches, only using context from one direction. Models trained in this way are limited to fixed generation orders and lack flexibility for tasks such as inpainting and outpainting.

In contrast, RandAR removes this inductive bias entirely by generating image token sequences in arbitrary orders. Building upon prior decoder-only visual autoregressive models [44], we introduce minimal modifications (only one additional trainable parameter) to support random-order next-token prediction. Furthermore, we show that our model achieves generation quality comparable to raster-order models under fair comparison, despite the increased complexity of learning across $(N!)$ possible orders.

### 3.2. RandAR Framework

Our goal is to introduce minimal modifications to the original GPT-style visual autoregressive framework [44] to enable random order generation. The key insight is that the model needs to be informed about the position of each next token. Our solution is straightforward: we insert a special token, called *position instruction token*, before each image token to be predicted, to represent its position. Specifically, we arrange image tokens in raster order, then randomly shuffle the sequence and drop the last:

$$[x_1^{\pi(1)}, x_2^{\pi(2)}, \ldots, x_{N-1}^{\pi(N-1)}], \tag{2}$$

where $x_i^{\pi(i)}$ is the $i$-th token in this randomly shuffled sequence of length $N$, and $\pi(i)$ denotes its original position in raster order. We then insert a positional instruction token $P_i^{\pi(i)}$ before each image token $x_i^{\pi(i)}$, as Fig. 2:

$$[P_1^{\pi(1)}, x_1^{\pi(1)}, P_2^{\pi(2)}, x_2^{\pi(2)}, \ldots, x_{N-1}^{\pi(N-1)}, P_N^{\pi(N)}]. \tag{3}$$

RandAR then applies a standard decoder-only transformer with causal attention to this sequence and supervises the prediction of each position instruction token with its subsequent image token. This random-order autoregressive modeling can be formalized as:

$$p_\theta(\mathbf{x}|\mathbf{P}) = \prod_{n=1}^{N} p_\theta(x_n^{\pi(n)}|P_1^{\pi(1)}, x_1^{\pi(1)}, \ldots, x_{n-1}^{\pi(n-1)}, P_n^{\pi(n)}). \tag{4}$$

For simplicity, we omit the subscript $\pi(i)$ in later equations. Adding position instruction tokens before each image token resembles the concept of target-aware representation, as discussed in XLNet [55].

**Position Instruction Tokens.** We insert *position instruction tokens*, representing the spatial location of each next token, to enable random-order generation, shown in Fig. 2(b). For each position, we use a shared learnable embedding $e$ "rotated" with the 2D coordinates of the next image token to be predicted, following 2D-RoPE [43]. The position instruction token for an image token at position $(h_i, w_i)$ is:

$$P_i = \text{RoPE}(e, h_i, w_i). \tag{5}$$

The ordinary RoPE [43] is a relative positional embedding only effective inside the attention operator. We empirically find that it also works well as a global position embedding in our position instruction design. Alternative designs, such as dense learnable positional embeddings or merging position instruction tokens with image tokens, are examined in the ablation study in Sec. 4.2.

**Architecture.** RandAR follows the architecture of LLaMAGen [44], using a stack of decoder-only transformers with

2D RoPE [43] as relative positional encoding within the attention module. For class-to-image generation on ImageNet [7], class IDs are embedded as learnable embeddings. Only one trainable embedding for position instruction tokens is added beyond LLaMAGen [44] to support random order next token prediction.

**Training.** We train RandAR with random sequence orders sampled from all $(N!)$ possible permutations. Using the tokenizer from LLaMAGen [44], which tokenizes a 256×256 image to $N$=16×16 2D discretized tokens, this leads to approximately $256! = 8 \times 10^{506}$ possible orders. Although training on ImageNet [7] for 300 epochs only covers a small number of $3 \times 10^8$ orders at most, RandAR learns the ability to generate images in random orders.

**Inference.** Given an arbitrary order for inference, we first compute the corresponding position instruction tokens, then iteratively sample the predicted image tokens with standard next token prediction. We discover that RandAR trained with random orders generates better images with random sequence orders at the inference time than raster orders. More analysis on inference orders is in Table B.

### 3.3. RandAR Enables Parallel Decoding

Decoder-only AR image models, by default, generate one token at a time during inference. However, this sequential decoding is bottlenecked by hardware's memory bandwidth [4, 20, 41] (also well-known as "memory wall"), as each new token generation step requires a forward pass through the model, and the model needs to load all parameters into GPU registers, which is a process considerably slower than computation. Therefore, the number of steps is a crucial factor for the latency of AR models.

Fortunately, RandAR can predict tokens at any location based on previously generated tokens. This enables parallel decoding, where RandAR simultaneously predicts tokens at multiple locations in one iteration. By reducing the number of forward steps, parallel decoding significantly decreases generation latency.

We illustrate two-token parallel decoding as an example. Suppose the generated token and position instruction token sequence up to now is $\mathbf{x}_{1:n-1} = [P_1, x_1, ..., P_{n-1}, x_{n-1}]$, at the new iteration, we append two position instruction tokens $[P_n, P_{n+1}]$ at the end of the sequence:

$$[P_1, x_1, ..., P_{n-1}, x_{n-1}, P_n, P_{n+1}], \tag{6}$$

and pass it through the network. RandAR then predicts and sample the next two tokens $[x_n, x_{n+1}]$. After sampling, we rearrange the newly added sequence to the training-time interleaved format as follows:

$$[P_1, x_1, ..., P_{n-1}, x_{n-1}, P_n, x_n, P_{n+1}, x_{n+1}]. \tag{7}$$

In subsequent iterations, we append new position instruction tokens at the sequence's end and repeat this process. Such rearrangement ensures the sequence maintains the same interleaved format used during training, with each image token preceded by a position instruction token.

Our parallel decoding requires no training modification or fine-tuning. It preserves causal masking and remains compatible with the KV cache. Such parallel decoding is already explored in masked AR methods like MaskGIT [5, 25], but lacks the support of KV-cache acceleration. We show our effective acceleration ratio with parallel decoding in Table 3.

### 3.4. Zero-shot Applications for RandAR

#### 3.4.1. Inpainting and Class-conditional Image Editing

A predefined generation order limits AR image generators in image manipulation tasks, as they cannot aggregate contextual information from different parts of the image. Consequently, decoder-only AR models, especially raster-order ones, lack zero-shot capability for tasks like inpainting and class-conditioned image editing, which are achievable with masked image modeling methods such as MaskGIT [5].

RandAR overcomes this limitation by enabling unidirectional transformers to incorporate contextual information from any part and any direction of the image. For inpainting and class-conditional image editing, we simply position visible image tokens and their corresponding position instruction tokens before the instruction tokens for the areas to be edited. RandAR then completes the remaining tokens autoregressively, as in Fig. 3(b). The capability to use spatially randomized context and support arbitrary sampling orders is necessary for these tasks and is also central to RandAR's functionality. Results are in Sec. 4.4.1.

#### 3.4.2. Outpainting

Outpainting requires extending the content of an existing image beyond its boundary in a visually coherent and contextually relevant way. Raster-order models can only extract contextual information from top-left image patches to predict the next token; thus, they have to employ strategies like sliding window as in VQGAN [10] and can only take partial contexts into account. In contrast, our RandAR can directly process all the image tokens from the conditional image, as in Fig. 5, where we outpaint the original 256×256 image to 256×1024 by extending it threefold. When outpainting beyond the training context length, we extrapolate RoPE to the target length, then use full sequence attention to model all the tokens jointly. For comparison, we also display results with sliding window attention using RandAR in the bottom row of Fig. 5 (Sec. 4.4.2).

#### 3.4.3. Resolution Extrapolation

Unlike outpainting, which extends an image's boundaries, resolution extrapolation requires generating finer details within the existing image boundaries — essentially *outpainting in the frequency domain*. We explore this task by generating 512×512 images with RandAR trained solely on 256×256 ImageNet images, without additional fine-tuning. Our resolution extrapolation involves two steps as illus-

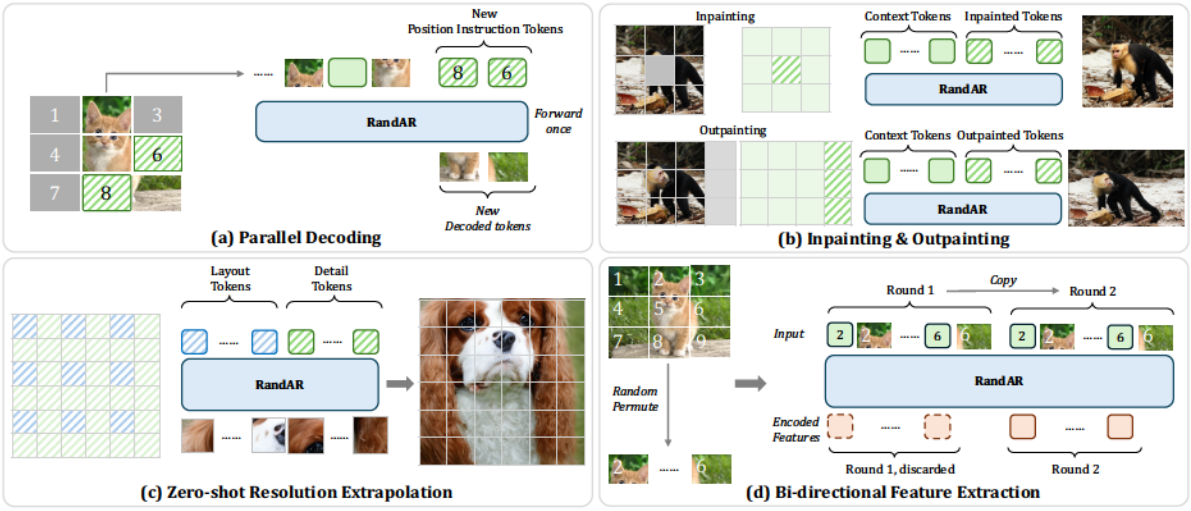



Figure 3. Zero-shot capabilities of RandAR. **(a)** RandAR directly enables parallel decoding to accelerate AR generation (Sec. 3.3). **(b)** Without the order constraint, our RandAR can support inpainting (Sec. 3.4.1) and outpainting (Sec. 3.4.2). **(c)** RandAR trained on 256×256 can also zero-shot generalize to synthesize 512×512 higher-resolution images with customized order (Sec. 3.4.3). **(d)** The random order training enables to extract features with bi-directional contexts, *i.e.*, the output from the 2nd pass of tokens (Sec. 3.4.4).

trated in Fig. 3(c): (1) Generating tokens at even coordinates to establish the overall layout, using interpolated RoPE positional embeddings; (2) Generating the tokens at the remaining coordinates. In this step, we use top-k high-frequency components in the extrapolated RoPE to replace the interpolated RoPE, which better captures the finer details, motivated by NTK-RoPE [12] for extending context lengths in language models.

This hierarchical decoding schedule relies on RandAR's ability to process tokens in random orders. For comparison, we also show generated results without this decoding approach, which exhibit visual inconsistencies. Inspired by classifier-free guidance [17], we introduce a new type of guidance: *spatial contextual guidance* (SCG) to enhance high-frequency details in resolution extrapolation. At inference, we maintain a secondary sequence where each newly sampled token is randomly dropped with a 25% probability. We then combine predicted logits from both the original and token-dropped sequences to sharpen the final output. Additional details and results are provided in Sec. G.

#### 3.4.4. Bi-directional Encoding

We find that RandAR can effectively extract features using *bi-directional context* without additional training, outperforming the representations encoded by raster-order models. Formally, an image is first tokenized and arranged into a sequence of image tokens $\mathbf{x} = [x_1, x_2, ..., x_N]$, following raster order for raster-order models and random order for random-order models. Position instruction tokens are inserted before each image token, creating an interleaved sequence, which is then passed to the model $p_\theta$ as: $p_\theta(P_1, x_1, P_2, x_2, ..., P_N, x_N)$. The output hidden embeddings corresponding to the position instruction tokens are treated as the extracted features for the image. In a unidirectional model, earlier tokens are restricted from receiving information from later parts of the image. To enable bi-directional information aggregation with a causal transformer, we pass the sequence through the model twice:

$$p_\theta(\underbrace{P_1, x_1, ..., P_N, x_N}_{\text{Round 1}}, \underbrace{P_1, x_1, ..., P_N, x_N}_{\text{Round 2}}), \quad (8)$$

and use the features from the second round as shown in Fig. 3(d). Notably, *only RandAR trained with random orders benefits from this second pass*, whereas raster-order models do not exhibit similar capabilities, as shown in Sec. 4.4.4. Moreover, RandAR leverages bi-directional context in a zero-shot manner, even though the sequence lengths and formats in Eqn. 8 differ significantly from its original training setups. For experimental results on ImageNet linear probing and zero-shot semantic image correspondence, please see Sec. 4.4.4 and Table 4.

## 4. Experiments

### 4.1. Implementation Details

We evaluate RandAR on class-conditional image generation using the ImageNet benchmark. RandAR is implemented based on LLaMAGen [44] for standardized comparison, with only one additional trainable embedding added for position instruction tokens.

**Image Tokenizer.** We use the VQGAN [10] tokenizer trained by LLaMAGen on ImageNet [7], which downsamples images by 16× and has vocabulary size of 16,384.

**Decoder-only Transformer.** RandAR follows the standard design in LLaMA [48] with RMSNorm [60] for normalization, SwiGLU [42] for activation functions, and 2D RoPE [43] for relative positional embeddings. The model architectures are shown in Table A.

**Training.** The model is trained with a batch size of

Table 1. Model comparisons on class-conditional ImageNet 256×256 benchmark. Metrics are Fréchet inception distance (FID), inception score (IS), precision and recall. "↓" or "↑" indicate lower or higher values are better. "-re" means using rejection sampling. * represents training at 384x384 resolution, and resized to 256x256 for evaluation. The raster-order counterpart is trained using the same architecture and setup as our RandAR for a fair comparison. RandAR is the only decoder-only method capable of generating images in random token orders, and it achieves comparable performance to the raster-order counterpart despite learning a more challenging task of 256! orders.

| Type | Model | #Para. | FID↓ | IS↑ | Precision↑ | Recall↑ | Steps |
|---|---|---|---|---|---|---|---|
| GAN | BigGAN [1] | 112M | 6.95 | 224.5 | 0.89 | 0.38 | 1 |
| | GigaGAN [18] | 569M | 3.45 | 225.5 | 0.84 | 0.61 | 1 |
| | StyleGan-XL [40] | 166M | 2.30 | 265.1 | 0.78 | 0.53 | 1 |
| Diffusion | ADM [8] | 554M | 4.59 | 186.70 | 0.82 | 0523 | 250 |
| | LDM-4 [38] | 400M | 3.60 | 247.7 | – | – | 250 |
| | DiT-XL [34] | 675M | 2.27 | 278.2 | 0.83 | 0.57 | 250 |
| | SiT-XL [31] | 675M | 2.06 | 270.3 | 0.82 | 0.59 | 250 |
| Bi-directional AR | MaskGIT-re [5] | 227M | 4.02 | 355.6 | – | – | 8 |
| | MAGVIT-v2 [57] | 307M | 1.78 | 319.4 | – | – | 64 |
| | MAR-L[25] | 479M | 1.98 | 290.3 | – | – | 64 |
| | MAR-H[25] | 943M | 1.55 | 303.7 | 0.81 | 0.62 | 256 |
| | TiTok-S-128 [59] | 287M | 1.97 | 281.8 | – | – | 64 |
| Casual AR | VQGAN-re [10] | 1.4B | 5.20 | 280.3 | – | – | 256 |
| | RQTran.-re [23] | 3.8B | 3.80 | 323.7 | – | – | 64 |
| | VAR [47] | 600M | 2.57 | 302.6 | 0.83 | 0.56 | 10 |
| | VAR [47] | 2.0B | 1.92 | 350.2 | 0.82 | 0.59 | 10 |
| | SAR-XL [27] | 893M | 2.76 | 273.8 | 0.84 | 0.55 | 256 |
| | RAR-B [58] | 261M | 1.95 | 290.5 | 0.82 | 0.58 | 256 |
| | RAR-L [58] | 461M | 1.70 | 299.5 | 0.81 | 0.60 | 256 |
| | RAR-XXL [58] | 955M | 1.50 | 306.9 | 0.80 | 0.62 | 256 |
| | RAR-XL [58] | 1.5B | 1.48 | 326.0 | 0.80 | 0.63 | 256 |
| | Open-MAGVIT2-XL [30] | 1.5B | 2.33 | 271.8 | 0.84 | 0.54 | 256 |
| | LlamaGen-L[44] | 343M | 3.07 | 256.06 | 0.83 | 0.52 | 256 |
| | LlamaGen-XL*[44] | 775M | 2.62 | 244.08 | 0.80 | 0.57 | 576 |
| | LlamaGen-XXL*[44] | 1.4B | 2.34 | 253.90 | 0.80 | 0.59 | 576 |
| | LlamaGen-3B*[44] | 3.1B | 2.18 | 263.33 | 0.81 | 0.58 | 576 |
| Casual AR | Raster-order Counterpart | 343M | 2.20 | 274.26 | 0.80 | 0.59 | 256 |
| | Raster-order Counterpart | 775M | 2.16 | 282.71 | 0.80 | 0.61 | 256 |
| Casual AR | RandAR-L | 343M | 2.55 | 288.82 | 0.81 | 0.58 | 88 |
| | RandAR-XL | 775M | 2.25 | 317.77 | 0.80 | 0.60 | 88 |
| | RandAR-XL | 775M | 2.22 | 314.21 | 0.80 | 0.60 | 256 |
| | RandAR-XXL | 1.4B | 2.15 | 321.97 | 0.79 | 0.62 | 88 |

1024 for 300 epochs (360K iterations) without exponential moving average. We use AdamW [21, 29] optimizer with $(\beta_1, \beta_2)$ as $(0.9, 0.95)$ and the weight decay of 0.05. The learning rate remains constant as $4 \times 10^{-4}$ for the first 250 epochs without warmup, then linearly decays to $1 \times 10^{-5}$ in the last 50 epochs. A standard token dropout of 0.1 is applied. To support Classifier-free guidance [17], class embedding is randomly dropped with a 10% probability.

**Inference.** The RandAR trained with random orders generates images following fully randomized orders. By default, we use 88 steps to generate 256 tokens for a 256x256 resolution image. Following MaskGIT [5] and MAR [25], we apply a cosine schedule for the parallel decoding step size and a linear schedule for classifier-free guidance [17].

## 4.2. Main Results

We use Fréchet Inception Distance (FID) [16] as our primary metric, sampling 50,000 images with a fixed random seed and evaluating FID using code from ADM [8]. Following LLamaGen [44], we also report Inception Score (IS) [39], Precision and Recall [22].

Table 2. The design choices for *position instruction tokens*. Our default design uses a single shared learnable embedding with 2D RoPE [43] for all image locations. We compare this with: (1) *Dense* learnable embeddings (256 unique embeddings for 16×16 tokens in 256×256 resolution images); (2) *Merge* position instruction tokens to its precedent image tokens by adding them together, reducing sequence length by half. However, this "*Merge*" design reduces performance significantly when parallel decoding is applied, so we report its results without parallel decoding.

| | FID↓ | Inception Score ↑ | #Steps |
|---|---|---|---|
| RandAR | 2.82 | 293.6 | 88 |
| Dense | 3.07 | 290.6 | 88 |
| Dense & Merge | 3.37 | 307.6 | 256 |

For a fair comparison, we create a raster-order counterpart using the identical setup. For all the experiments, we sweep the optimal weight for classifier-free guidance. We compare our results with current state-of-the-art methods and the raster-order counterparts in Table 1. Results

show that the XL-sized random order model reaches comparable performance with the raster counterpart, despite the increased difficulty of random-order generation.

We also plot FID over training iterations for different model sizes in Fig. 4(a), showing consistent improvements with larger models and longer training.

**Ablation Study: Design Choices of Position Instruction Tokens.** To enable random-order generation, we insert a *position instruction token* before each image token to be predicted. Our default design uses a single learnable embedding with 2D RoPE [43] to represent all image locations, adding only one additional learnable embedding to the existing decoder-only visual AR model. We explore two additional design choices. *Dense*: A unique learnable embedding is used for each spatial location, resulting in 256 position instruction tokens for a 256×256 resolution. *Merge*: The position instruction token is added with the image token before it, *i.e.*, each image token directly incorporates the position of the next token to be predicted.

We train an XL-sized model with 775M parameters for each design choice over 100k iterations, following the setup in Sec. 4.1. We report FID-50K and Inception Score for each in Table 2. Our default design shows the best performance. Notably, the *Merge* design shows degraded performance when using parallel decoding directly.

### 4.3. Effects of Parallel Decoding

We apply parallel decoding to reduce inference steps and generation latency. To assess its impact on quality, we apply parallel decoding to both RandAR and a raster-order model trained under the same setup, reporting FID-50K. As shown in Fig. 4(b), the raster-order model experiences a significant performance drop, while RandAR maintains consistent quality up to 88 inference steps. This zero-shot ability comes from random order training.

To measure its improvement in generation speed, we assess latency with varied inference steps using PyTorch on an A100 GPU (40G VRAM). Specifically, we generate a batch of 64 images (equivalent to a batch size of 128 with Classifier-free guidance) at a 256x256 resolution. We also measure the latency of LLaMAGen [44] and MAR [25] in the same way. For a fair comparison, the time spent on decoding latent image tokens to pixel space is omitted. As in Table 3, RandAR using an 88-step schedule requires 2.9 × less steps and consequently lowers the latency by 2.5 ×.

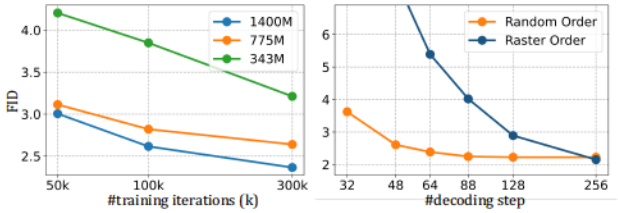
### 4.4. Case Studies: Random v.s. Raster Order

#### 4.4.1. Inpainting and Class-conditioned Editing

As described in Sec. 3.4.1, RandAR can autoregressively fill blank patches in an image using all the visible tokens as context. Previously, only methods using bi-directional attention [5, 47] demonstrated this capability. In Fig. 5, we show zero-shot inpainting results from RandAR.

Table 3. Latency of generating 256×256 images. We test the latency on of our model with different decoding steps with Pytorch on A100 GPU (40G VRAM) and a batch size of 64 (128 with classifier-free guidance). Reducing inference steps with parallel decoding greatly lowers latency.

| Method | Latency (sec.) | #Params | #Steps | Parallel Decoding | KV-Cache |
|---|---|---|---|---|---|
| RandAR | 16.8 | 1.4B | 256 | Support | Support |
| RandAR | 6.6 | 1.4B | 88 | Support | Support |
| RandAR | 4.6 | 1.4B | 48 | Support | Support |
| LLamaGen [44] | 15.9 | 1.4B | 256 | Noncompatible | Support |
| MAR [25] | 53.3 | 943M | 64 | Support | Noncompatible |
| MAR [25] | 220.0 | 943M | 256 | Support | Noncompatible |




(a) Effect of Scaling (b) Effect of Parallel Decoding

Figure 4. (a) FID-50K over training iterations for RandAR in three different sizes. (b) Effect of inference steps on FID-50K for RandAR and the raster-order counterpart (775M models).

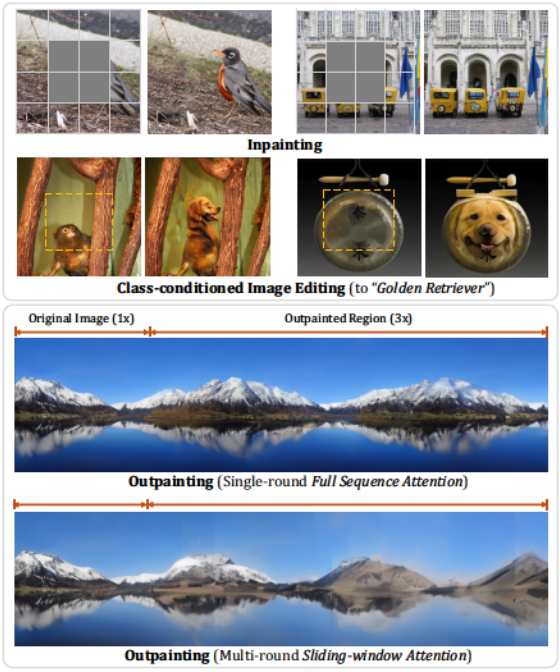



Figure 5. RandAR supports image manipulation tasks of inpainting and class-conditional image editing with a uni-directional transformer. Moreover, RandAR can use full casual attention for outpainting a 256×256 image into a consistent 256×1024.

#### 4.4.2. Outpainting

As described in Sec. 3.4.2, RandAR can extend an image beyond its boundaries directly using full causal sequence attention. In contrast, previous decoder-only visual AR models [10] are limited to using partial context, relying only on image tokens to the left or above the target token, and


Figure 6. RandAR, trained on 256×256 images, can generate 512×512 images with finer details in zero-shot (Sec 3.4.3). This is achieved using a hierarchical decoding order, benefiting the unified layout, and SCG (Spatial Context Guidance) enhancing the visual quality by refining high-frequency details.

typically employ an iterative sliding window attention approach. In Fig. 5, we demonstrate outpainting by extending a 256×256 image to 256×1024 in the rightward direction. RandAR leverages *full casual attention* for both source image context and newly generated tokens to maintain consistent patterns. By contrast, results from sliding window attention show a noticeable degradation in quality. More visualization examples are displayed in Sec. H.1.

#### 4.4.3. Resolution Extrapolation

Fig. 6 shows RandAR's zero-shot resolution extrapolation capability. Trained on 256×256 images, RandAR can generate 512×512 images with finer details. Unlike iterative outpainting [5, 9, 61], our approach does not aim to expand an image's content but to enhance its detail with higher output resolution. The resolution extrapolation process uses a hierarchical order described in Sec.3.4.3. Images generated without this order, shown at the bottom of Fig.6, exhibit inconsistent patterns. The spatial contextual guidance (SCG) further enhances consistency in details, *e.g.*, the two eyes of the chameleon. Please check Sec. H.3 for more results.

Zero-shot generalization to high-resolution images remains challenging, particularly in generating high-frequency patterns not prevalent in low-resolution training data. For instance, the space shuttle in Fig. 6 struggles with intricate structures and boundaries. Addressing these challenges and supporting varied extrapolation ratios are areas for future exploration.

#### 4.4.4. Feature Encoding

We compare the representations learned by random-order and raster-order generation models, both XL-sized models trained on ImageNet under the same setup. More detailed experiments are in Sec. F.

Table 4. Random order training enables a *uni-directional* decoder-only transformer to extract *bi-directional* context by passing image token sequence twice through the model (2nd round), while raster-order models fail to extract bi-directional contextual features.

| Model | Feature Correspondence (SPair71k) | | Linear Probing (ImageNet) | |
|---|---|---|---|---|
| | PCK (Per Image) ↑ | PCK (Per Point) ↑ | Top-1 Acc ↑ | Top-5 Acc ↑ |
| RasterAR | 24.5 | 28.6 | 62.6 | 83.9 |
| w/ 2nd Round | 3.6 | 3.9 | 58.3 | 80.7 |
| RandAR | 22.1 | 25.8 | 57.3 | 80.3 |
| w/ 2nd Round | **31.3** | **36.4** | **63.1** | **84.2** |

**Local Representation.** We evaluate local representation using zero-shot semantic correspondence on the SPair71k benchmark [32]. SPair71k contains nearly 71k paired images with sparse semantic-level correspondence annotated per pair. Following DIFT [45], We extract features for each image following Sec. 3.4.4, and detect correspondence using dot product between tokens from paired images. We compute the "percentage of correct key points" (PCK), averaging the metrics per image or point.

As shown in Table 4, by extracting features from the second round of tokens, RandAR effectively improves the correspondences. On the contrary, the raster order model experiences a significant drop and struggles to understand longer sequence lengths than training time. Although the random order model under-performs the raster one in the first round, possibly due to learning from a more challenging combination of orders, its features are finally better than a raster-order model with sufficient context. This shows that RandAR can directly generalize to bi-directional contexts without additional training.

**Global Representation.** We average-pool the embeddings and perform linear probing, following MAE [15] on ImageNet [7]. As shown in Table 4, the global representation follows a similar trend as the local features: RandAR successfully generalizes to the bi-directional context in the second round of tokens, while the raster-order model fails to leverage additional context. This experiment further supports the idea that random-order unidirectional transformers can learn to model bi-directional contextual information.

## 5. Conclusions

We introduce RandAR, a GPT-style causal decoder-only transformer that generates image tokens autoregressively in random orders. Our RandAR achieves this with specially designed position instruction tokens representing the location of next-token prediction. Despite the challenges of learning random order generation, RandAR achieves comparable performance with raster-order counterparts. Moreover, RandAR shows several new zero-shot applications for decoder-only models, including parallel decoding for 2.5× acceleration, inpainting, outpainting, resolution extrapolation, and feature extraction with bi-directional contexts. We hope RandAR inspires further exploration of unidirectional decoder-only models for visual tasks.

**Acknowledgments.** We thank Tianhong Li for the fruitful discussion. This work was supported in part by NSF Grant 2106825, NIFA Award 2020-67021-32799, the Department of the Air Force Artificial Intelligence Accelerator under Cooperative Agreement Number FA8750-19-2-1000, NSF PHY-2019786 (the NSF AI Institute for Artificial Intelligence and Fundamental Interactions, http://iaifi.org/), NSF CIF 1955864 (Occlusion and Directional Resolution in Computational Imaging). This work used computational resources, including the NCSA Delta and DeltaAI supercomputers through allocations CIS230012 and CIS240387 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, as well as the TACC Frontera supercomputer and Amazon Web Services (AWS) through the National Artificial Intelligence Research Resource (NAIRR) Pilot. We also acknowledge support from Quanta Computer.

## References


[1] Andrew Brock. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018. 6

[2] Tim Brooks, Aleksander Holynski, and Alexei A Efros. InstructPix2Pix: Learning to follow image editing instructions. In *CVPR*, 2023. 3

[3] Tom B Brown. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020. 2, 3

[4] Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. In *ICML*, 2024. 4

[5] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *CVPR*, 2022. 1, 2, 4, 6, 7, 8

[6] Huiwen Chang, Han Zhang, Jarred Barber, AJ Maschinot, José Lezama, Lu Jiang, Ming-Hsuan Yang, Kevin Murphy, William T Freeman, Michael Rubinstein, et al. Muse: Text-to-image generation via masked generative transformers. In *ICML*, 2023. 2

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 2, 4, 5, 8

[8] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In *NeurIPS*, 2021. 6

[9] Zheng Ding, Mengqi Zhang, Jiajun Wu, and Zhuowen Tu. Patched denoising diffusion models for high-resolution image synthesis. In *ICLR*, 2023. 8

[10] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021. 2, 3, 4, 5, 6, 7, 1

[11] Lijie Fan, Tianhong Li, Siyang Qin, Yuanzhen Li, Chen Sun, Michael Rubinstein, Deqing Sun, Kaiming He, and Yonglong Tian. Fluid: Scaling autoregressive text-to-image generative models with continuous tokens. *arXiv preprint arXiv:2410.13863*, 2024. 2

[12] Yao Fu, Rameswar Panda, Xinyao Niu, Xiang Yue, Hannaneh Hajishirzi, Yoon Kim, and Hao Peng. Data engineering for scaling language models to 128k context. In *ICML*, 2024. 5

[13] Shanghua Gao, Pan Zhou, Ming-Ming Cheng, and Shuicheng Yan. Masked diffusion transformer is a strong image synthesizer. In *ICCV*, 2023. 2

[14] Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, et al. Olmo: Accelerating the science of language models. *arXiv preprint arXiv:2402.00838*, 2024. 2

[15] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022. 8

[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. In *NeurIPS*, 2017. 6

[17] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 5, 6

[18] Minguk Kang, Jun-Yan Zhu, Richard Zhang, Jaesik Park, Eli Shechtman, Sylvain Paris, and Taesung Park. Scaling up gans for text-to-image synthesis. In *CVPR*, 2023. 6

[19] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019. 2

[20] Sehoon Kim, Coleman Richard Charles Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W Mahoney, and Kurt Keutzer. SqueezeLLM: Dense-and-sparse quantization. In *ICML*, 2024. 4

[21] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[22] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. In *NeurIPS*, 2019. 6

[23] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *CVPR*, 2022. 2, 6

[24] Tianhong Li, Huiwen Chang, Shlok Mishra, Han Zhang, Dina Katabi, and Dilip Krishnan. Mage: Masked generative encoder to unify representation learning and image synthesis. In *CVPR*, 2023. 2

[25] Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. In *NeurIPS*, 2024. 1, 2, 3, 4, 6, 7

[26] Nan Liu, Shuang Li, Yilun Du, Antonio Torralba, and Joshua B Tenenbaum. Compositional visual generation with composable diffusion models. In *ECCV*, 2022. 3

[27] Wenze Liu, Le Zhuo, Yi Xin, Sheng Xia, Peng Gao, and Xiangyu Yue. Customize your visual autoregressive recipe with set autoregressive modeling. *arXiv preprint arXiv:2410.10511*, 2024. 2, 6

[28] Yinhan Liu. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 364, 2019. 2

[29] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 6

[30] Zhuoyan Luo, Fengyuan Shi, Yixiao Ge, Yujiu Yang, Limin Wang, and Ying Shan. Open-magvit2: An open-source project toward democratizing auto-regressive visual generation. *arXiv preprint arXiv:2409.04410*, 2024. 2, 6

[31] Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and Saining Xie. SIT: Exploring flow and diffusion-based generative models with scalable interpolant transformers. *arXiv preprint arXiv:2401.08740*, 2024. 6

[32] Juhong Min, Jongmin Lee, Jean Ponce, and Minsu Cho. Spair-71k: A large-scale benchmark for semantic correspondence. *arXiv preprint arXiv:1908.10543*, 2019. 8, 2, 3

[33] Ziqi Pang, Ziyang Xie, Yunze Man, and Yu-Xiong Wang. Frozen transformers in language models are effective visual encoder layers. In *ICLR*, 2024. 2

[34] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023. 6

[35] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding with unsupervised learning. *Technical report, OpenAI*, 2018. 2

[36] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020. 2

[37] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *ICML*, 2021. 2

[38] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 6

[39] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NeurIPS*, 2016. 6

[40] Axel Sauer, Katja Schwarz, and Andreas Geiger. Stylegan-xl: Scaling stylegan to large diverse datasets. In *SIGGRAPH*, 2022. 6

[41] Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019. 4

[42] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020. 5

[43] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. 3, 4, 5, 6, 7

[44] Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. Autoregressive model beats diffusion: Llama for scalable image generation. *arXiv preprint arXiv:2406.06525*, 2024. 1, 2, 3, 4, 5, 6, 7

[45] Luming Tang, Menglin Jia, Qianqian Wang, Cheng Perng Phoo, and Bharath Hariharan. Emergent correspondence from image diffusion. In *NeurIPS*, 2023. 8, 2

[46] Chameleon Team. Chameleon: Mixed-modal early-fusion foundation models. *arXiv preprint arXiv:2405.09818*, 2024. 1, 2, 4

[47] Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. In *NeurIPS*, 2024. 2, 6, 7

[48] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 2, 5

[49] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. 2

[50] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, and Alex Graves. Conditional image generation with pixelcnn decoders. In *NeurIPS*, 2016. 3

[51] Xinlong Wang, Xiaosong Zhang, Zhengxiong Luo, Quan Sun, Yufeng Cui, Jinsheng Wang, Fan Zhang, Yueze Wang, Zhen Li, Qiying Yu, Yingli Zhao, Yulong Ao, Xuebin Min, Tao Li, Boya Wu, Bo Zhao, Bowen Zhang, Liangdong Wang, Guang Liu, Zheqi He, Xi Yang, Jingjing Liu, Yonghua Lin, Tiejun Huang, and Zhongyuan Wang. Emu3: Next-token prediction is all you need. *arXiv preprint arXiv:2409.18869*, 2024. 1, 2, 4

[52] Yuqing Wang, Shuhuai Ren, Zhijie Lin, Yujin Han, Haoyuan Guo, Zhenheng Yang, Difan Zou, Jiashi Feng, and Xihui Liu. Parallelized autoregressive visual generation. *arXiv preprint arXiv:2412.15119*, 2024. 2

[53] Mark Weber, Lijun Yu, Qihang Yu, Xueqing Deng, Xiaohui Shen, Daniel Cremers, and Liang-Chieh Chen. Maskbit: Embedding-free image generation via bit tokens. *arXiv preprint arXiv:2409.16211*, 2024. 2

[54] Jinheng Xie, Weijia Mao, Zechen Bai, David Junhao Zhang, Weihao Wang, Kevin Qinghong Lin, Yuchao Gu, Zhijie Chen, Zhenheng Yang, and Mike Zheng Shou. Show-o: One single transformer to unify multimodal understanding and generation. *arXiv preprint arXiv:2408.12528*, 2024. 2

[55] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, 2019. 2, 3

[56] Lijun Yu, Yong Cheng, Kihyuk Sohn, José Lezama, Han Zhang, Huiwen Chang, Alexander G. Hauptmann, Ming-Hsuan Yang, Yuan Hao, Irfan Essa, and Lu Jiang. Magvit: Masked generative video transformer. In *CVPR*, 2023. 2

[57] Lijun Yu, José Lezama, Nitesh B. Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Vighnesh Birodkar, Agrim Gupta, Xiuye Gu, Alexander G. Hauptmann, Boqing Gong, Ming-Hsuan Yang, Irfan Essa, David A. Ross, and Lu Jiang. Language model beats diffusion–tokenizer is key to visual generation. In *ICLR*, 2024. 6

[58] Qihang Yu, Ju He, Xueqing Deng, Xiaohui Shen, and Liang-Chieh Chen. Randomized autoregressive visual generation. *arXiv preprint arXiv:2411.00776*, 2024. 1, 2, 6

[59] Qihang Yu, Mark Weber, Xueqing Deng, Xiaohui Shen, Daniel Cremers, and Liang-Chieh Chen. An image is worth 32 tokens for reconstruction and generation. *arXiv preprint arXiv:2406.07550*, 2024. 2, 6

[60] Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *NeurIPS*, 2019. 5

[61] Qinsheng Zhang, Jiaming Song, Xun Huang, Yongxin Chen, and Ming-Yu Liu. Diffcollage: Parallel generation of large content with diffusion models. In *CVPR*, 2023. 8

[62] Chunting Zhou, Lili Yu, Arun Babu, Kushal Tirumala, Michihiro Yasunaga, Leonid Shamis, Jacob Kahn, Xuezhe Ma, Luke Zettlemoyer, and Omer Levy. Transfusion: Predict the next token and diffuse images with one multi-modal model. *arXiv preprint arXiv:2408.11039*, 2024. 1, 2, 4

[63] Xueyan Zou, Jianwei Yang, Hao Zhang, Feng Li, Linjie Li, Jianfeng Wang, Lijuan Wang, Jianfeng Gao, and Yong Jae Lee. Segment everything everywhere all at once. In *NeurIPS*, 2024. 2