FastVAR: Linear Visual Autoregressive Modeling via Cached Token Pruning

Hang Guo[1] Yawei Li[2,*,†] Taolin Zhang[1] Jiangshan Wang[1]
Tao Dai[3,*] Shu-Tao Xia[1,4] Luca Benini[2]
[1]Tsinghua University [2]ETH Zürich [3]Shenzhen University [4]Peng Cheng Laboratory

Figure 1. Our FastVAR can generate one 2K image using one NVIDIA 3090 GPU, while existing baseline fails due to out of memory.

Abstract

*Visual Autoregressive (VAR) modeling has gained popularity for its shift towards next-scale prediction. However, existing VAR paradigms process the entire token map at each scale step, leading to the complexity and runtime scaling dramatically with image resolution. To address this challenge, we propose FastVAR, a post-training acceleration method for efficient resolution scaling with VARs. Our key finding is that the majority of latency arises from the large-scale step where most tokens have already converged. Leveraging this observation, we develop the cached token pruning strategy that only forwards pivotal tokens for scale-specific modeling while using cached tokens from previous scale steps to restore the pruned slots. This significantly reduces the number of forwarded tokens and improves the efficiency at larger resolutions. Experiments show the proposed FastVAR can further speedup FlashAttention-accelerated VAR by **2.7**× with negligible performance drop of **<1%**. We further extend FastVAR to zero-shot generation of higher resolution images. In particular, FastVAR can generate one **2K** image with **15GB** memory footprints in **1.5s** on a single NVIDIA 3090 GPU. Code is available at* https://github.com/csguoh/FastVAR.

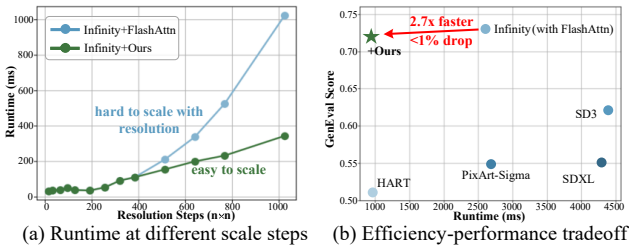
1. Introduction

The next-token prediction of Autoregressive (AR) models [31, 33, 48, 56, 61] has demonstrated performance competitive with diffusion models for visual generation [48] as well as the potential for unified vision understanding and generation [56, 57, 59, 61]. However, this token-by-token paradigm suffers from numerous decoding steps. Recently, Visual Autoregressive (VAR) modeling [51] has shifted the paradigm to next-scale prediction, enabling image generation in fewer steps. Under this new paradigm, some works [22, 49] have developed VAR-based models for text-to-image generation and obtained promising results.

Despite their potential, existing VAR-based methods [22, 49, 51] face a critical challenge: *the computational complexity and runtime latency scale dramatically with image resolution*. Specifically, unlike next-token prediction which processes only one token per step, the next-scale prediction of VAR requires processing the entire token map at each decoding step. As a result, the number of tokens increases in $\mathcal{O}(n^2)$ with the image resolution $n \times n$, and even leads to $\mathcal{O}(n^4)$ complexity in the attention [54] layer. Empirically, as shown in Fig. 2(a), even when FlashAttention [13] is enabled, VAR models still exhibit super-linear runtime latency. Consequently, this significant computational complexity prevents existing VAR-based models from scaling to higher resolutions, such as 2K.

In this work, we aim to address the resolution scaling challenge of VARs by pruning forwarded tokens. Through an in-depth analysis of the pre-trained VAR models, we identify the following key findings. ***1) Large-scale steps are speed bottleneck but appear robustness.*** Runtime profiling in Fig. 2(a) finds that the last two large-scale steps account

*Corresponding Authors, †Project Lead.



(a) Runtime at different scale steps (b) Efficiency-performance tradeoff

Figure 2. FastVAR exhibits promising resolution scalability, and can achieve noticeable speedup with negligible performance drop.

for 60% of the total runtime. Further investigation in Tab. 4 reveals that VAR is more resilient to token pruning at large-scale steps than at smaller scales. Thus, we direct our efforts to the large-scale steps. ***2) High-frequency modeling matters at large-scale steps.*** Spectrum analysis in Fig. 3(b) reveals that the large-scale steps are optimized to model high-frequency tokens, such as texture details, and the remaining low-frequency tokens almost converge in these steps. Consequently, we can only forward the high-frequency tokens for pruning. ***3) Tokens from different scales are related.*** Attention map analysis in Fig. 3(c) shows strong diagonal sparsity across scales, indicating that tokens attend not only to same-scale neighbors but also to those from the preceding scales. Thus, we cache tokens in previous steps to compensate for the pruned slots, preserving the 2D image lattice structure and maintaining information flow.

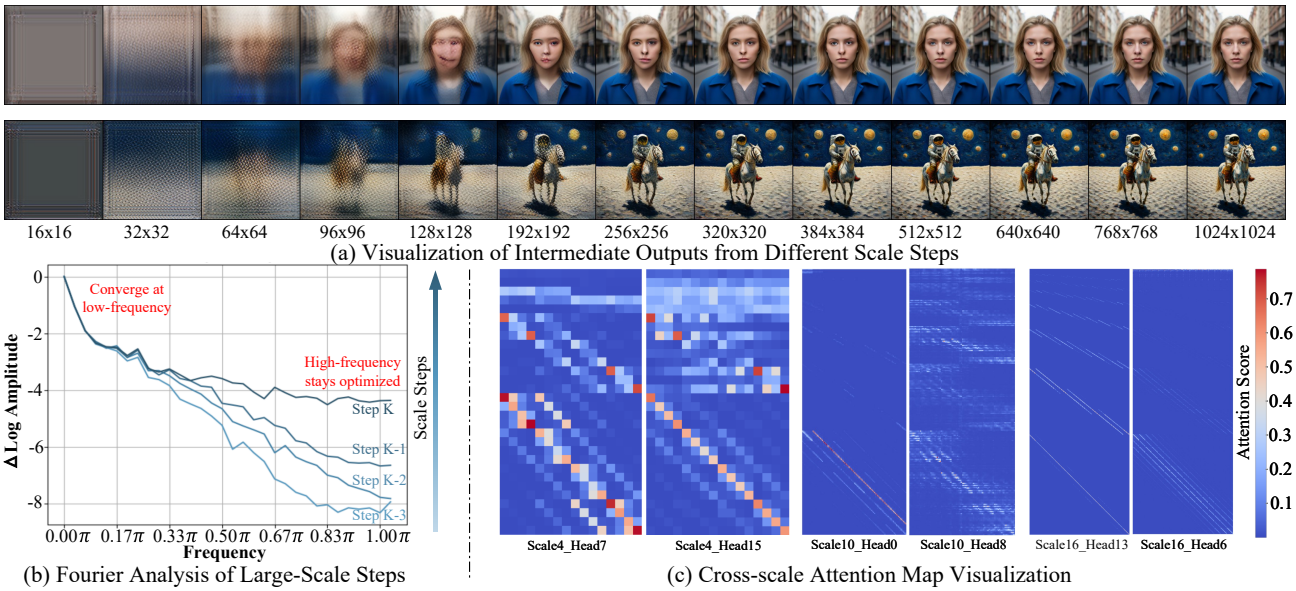
Based on the above observations, we propose FastVAR, a post-training acceleration recipe for efficient resolution scaling with VARs. At its core, FastVAR employs **"cached token pruning"**, which retains only pivotal tokens at large-scale steps to reduce computational overhead while using cached token maps from early-scale steps to compensate for information loss. To identify the pivotal tokens, we develop Pivotal Token Selection (PTS), a frequency-based scoring mechanism. Token importance is determined by filtering out the low-frequency component estimated via the direct current component, which enables efficient frequency-based token selection directly in the spatial domain. The selected Top-K pivotal tokens are then processed by the VAR model. To restore pruned tokens, we propose Cached Token Restoration (CTR), which first interpolates the token map from the cached scale and then reinstates the pruned tokens by indexing the interpolated token map to the pruned location. By integrating these strategies, we present FastVAR which enjoys the following benefits:

- The proposed method is training-free and plug-and-play for various VAR-based backbones.
- As shown in Fig. 2(b), FastVAR can be integrated with FlashAttention, with further 2.7× speedup and <1% performance drop.
- In Fig. 1, FastVAR facilitates zero-shot scaling to larger-resolution and can produce one 2K image using only 15GB memory in 1.5s on a single NVIDIA 3090 GPU.

2. Related Work

Autoregressive Visual Generation. The previous autoregressive (AR) methods [27, 42, 64, 67] mostly adopt the next-token prediction paradigm, which treats each pixel as one token and generates pixels in a GPT or BERT style [1, 4, 6, 14, 35]. For instance, some pioneering works [9, 52] generate pixels in raster scan order with transformer models. Later, VQVAE [53] and VQGAN [15] propose to quantize image patches into discrete tokens to ease the training process. Benefiting from the scaling up [24, 26], recent works have shown promising results that the autoregressive models can achieve a more competitive performance than state-of-the-art Diffusion models [7, 8, 38, 39, 41, 43]. Despite this progress, existing next-token generation pipelines struggle to efficiently synthesize high-resolution images due to the numerous decoding steps [10]. More recently, the Visual Autoregressive (VAR) modeling [51], which shifts to next-scale generation fashion to allow only a few forward steps, further advances the generation quality as well as the model efficiency. Building on top of VAR, existing methods have scaled up the model for production-level text-to-image generation. For example, HART [49] adopts the continuous diffusion module to compensate for the quantization error. Infinity [22] employs a bitwise tokenizer for an extremely large vector vocabulary. Despite the promising ability of VAR-based methods, they struggle to scale to larger resolutions with the increasing token numbers.

Efficient Visual Generation. To speed up diffusion models [25], many efforts have been made in recent years, including distillation [40, 44], quantization [21, 32, 46], pruning [2, 17, 55, 65, 68], and caching [30, 34, 36, 37], which either reduce the total denoising steps or to reduce the cost of single forward. Specifically, DeepCache [37] proposes to reuse the intermediate features of low-resolution layers in the U-Net. ToMeSD [2] merges similar tokens into one token with subsequent unmerging for acceleration. Unfortunately, these methods are specifically designed for diffusion, which can achieve only sub-optimal performance or cannot be used in VAR. In the efforts to accelerate AR modeling [31, 33, 48, 56, 61], one prevalent solution is to reduce the forward step using parallel decoding strategies [23, 45, 50]. For instance, speculative decoding [5, 28] utilizes a small draft model to generate candidate tokens, which are then verified in parallel by the larger model. However, the well-developed decoding strategy cannot be directly applied to the next-scale paradigm, whose each step involves multiple pixels [51]. As for fast VAR models, one related work is the CoDe [10], which uses model ensemble techniques to use a small model on the costly large resolution. However, CoDe highly depends on the availability of different-sized models. In this work, we attempt to speed up the VARs by pruning extra tokens, which is training-free and is generic for multiple VAR methods.



(a) Visualization of Intermediate Outputs from Different Scale Steps

(b) Fourier Analysis of Large-Scale Steps

(c) Cross-scale Attention Map Visualization

Figure 3. (a) We resize different $\tilde{r}_k$ to the same size for better presentation. (b) Each curve represents the frequency characteristics of a certain-scale token map. (c) The row represents keys of scale steps $\{1, 2, \cdots, k\}$, and the column is queries of the k-th scale step.

3. Method

3.1. Preliminary

The Visual Autoregressive (VAR) modeling [51] redefines the traditional AR by shifting from a "next-token prediction" to a "next-scale prediction", where each autoregressive unit is a token map of varying scales instead of a single token. For a given image feature map, VAR first quantizes it into K multi-scale token maps $\mathcal{R} = \{r_1, r_2, \cdots, r_K\}$ with progressively larger resolutions using pre-defined scale schdual $\{(h_1, w_1), (h_2, w_2), \cdots, (h_K, w_K)\}$. Then the $\mathcal{R}$ is used to train a Transformer [54] to learn the joint distribution using the causal formulation:

$$p(r_1, r_2, \cdots, r_K) = \prod_{k=1}^{K} p(r_k | r_1, r_2, \cdots, r_{k-1}), \quad (1)$$

where the start token r_1 is the encoded text embedding from the LLMs [11, 63] under given user prompts. During training, existing methods [22, 49, 51] usually employ a residual strategy to reduce learning difficulty. In detail, the model outputs at the k-th scale step f_k is the residual of the intermediate prediction $\tilde{r}_k$:

$$\tilde{r}_k = \text{interpolate}(\tilde{r}_{k-1}, (h_k, w_k)) + f_k, \quad (2)$$

where the "$\text{interpolate}(\cdot, (h_k, w_k))$" denotes upsample a token map to the size of (h_k, w_k). Unfolding Eq. (2) derives the following cumulative form:

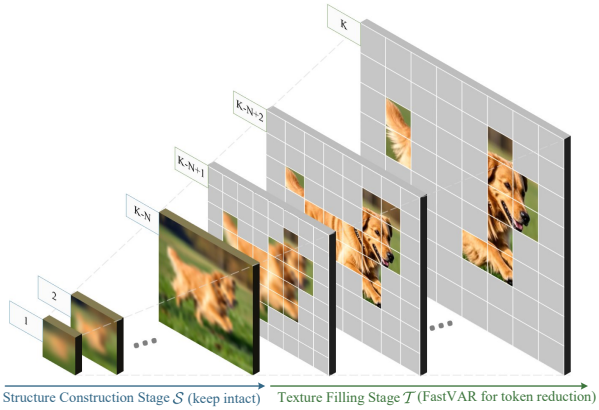
$$\tilde{r}_k = \sum_{i=1}^{k} \text{interpolate}(f_i, (h_k, w_k)). \quad (3)$$

In the last scale step, the predicted token maps $\tilde{r}_K$ is used to generate the final image. During inference, since the outputs at different scales are generated autoregressively, the KV-Cache can be adopted to avoid re-computing of previous steps. Although the VAR paradigm can generate images within only K scale steps, the Transformer needs to process all $h_k \times w_k$ tokens at the k-th step, hindering scaling to higher-resolution image synthesis.

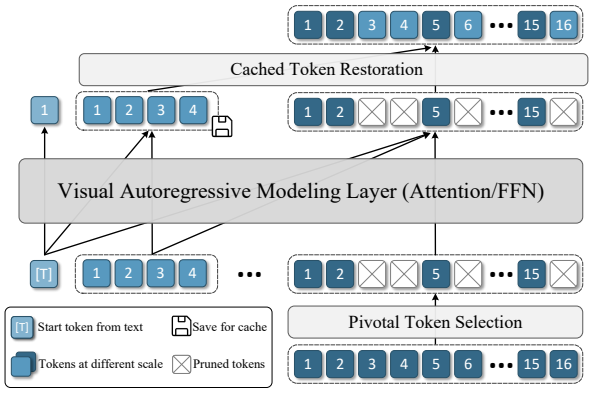
3.2. Motivation

In this work, we aim to approach linear VAR models to advance high-resolution image generation. To this end, we investigate the scale-wise behavior of the pre-trained VAR model [22]. We examine from three angles: the latency profiling, the specific role of each step, and the cross-scale token dependency. Conclusions are summarized as follows.

Observation 1: Large-scale steps are speed bottleneck but appear robustness. As shown in the runtime curve in Fig. 2(a), even the FlashAttention is enabled, the inference latency of the VAR exhibits a super-linear complexity with resolution steps. This is because the Transformer needs to take all the tokens at once. As a result, the tokens from the large-scale steps make up the majority of the total token length. For example, the last two steps even occupy 60% of the total time. On the other hand, the experiments in Sec. 4.3 suggest the large-scale steps are more robust to token drop than small-scale counterparts under the same pruning ratio. Therefore, it is accessible to prune some "unimportant tokens" at the large scale step to speed up token processing. However, challenges still remain about which tokens should be pruned and how to compensate for the information loss from these pruned tokens.



(a) Overall pipeline of the proposed FastVAR



(b) Token-wise implementation for one VAR layer at stage $\mathcal{T}$

Figure 4. (a) We keep the **Structure Construction Stage** as standard VAR and additionally store token maps at the $(K-N)$-th step as the cache. Then the **Texture Filling Stage** applies FastVAR to process only pivotal tokens for fast inference. (b) For Attention or FFN layers using FastVAR, we first employ the Pivotal Token Selection (PTS) for token reduction. After the model forward, we develop the Cached Token Restoration (CTR) to restore the original token numbers using the caches from the previous scale step.

Observation 2: High-frequency modeling matters at large-scale steps. To find which token should be pruned, we visualize the intermediate predictions $\tilde{r}_k$ in Fig. 3(a). Interestingly, we find that the generation process of the pre-trained VAR model can be decomposed into two stages. The Structure Construction Stage, which includes small-scale steps, is trained to generate the outline of a subject. After that, at the Texture Filling Stage, which is composed of large-scale steps, is responsible for adding details based on the previous sketch. Given that the outline is low-frequency while the texture is high-frequency, we therefore infer that the VAR mainly models the high-frequency texture in the large-scale stage while keeping the low-frequency structure almost intact. For further verification, we show in Fig. 3(b) that the low-frequency component almost converges in the large-scale steps, while the high-frequency modeling stays optimized with noticeable variations. Therefore, we can prune the redundant low-frequency tokens and forward only the pivotal high-frequency tokens at the large-scale steps for token number reduction.

Observation 3: Tokens from different scales are related. After reducing the input tokens for fast forward, we still need to restore the output tokens to the original numbers in order to maintain the 2D image structure. To this end, we analyze the cross-scale attention map of the unpruned VAR model, where the query at the current scale interacts with KV caches from all previous scales. As depicted in Fig. 3(c), we observe that the attention map not only exhibits a high diagonal score at self-scale, but also a high diagonal score at cross-scale. This behavior indicates that a token in the current scale not only attends to its same-scale locals but also exhibits strong correlation to tokens at the neighboring positions in previous scales. This diagonal attention sparsity thus facilitates us to use only a few tokens to estimate the original output of the pruned slots, and avoids weighting multiple tokens across the token map. Specifically, we can approximate the outputs of pruned tokens by indexing in the previous-scale token maps based on the pruning location. In this way, the information loss can be compensated thanks to this high cross-scale similarity.

3.3. Efficient Resolution Scaling for VARs

We instantiate the above idea and present FastVAR to approximate linear VAR models. The core is the cached token pruning, which prunes tokens at large scale steps while using cache from early scales to maintain information flow.

Overview. As shown in Fig. 4(a), denote the scale step set of Structure Construction Stage as $\mathcal{S} = \{1, 2, \cdots, K-N\}$, the step set of Texture Filling Stage as $\mathcal{T} = \{K-N+1, \cdots, K\}$. As stated in Sec. 3.2, the speedup of pruning token maps in $\mathcal{S}$ is insignificant while producing a noticeable performance drop, we therefore keep $\mathcal{S}$ intact as the standard VAR. In addition, we save the per-layer outputs at the last scale in $\mathcal{S}$, *i.e.*, the $(K-N)$-th step, as the cache for subsequent token restoration. For the set $\mathcal{T}$, we apply FastVAR to prune tokens for fast-forward, followed by the token number restoration. More details are given below.

Pivotal Token Selection. We first introduce the Pivotal Token Selection (PTS) to reduce the number of tokens. As observed in Fig. 3(b), the low-frequency tokens almost converge in the later scale steps. Therefore, we can feed only the high-frequency tokens and prune the remaining. However, the challenge lies in that common frequency operators, such as FFT, work in the frequency domain, making it difficult to recognize the frequency characteristics of certain tokens in the original token map. To this end, our proposed PTS adopts an approximation solution. Specifically, denote x_k as the input of one layer, we estimate its low-frequency

component $\bar{x}_k$ as the direct current component, which can be easily obtained via the spatially global average pooling:

$$\bar{x}_k = \text{global\_avg\_pool}(x_k). \tag{4}$$

After that, the high-frequency component is computed as the difference between the x_k and $\bar{x}_k$. The pivotal score s_k is defined as the L2 norm of the high-frequency maps:

$$s_k = ||x_k - \bar{x}_k||_2. \tag{5}$$

Subsequently, we obtain the index set $\mathcal{I}$ for pivotal token selection by keeping TopK scores in s_k. Finally, we perform token pruning on x_k through indexing with $\mathcal{I}$ to allow the fast model forward. Notably, since the PTS reduces the number of input tokens in one transformer layer, as an additional benefit, the KV-cache is also reduced accordingly, thus optimizing the GPU footprints as well as the cross-scale attention for the subsequent scale steps.

Cached Token Restoration. The proposed PTS can effectively reduce the number of tokens. However, the visual generation tasks need to restore the original token numbers to regain the structural 2D image. For this reason, we propose Cached Token Restoration (CTR). The main rationale stems from the diagonal attention sparsity that the token at the k-th scale not only attends to itself, but also exhibits strong correlation with the corresponding slot from previous $\{1, 2, , \cdots, k-1\}$ steps. Therefore, our proposed CTR approximates the layer outputs in the pruned locations using the counterparts from the cached step. Formally, given the cached token map y_{K-N} with size (h_{K-N}, w_{K-N}), which is the output of the corresponding layer at step $K-N$, we first upsample it to the same size of the current scale:

$$y_k^{cache} = \text{interpolate}(y_{K-N}, (h_k, w_k)). \tag{6}$$

Then the value from y_k^{cache} is scattered into the pruned slots of y_k using the index set $\mathcal{I}$, to generate the restored layer output y'_k, which shares the same token numbers as the x_k. We also provide an explanation for the choice of $K-N$ as the caching step, see *Suppl.* for more details.

Implementation of FastVAR. Thanks to the token number invariance of the PTS+CTR, as shown in Fig. 4(b), we apply this operator pair for each Attention&FFN layer at the large-scale steps. Furthermore, we empirically find in Sec. 4.3 that the larger scale steps exhibit stronger robustness to pruning than that of early scale steps. Therefore, we develop a progressive pruning ratio schedule, in which we assign larger pruning ratios to larger scale steps in the set $\mathcal{T}$. We summarize the complete algorithm of FastVAR in the *Suppl.*. At last, it is worth noting that our FastVAR does not access the attention map, and we show in Sec. 4.4 that the proposed FastVAR can be used in combination with other acceleration techniques like FlashAttention [12, 13] for even faster VAR generation.

4. Experiments

4.1. Experimental Setup

Models and Evaluations. We apply our FastVAR on two VAR-based text-to-image models, namely HART [49] and Infinity [22]. Both models can generate images of up to 1024×1024 resolution. For a fair comparison, we keep all the hyperparameters the same as their default settings. For evaluation metrics, we compare both in terms of generation quality and inference efficiency. For generation quality, we use two popular benchmarks, *i.e.*, the GenEval [20] and MJHQ30K [29] to validate the high-level semantic consistency and the perceptual quality, respectively. For efficiency evaluation, we adopt metrics including running time, throughputs, speedup ratio, and GPU memory costs.

Implementation Details. For the number of pruned scale steps N, we set $N = 4$ for Infinity and $N = 2$ for HART, *i.e.*, only token maps from the last 4 or 2 scale steps are applied with the proposed FastVAR, with other steps kept as standard VAR. For the progressive pruning ratio schedule, we set it to $\{40\%, 50\%, 100\%, 100\%\}$ for Infinity, and $\{50\%, 75\%\}$ for HART. The 100% pruning ratio indicates all tokens are dropped, *i.e.*, we skip the corresponding steps and interpolate the intermediate outputs to the target resolution as the final outputs. Note that this extreme ratio depends on the pruned backbones, and we provide further discussion in the *Suppl.*. Unless specified, the efficiency of the unpruned baselines are already accelerated by FlashAttention [13], and we apply the proposed FastVAR on top of it. Following the setup of existing methods [10, 48, 51], the inference speeds for all methods are measured without including the VAE time cost since this is a shared cost for all methods. All experiments are conducted on one single NVIDIA RTX 3090 GPU with 24GB memory.

4.2. Main Results

Comparison on GenEval. We first evaluate the quality-efficiency trade-off on 1024×1024 text-to-image generation using the GenEval benchmark [20]. We compare our FastVAR against various state-of-the-art methods, including Diffusion, AR, and VAR models. The results are shown in Tab. 1. Compared with traditional AR models, our Infinity+FastVAR can achieve **even 39.7**$\times$ speedup than LlamaGen [48], while achieving **125%** performance boosts on the GenEval. Moreover, our FastVAR achieves acceleration almost without performance loss, *e.g.*, **1.5**$\times$ speedup on the HART with the same GenEval score as the unpruned baseline. Our approach also reduces the inference GPU memory. For example, FastVAR achieves **22.2%** reduction to **14.7GB** on top of FlashAttention of 18.9GB, to produce a 1024×1024 image in **0.95s**, facilitating generation on consumer-level GPUs. The above results demonstrate the generality and effectiveness of our method.

Table 1. Quantitative comparison on efficiency and quality on 1024×1024 GenEval benchmarks. The marks ●, ○, and ◇ denote the Diffusion, AR, and VAR-based methods, respectively. Note that the efficiency of HART and Infinity baselines are tested under FlashAttention.

Methods	Inference Efficiency					Generation Quality			
	#Steps↓	Speedup↑	Latency↓	Throughput↑	#Param↓	two_object↑	position↑	color_attr↑	Overall↑
●SDXL [41]	40	-	4.3s	0.23it/s	2.6B	0.74	0.15	0.23	0.55
●PixArt-Sigma [8]	20	-	2.7s	2.50it/s	0.6B	0.62	0.14	0.27	0.55
●SD3-medium [16]	28	-	4.4s	3.45it/s	2.0B	0.74	0.34	0.36	0.62
○LlamaGen [48]	1024	-	37.7s	0.03it/s	0.8B	0.34	0.07	0.04	0.32
○Show-o [61]	1024	-	50.3s	0.02it/s	1.3B	0.80	0.31	0.50	0.68
◇HART [49]	14	1.0×	0.95s	1.05it/s	0.7B	0.62	0.13	0.18	0.51
◇**+ FastVAR**	14	1.5×	0.63s	1.59it/s	0.7B	0.57	0.16	0.24	0.51
◇Infinity [22]	13	1.0×	2.61s	0.38it/s	2.0B	0.85	0.44	0.53	0.73
◇**+ FastVAR**	13	2.7×	0.95s	1.05it/s	2.0B	0.81	0.45	0.52	0.72

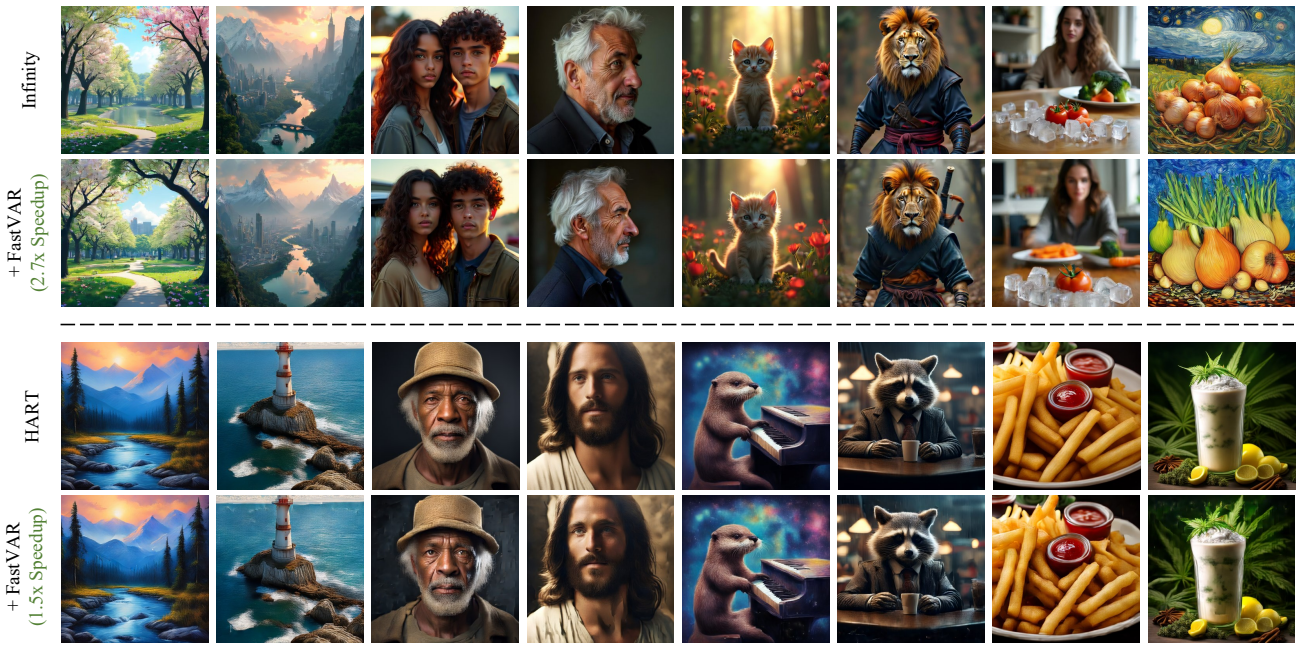


Figure 5. Qualitative comparison between the original baselines and our proposed FastVAR on 1024×1024 image generation. Our FastVAR achieves significant speedups, *e.g.*, 2.7× on Infinity [22], while keeping high-quality results similar to the original model.

Comparison on MJHQ30K. In Tab. 2, we further validate the perceptual quality on the MJHQ30K [29] benchmark. It can be seen that our FastVAR achieves a reasonable performance while maintaining a high speedup ratio. For instance, on the well-known challenging "people" category, our HART+FastVAR even achieves a FID reduction of **2.42** with **1.5**× acceleration. In other categories, our method also maintains good performance with significant acceleration. We also give a qualitative visualization in Fig. 5, and one can see that the generated images from FastVAR maintain exceptionally high quality and accurate semantic information, indicating the effectiveness of the proposed method.

Difference from ToMe. Token Merging (ToMe) [3], which merges multiple tokens into one token for efficient forward followed by token unmerging to restore the 2D shape, appears applicable to accelerate VAR backbones due to its token number invariance. In Tab. 3, we set different FastVAR and ToMe variants using varying pruning ratios. One can see that ToMe fails to achieve high speedup. For example, even the 1.36× speedup can lead to noticeable FID degradation since it is difficult to compress the whole token map into limited tokens. In contrast, our FastVAR can achieve 1.7× higher speedup with better FID performance.

Table 2. Quantitative comparisons of FID and CLIP score with different generation categories on the MJHQ30K dataset.

Methods	Speedup	landscape		people		food	
		FID↓	CLIP↑	FID↓	CLIP↑	FID↓	CLIP↑
SDXL [41]	-	30.78	26.35	35.56	28.01	35.26	27.98
HART [49]	1.0×	25.43	26.82	30.61	28.47	30.37	28.03
+ FastVAR	1.5×	22.52	26.51	28.19	28.34	30.97	28.25
Infinity [22]	1.0×	24.68	26.62	30.27	27.82	31.55	26.66
+ FastVAR	2.7×	24.68	26.62	30.55	28.28	32.54	27.08

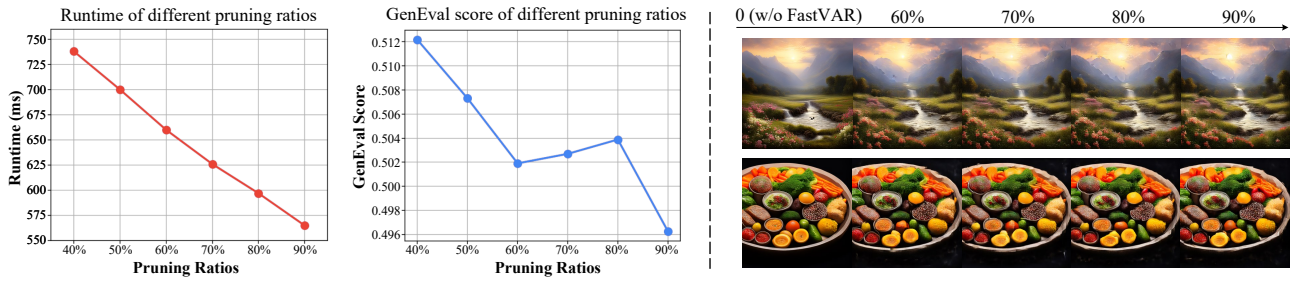


Figure 6. Quantitative and qualitative ablation experiments on different pruning ratios. We set varying pruning ratios on the last scale step of the HART model while keeping the other scale steps as the original setup.

Table 3. Quantitative comparison results with ToMe [2] under different speedup settings.

Methods	Speedup↑	Latency↓	Throughput↑	FID↓	CLIP↑	GenEval↑
HART [49]	1.00×	0.95s	1.05it/s	30.61	28.47	0.51
+ ToMe [2]	1.19×	0.80s	1.25it/s	29.07	28.33	0.48
+ ToMe [2]	1.36×	0.70s	1.43it/s	35.22	27.99	0.46
+ FastVAR	1.51×	0.63s	1.59it/s	28.19	28.34	0.51
+ FastVAR	1.70×	0.56s	1.79it/s	28.97	28.24	0.50

Table 4. Ablation experiments on the scale-wise sensitivity. We focus on the scale range [16,21,27,36,48,64] with every two consecutive scales as a pruning group. The FastVAR is applied to the selected group while keeping the others unpruned.

scales	ratio	Speedup↑	Latency↓	FID↓	CLIP↑	GenEval↑
baseline	0	1.00×	0.95s	30.61	28.47	0.51
[16, 21]	75%	1.01×	0.94s	34.27	28.43	0.48
[27, 36]	50%	1.09×	0.87s	27.92	28.49	0.51
[27, 36]	75%	1.14×	0.83s	29.38	28.56	0.50
[48, 64]	50%	1.30×	0.73s	27.13	28.43	0.51
[48, 64]	75%	1.64×	0.58s	28.56	28.36	0.50

4.3. Ablation Studies

Different Pruning Ratios. The pruning ratio plays an important role in balancing between efficiency and performance. Here, we conduct ablation experiments to investigate the impact of different pruning ratios. Both the quantitative and qualitative results are given in Fig. 6. Intuitively, increasing the pruning ratio leads to a stable runtime reduction since the model only needs to process fewer tokens. However, an over-large pruning ratio can also bring performance degradation due to the information loss of pivotal high-frequency tokens. Furthermore, the quantitative visualization suggests that increasing the pruning ratio makes certain textures and details discontinuous since the cache from the previous scale is not always an optimal approximation of the pruned tokens. Therefore, a moderate pruning ratio, *e.g.*, 40%-75%, is adopted for HART+FastVAR to strike a sweet spot between performance and efficiency.

Scale-wise Sensitivity. As discussed in Sec. 3.2, we only focus on the token pruning of the last few scale steps. Here, we conduct ablation experiments in Tab. 4 to justify the rationality. It can be seen that the acceleration from pruning at an early scale steps is very limited due to the small token map size. For example, even reducing 50% tokens at the 48 and 64 scale steps is significantly faster than that of 75% at 16 and 21 scale steps. In addition, pruning at early scales also leads to a noticeable performance drop on both perceptual quality and semantic consistency. For instance, pruning at 48 and 64 scale steps brings 5.71 lower FID compared to that of 16 and 21 scale steps. This is because token reduction at small-scale steps disrupts the structure construction stage, and the inaccurate subject structure can exacerbate errors in subsequent scale steps.

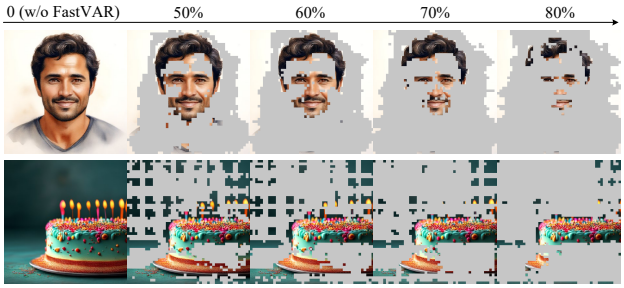


Figure 7. Visualization of pruned tokens with different ratios.

Visualization of Pruned Slots. In Fig. 7, we visualize the selected TopK pivotal tokens under different pruning ratios. For a given pruning ratio, *e.g.*, 50%, our FastVAR prioritizes the retention of high-frequency edges or texture slots, and removes tokens from flat regions that have already converged at early scale steps. Furthermore, the importance order within the selected tokens is also meaningful. For example, in the first row, when increasing the pruning ratio from 50% to 90%, the human eyes, hair and mouth regions, which are the most detailed, are consistently preserved while the low-frequency cheek regions are gradually pruned. Given that our FastVAR is gradient-free, this quantitative result confirms that the proposed frequency-based pivotal token selection is a reasonable heuristic.

4.4. Discussion

In Conjunction with FlashAttention. Existing token pruning methods [18, 19, 60, 66, 68] often depend on the attention map for token importance, leading to difficulties in combination with FlashAttention [13] in which the attention map is inaccessible. In comparison, our FastVAR

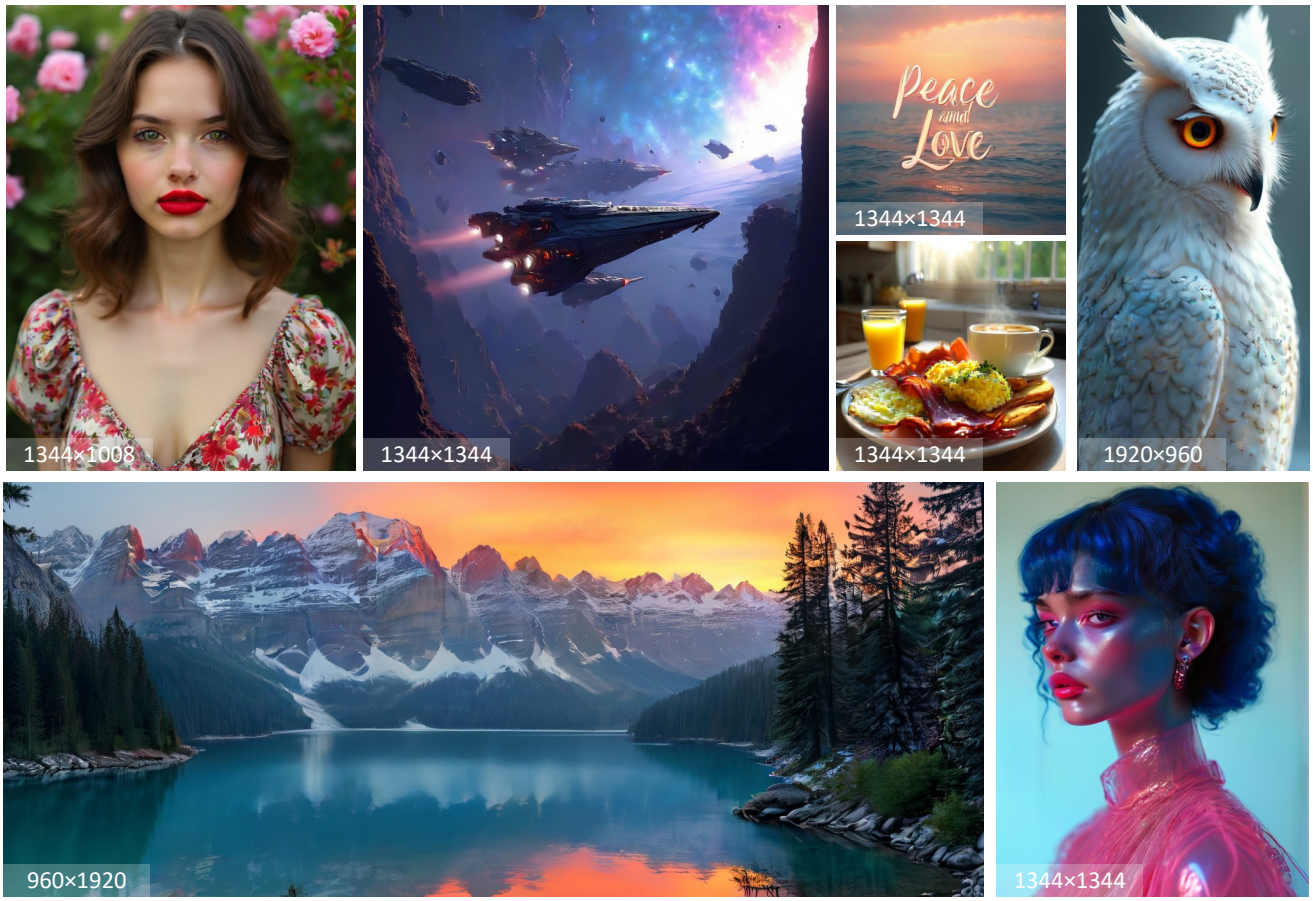


Figure 8. We extend the scale schedule of pre-trained VAR model [22] for zero-shot larger resolution generation. We chose a modest extended scale since a larger one is observed quality drop due to the resolution gap. The images are re-scaled for better presentation.

can be seamlessly integrated with FlashAttention to facilitate more efficient inference. In Tab. 5, we show that using the FastVAR-only can obtain even **2.1**× speedup against FlashAttention only, with **20.4%** GPU memory reduction. After combining both, the resulting version leads to **2.7**× speedup than FlashAttention only. This experiment validates that our FastVAR can be orthogonally combined with other methods for further acceleration.

Towards Larger Resolution. As presented in Sec. 3.2, the original VAR struggles to scale to higher-resolution due to the increasing token numbers. Our FastVAR can facilitate efficient resolution scaling, offering the potential to generate images at a larger resolution. To this end, we apply our FastVAR to the Infinity [22] for zero-shot larger resolution synthesis by appending additional steps. The results are shown in Fig. 8. It is noteworthy that even the FlashAttention accelerated baseline is **out of memory** on a 24GB NVIDIA 3090 GPU. In contrast, our FastVAR costs only **15GB** memory and **1.3s** runtime to generate a 1344×1344 image. Moreover, the generated images are of high quality, suggesting that our FastVAR can facilitate production-level image generation on consumer-level GPUs.

Table 5. Efficiency comparison with FlashAttn on 1024×1024 image generation.

setups	Speedup↑	Latency↓	Throughput↑	Memory↓
SlowAttn [54]	-	-	-	OOM
FlashAttn only [13]	1.0×	2.61s	0.38it/s	16.1GB
SlowAttn+FastVAR	2.1×	1.25s	0.80it/s	12.8GB
FlashAttn+FastVAR	2.7×	0.95s	1.05it/s	11.9GB

5. Conclusion

This work presents FastVAR to advance high-resolution image synthesis with VARs by addressing the resolution scaling challenge. We reveal that the main latency bottleneck is the large-scale steps at which the low-frequency tokens have almost converged. To this end, we develop the cached token pruning technique to allow only process pivotal high-frequency tokens and use caches from the previous scale steps to restore the original token numbers. Thanks to the reduced forwarded tokens, our FastVAR approximates linear complexity, further enabling generation at a larger resolution such as 2K. Extensive experiments validate our FastVAR as an effective and generic solution for efficient resolution scaling of VAR models.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 2

[2] Daniel Bolya and Judy Hoffman. Token merging for fast stable diffusion. In *CVPR*, pages 4599–4603, 2023. 2, 7

[3] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token Merging: Your ViT but faster. *arXiv preprint arXiv:2210.09461*, 2022. 6, 12

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *NeurIPS*, 33:1877–1901, 2020. 2

[5] 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. *arXiv preprint arXiv:2401.10774*, 2024. 2

[6] Huiwen Chang, Han Zhang, Jarred Barber, AJ Maschinot, Jose Lezama, Lu Jiang, Ming-Hsuan Yang, Kevin Murphy, William T Freeman, Michael Rubinstein, et al. Muse: Text-to-image generation via masked generative transformers. *arXiv preprint arXiv:2301.00704*, 2023. 2

[7] Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, et al. PixArt-Alpha: Fast training of diffusion transformer for photorealistic text-to-image synthesis. *arXiv preprint arXiv:2310.00426*, 2023. 2

[8] Junsong Chen, Chongjian Ge, Enze Xie, Yue Wu, Lewei Yao, Xiaozhe Ren, Zhongdao Wang, Ping Luo, Huchuan Lu, and Zhenguo Li. PixArt-Sigma: Weak-to-strong training of diffusion transformer for 4k text-to-image generation. In *ECCV*, pages 74–91. Springer, 2024. 2, 6

[9] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *ICML*, pages 1691–1703. PMLR, 2020. 2

[10] Zigeng Chen, Xinyin Ma, Gongfan Fang, and Xinchao Wang. Collaborative decoding makes visual auto-regressive modeling efficient. *arXiv preprint arXiv:2411.17787*, 2024. 2, 5, 12

[11] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024. 3

[12] Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *ICLR*, 2024. 5

[13] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flash Attention: Fast and memory-efficient exact attention with IO-awareness. In *NeurIPS*, 2022. 1, 5, 7, 8

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *ACL*, pages 4171–4186, 2019. 2

[15] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, pages 12873–12883, 2021. 2

[16] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *ICML*, 2024. 6

[17] Gongfan Fang, Xinyin Ma, and Xinchao Wang. Structural pruning for diffusion models. In *NeurIPS*, 2023. 2

[18] Qichen Fu, Minsik Cho, Thomas Merth, Sachin Mehta, Mohammad Rastegari, and Mahyar Najibi. LazyLLM: Dynamic token pruning for efficient long context llm inference. *arXiv preprint arXiv:2407.14057*, 2024. 7

[19] Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive KV cache compression for LLMs. *arXiv preprint arXiv:2310.01801*, 2023. 7

[20] Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. GenEval: An object-focused framework for evaluating text-to-image alignment. *NeurIPS*, 36, 2024. 5, 12

[21] Hang Guo, Yawei Li, Tao Dai, Shu-Tao Xia, and Luca Benini. Intlora: Integral low-rank adaptation of quantized diffusion models. *arXiv preprint arXiv:2410.21759*, 2024. 2

[22] Jian Han, Jinlai Liu, Yi Jiang, Bin Yan, Yuqi Zhang, Zehuan Yuan, Bingyue Peng, and Xiaobing Liu. Infinity: Scaling bitwise autoregressive modeling for high-resolution image synthesis. *arXiv preprint arXiv:2412.04431*, 2024. 1, 2, 3, 5, 6, 8, 13, 14

[23] Yefei He, Feng Chen, Yuanyu He, Shaoxuan He, Hong Zhou, Kaipeng Zhang, and Bohan Zhuang. ZIPAR: Accelerating autoregressive image generation through spatial locality. *arXiv preprint arXiv:2412.04062*, 2024. 2

[24] Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B Brown, Prafulla Dhariwal, Scott Gray, et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020. 2

[25] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 33:6840–6851, 2020. 2

[26] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. 2

[27] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *CVPR*, pages 11523–11532, 2022. 2

[28] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *ICML*, pages 19274–19286. PMLR, 2023. 2

[29] Daiqing Li, Aleks Kamko, Ehsan Akhgari, Ali Sabet, Linmiao Xu, and Suhail Doshi. Playground v2.5: Three insights

towards enhancing aesthetic quality in text-to-image generation. *arXiv preprint arXiv:2402.17245*, 2024. 5, 6, 12, 13

[30] Senmao Li, Taihang Hu, Fahad Shahbaz Khan, Linxuan Li, Shiqi Yang, Yaxing Wang, Ming-Ming Cheng, and Jian Yang. Faster Diffusion: Rethinking the role of unet encoder in diffusion models. *NeurIPS*, 2023. 2

[31] Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. *arXiv preprint arXiv:2406.11838*, 2024. 1, 2

[32] Xiuyu Li, Yijiang Liu, Long Lian, Huanrui Yang, Zhen Dong, Daniel Kang, Shanghang Zhang, and Kurt Keutzer. Q-Diffusion: Quantizing diffusion models. In *ICCV*, pages 17535–17545, 2023. 2

[33] Dongyang Liu, Shitian Zhao, Le Zhuo, Weifeng Lin, Yu Qiao, Hongsheng Li, and Peng Gao. Lumina-mgpt: Illuminate flexible photorealistic text-to-image generation with multimodal generative pretraining. *arXiv preprint arXiv:2408.02657*, 2024. 1, 2

[34] Haozhe Liu, Wentian Zhang, Jinheng Xie, Francesco Faccio, Mengmeng Xu, Tao Xiang, Mike Zheng Shou, Juan-Manuel Perez-Rua, and Jürgen Schmidhuber. Faster diffusion via temporal attention decomposition. *arXiv preprint arXiv:2404.02747*, 2024. 2

[35] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019. 2

[36] Xinyin Ma, Gongfan Fang, Michael Bi Mi, and Xinchao Wang. Learning-to-Cache: Accelerating diffusion transformer via layer caching. *arXiv preprint arXiv:2406.01733*, 2024. 2

[37] Xinyin Ma, Gongfan Fang, and Xinchao Wang. DeepCache: Accelerating diffusion models for free. In *CVPR*, pages 15762–15772, 2024. 2

[38] Yue Ma, Yingqing He, Xiaodong Cun, Xintao Wang, Siran Chen, Xiu Li, and Qifeng Chen. Follow your pose: Pose-guided text-to-video generation using pose-free videos. In *AAAI*, pages 4117–4125, 2024. 2

[39] Yue Ma, Hongyu Liu, Hongfa Wang, Heng Pan, Yingqing He, Junkun Yuan, Ailing Zeng, Chengfei Cai, Heung-Yeung Shum, Wei Liu, et al. Follow-your-emoji: Fine-controllable and expressive freestyle portrait animation. In *SIGGRAPH Asia*, pages 1–12, 2024. 2

[40] Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *CVPR*, pages 14297–14306, 2023. 2

[41] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023. 2, 6

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

[43] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, pages 10684–10695, 2022. 2

[44] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022. 2

[45] Andrea Santilli, Silvio Severino, Emilian Postolache, Valentino Maiorca, Michele Mancusi, Riccardo Marin, and Emanuele Rodolà. Accelerating transformer inference for translation via parallel decoding. *arXiv preprint arXiv:2305.10427*, 2023. 2

[46] Yuzhang Shang, Zhihang Yuan, Bin Xie, Bingzhe Wu, and Yan Yan. Post-training quantization on diffusion models. In *CVPR*, pages 1972–1981, 2023. 2

[47] Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling test-time compute optimally can be more effective than scaling LLM parameters. In *ICLR*, 2025. 13

[48] 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, 5, 6

[49] Haotian Tang, Yecheng Wu, Shang Yang, Enze Xie, Junsong Chen, Junyu Chen, Zhuoyang Zhang, Han Cai, Yao Lu, and Song Han. HART: Efficient visual generation with hybrid autoregressive transformer. *arXiv preprint arXiv:2410.10812*, 2024. 1, 2, 3, 5, 6, 7, 13

[50] Yao Teng, Han Shi, Xian Liu, Xuefei Ning, Guohao Dai, Yu Wang, Zhenguo Li, and Xihui Liu. Accelerating auto-regressive text-to-image generation with training-free speculative jacobi decoding. *arXiv preprint arXiv:2410.01699*, 2024. 2

[51] Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *NeurIPS*, 2024. 1, 2, 3, 5, 12

[52] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. *NeurIPS*, 29, 2016. 2

[53] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *NeurIPS*, 30, 2017. 2

[54] A Vaswani. Attention is all you need. *NeurIPS*, 2017. 1, 3, 8

[55] Hongjie Wang, Difan Liu, Yan Kang, Yijun Li, Zhe Lin, Niraj K Jha, and Yuchen Liu. Attention-driven training-free efficiency enhancement of diffusion models. In *CVPR*, pages 16080–16089, 2024. 2

[56] Xinlong Wang, Xiaosong Zhang, Zhengxiong Luo, Quan Sun, Yufeng Cui, Jinsheng Wang, Fan Zhang, Yueze Wang, Zhen Li, Qiying Yu, et al. EMU3: Next-token prediction is all you need. *arXiv preprint arXiv:2409.18869*, 2024. 1, 2

[57] Chengyue Wu, Xiaokang Chen, Zhiyu Wu, Yiyang Ma, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, Chong Ruan, et al. Janus: Decoupling visual encoding for unified multimodal understanding and generation. *arXiv preprint arXiv:2410.13848*, 2024. 1

[58] Xiaoshi Wu, Yiming Hao, Keqiang Sun, Yixiong Chen, Feng Zhu, Rui Zhao, and Hongsheng Li. Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis. *arXiv preprint arXiv:2306.09341*, 2023. 12

[59] Yecheng Wu, Zhuoyang Zhang, Junyu Chen, Haotian Tang, Dacheng Li, Yunhao Fang, Ligeng Zhu, Enze Xie, Hongxu Yin, Li Yi, et al. VILA-U: A unified foundation model integrating visual understanding and generation. *arXiv preprint arXiv:2409.04429*, 2024. 1

[60] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023. 7

[61] 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. 1, 2, 6

[62] Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. *Advances in Neural Information Processing Systems*, 36:15903–15935, 2023. 12

[63] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. 3

[64] Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. Vector-quantized image modeling with improved VQGAN. *arXiv preprint arXiv:2110.04627*, 2021. 2

[65] Evelyn Zhang, Bang Xiao, Jiayi Tang, Qianli Ma, Chang Zou, Xuefei Ning, Xuming Hu, and Linfeng Zhang. Token pruning for caching better: 9 times acceleration on stable diffusion for free. *arXiv preprint arXiv:2501.00375*, 2024. 2

[66] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2O: Heavy-hitter oracle for efficient generative inference of large language models. *NeurIPS*, 36:34661–34710, 2023. 7

[67] Chuanxia Zheng, Tung-Long Vuong, Jianfei Cai, and Dinh Phung. MOVQ: Modulating quantized vectors for high-fidelity image generation. *NeurIPS*, 35:23412–23425, 2022. 2

[68] Chang Zou, Xuyang Liu, Ting Liu, Siteng Huang, and Linfeng Zhang. Accelerating diffusion transformers with token-wise feature caching. *arXiv preprint arXiv:2410.05317*, 2024. 2, 7

FastVAR: Linear Visual Autoregressive Modeling via Cached Token Pruning

Supplementary Material

A. Results on ImageNet with VAR

In the main paper, we focus on the performance of our FastVAR on high-resolution image generation tasks. As stated in Section 3.2, applying token pruning on early small scale steps can lead to performance degradation due to the interference of the structure construction. Given that pruning on small token maps can not bring significant speedups, we thus do not design over-complex algorithms to further accelerate small-scale steps. However, for the sake of experimental completeness, we give the results of our FastVAR on the 256×256 class conditional generation on ImageNet in Tab. A.1. It can be seen that our FastVAR can achieve very competitive performance against existing methods, while maintaining a high speedup ratio. Since the VAE in the existing VAR methods adopts a high compression rate, *e.g.*, 16× downsample, the token map resolution at the largest scale in the VAR model [51] is only 16×16 for the 256×256 image generation. As a result, token pruning on this small-scale generation is much less robust compared to the larger resolution, *e.g.*, 1024×1024 resolution. We leave it for future work to design more generalized strategies to further include token maps at small scales.

B. Further Efficiency Profiling

As demonstrated in the experiments, our FastVAR can achieve significant speedup without performance degradation, *e.g.*, 1.5× speedup for the HART backbone. However, this speedup ratio still shares some similar latencies as the unpruned benchmark, such as the forward pass at small scale steps. Here, we give a more fine-grained speedup comparison by directly comparing the attention and FFN under the condition of with and without the proposed FastVAR. As illustrated in Fig. A.1, FastVAR (ratio=75%) can bring even a 4.6× speedup for the attention and a 3.8× speedup for the FFN. This result demonstrates a promising speedup upper bound of our FastVAR.

Furthermore, compared to the runtime of the standard benchmark, our FastVAR adds additional token importance calculation, as well as token number restoration, which may introduce additional time. Here, we give experimental results to validate the efficiency of our FastVAR. As shown in Fig. A.1, the additional computational cost accompanying our FastVAR is almost negligible. For example, the proposed PTS occupies only 0.59 ms, while the CTR occupies 0.24 ms. Thus the total additional latency from our FastVAR, *i.e.*, 0.63 ms, occupies only 5% of the original attention module, which is significantly lower than the speedup brought from FastVAR.

Table A.1. Quantitative comparison on 256×256 generation on ImageNet.

Methods	#param	runtime	memory	IS↑	FID↓	Precision↑	Recall↑
VAR(d=24) [51]	1.0B	1.2s	14GB	313.7	2.29	82.50	57.45
VAR(d=30) [51]	2.0B	2.2s	22GB	306.6	2.05	81.76	58.20
CoDe(N=8) [10]	2.3B	1.7s	15GB	300.4	2.26	81.31	58.63
CoDe(N=9) [10]	2.3B	2.2s	16GB	297.2	2.16	81.07	59.03
FastVAR(d=24)	1.0B	1.1s	13GB	287.4	2.64	79.76	58.22
FastVAR(d=30)	2.0B	1.9s	15GB	288.7	2.30	80.72	58.64

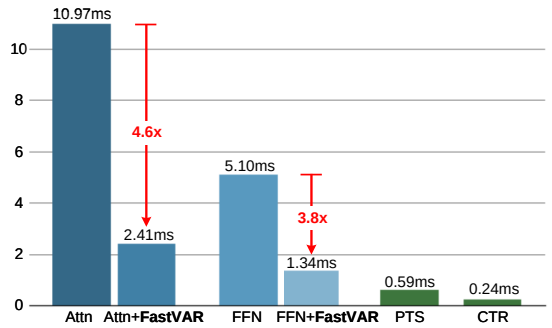


Figure A.1. Efficiency Profiling of different modules in one Transformer layer. Note that the runtime of the Attn and FFN baseline is evaluated using FlashAttention.

Table A.2. More evaluation results on HPSv2.1 [58] and ImageReward [62] benchmarks.

benchmark	HART	ToMe	Ours	Infinity	ToMe	Ours
Latency↓	950ms	800ms	630ms	2600ms	1130ms	950ms
Speedup↑	1.0×	1.2×	1.5×	1.0×	2.3×	2.7×
HPSv2.1↑	28.75	27.04	27.85	30.36	29.85	30.03
ImageReward↑	0.5658	0.4988	0.5370	0.9245	0.8992	0.9129

C. Comparison on More Benchmarks

In the main paper, we compare our FastVAR with different methods on the Geneval [20] and MJHQ30K [29] datasets. In order to provide a systematic evaluation, we further compare different methods on more benchmarks including HPSv2.1 [58], and ImageReward [62]. The experimental results are given in Tab. A.2. It can be seen that our FastVAR maintains consistently favorable performance than other competitive token pruning baseline, while allowing for significant speedup than the original backbones. For instance, FastVAR achieves 0.81 higher HPSv2.1 score than ToMe [3] while being more efficient. The above results on more evaluation benchmarks further demonstrate the effectiveness of our FastVAR.

Table A.3. Ablation experiments of applying extreme pruning ratios to other VAR backbone HART [49]. We set $N = 2$ in all setups, *i.e.*, only the last two scale steps are pruned with FastVAR.

ratios	Speedup↑	Latency↓	Throughput↑	FID↓	CLIP↑	GenEval↑
no_pruning	1.0×	0.95s	1.05	30.61	28.47	0.51
{50%,75%}	1.5×	0.63s	1.59	28.19	28.34	0.51
{50%,100%}	1.9×	0.51s	1.96	48.54	28.46	0.48

Table A.4. Ablation experiments of different caching scale steps.

cached steps	GenEval				MJHQ30K	
	two_object↑	position↑	color_attr↑	Overall↑	FID↓	CLIP↑
no_pruning	0.62	0.13	0.18	0.51	30.61	28.47
K-N-3	0.53	0.11	0.15	0.47	40.61	27.36
K-N-2	0.60	0.13	0.19	0.50	32.39	27.94
K-N-1	0.57	0.13	0.20	0.49	29.83	28.25
K-N	0.57	0.16	0.24	0.51	28.19	28.34

D. Ablation on Caching Step

In the proposed Cached Token Restoration (CTR), we use the token map from the last scale step of the Structure Construction Stage $\mathcal{S}$, *i.e.*, the $(N-K)$-th step, as the cache, which will be used to restore the original token numbers during token pruning. Here, we conduct ablation experiments to justify the rationality by setting different scale steps as the caching step. The results are shown in Tab. A.4. It can be seen that setting the last element in $\mathcal{S}$ as the caching step achieves consistently the best results on all evaluation metrics. Notably, this setup has almost no performance degradation compared to the unpruned baseline models. In addition, we observe a steady performance drop when the caching step gradually moves small. This is because we use the cached token map to approximate the pruned tokens, so the gap between the last element in $\mathcal{S}$ and the steps in $\mathcal{T}$ is the smallest. Therefore, using the step that is closer to the pruned scale steps as the caching step can achieve better performance.

E. Discussion on Extreme Pruning Ratios

In the main paper, we mentioned that different backbones exhibit different levels of tolerance for the pruning ratio. For example, we used an even 100% ratio for the last two scale steps of the Infinite model [22]. However, we point out that this extreme pruning ratio does not apply to HART model [49]. Specifically, we apply the $N = 2$ and $\{50\%, 100\%\}$ FastVAR to the HART model. The experimental results are shown in Tab. A.3. It can be seen that the extreme pruning ratio produces serious performance degradation for HART. We argue that this is due to the difference in the pre-trained model size. Specifically, the size of Infinite 2B is significantly larger than the 700M of HART. The stronger capabilities of the larger model allow for modeling more challenging textures in the earlier scale steps. As a contrast, the smaller model relies on test-time scaling [47] to use longer scale steps to produce complex details, and thus suffers from severe degradation when extreme pruning is applied on the last few steps.

F. Limitation and Future Work

Our FastVAR can effectively alleviate the quadratically increasing complexity with scales, benefiting from the proposed cached token pruning. Nonetheless, our work can be further improved in the future in the following aspects. First, the proposed FastVAR focuses mainly on the acceleration of the large-scale step which occupies the main inference time. Therefore, our method can be further improved in accelerating small-resolution image generation tasks by designing more generalized pruning strategies to include pruning small-scale token maps as well. Second, we have revealed the scale-wise sensitivity of pre-trained VAR models, *i.e.*, large-scale steps are more robust to small-scale steps for pruning, which inspires us to adopt a progressive pruning ratio schedule. Therefore, utilizing more fine-grained pruning prior, *e.g.*, layer-wise or even developing adaptive pruning ratios, is promising to achieve higher speedup ratios. Third, we show that the current FastVAR can be combined with Flash Attention to achieve combined speedup. As other potential work on accelerating VAR models emerges, such as network quantization or fewer decoding steps, our FastVAR could potentially integrate with these approaches to achieve further acceleration.

G. Algorithm of FastVAR

In Algo. 1, we give the Pytorch-like pseudocode of the proposed FastVAR. Thanks to the simplicity and generality, our proposed FastVAR can be seamlessly integrated into various VAR models using a few code lines.

H. More Visual Results

In this section, we provide more visual results, which are organized as follows:

- In Fig. A.2, we give more visualization results about the intermediate outputs of the pre-trained VAR model at different scale steps.
- In Fig. A.3, we give more qualitative results of Infinite [22] on the MJHQ30K dataset.
- Fig. A.4 gives more qualitative results of the HART [49] model on the MJHQ30K [29] dataset.
- In Fig. A.5, Fig. A.6, and Fig. A.7, we give more generation results on the zero-shot higher-resolution image synthesis tasks.

Algorithm 1: The pseudo-code of FastVAR algorithm, Pytorch-like

```
def pivotal_token_selection(x, topk):
    # calculate the direct-through component
    pool_x = rearrange(x, 'b (h w) c -> b c h w')
    pool_x = adaptive_avg_pool2d(x, (1, 1))
    pool_x = rearrange(pool_x, 'b c 1 1 -> b 1 c')
    score = sum((x - pool_x)**2, dim=-1)
    # select the topK high frequency tokens
    pivotal_idx = argsort(score, dim=1, descending=True)[:, :topk, :]
    return gather(x, dim=1, index=pivotal_idx)
def cached_token_restoration(x, cache):
    # up-sample cache features to the size of x
    restored_x = interpolate(cache)
    restored_x = rearrange(restored_x, 'b c h w -> b h w c')
    # fuse the cached and the current tokens
    restored_x.scatter_(dim=1, index=pivotal_idx, src=x)
    return restored_x
```

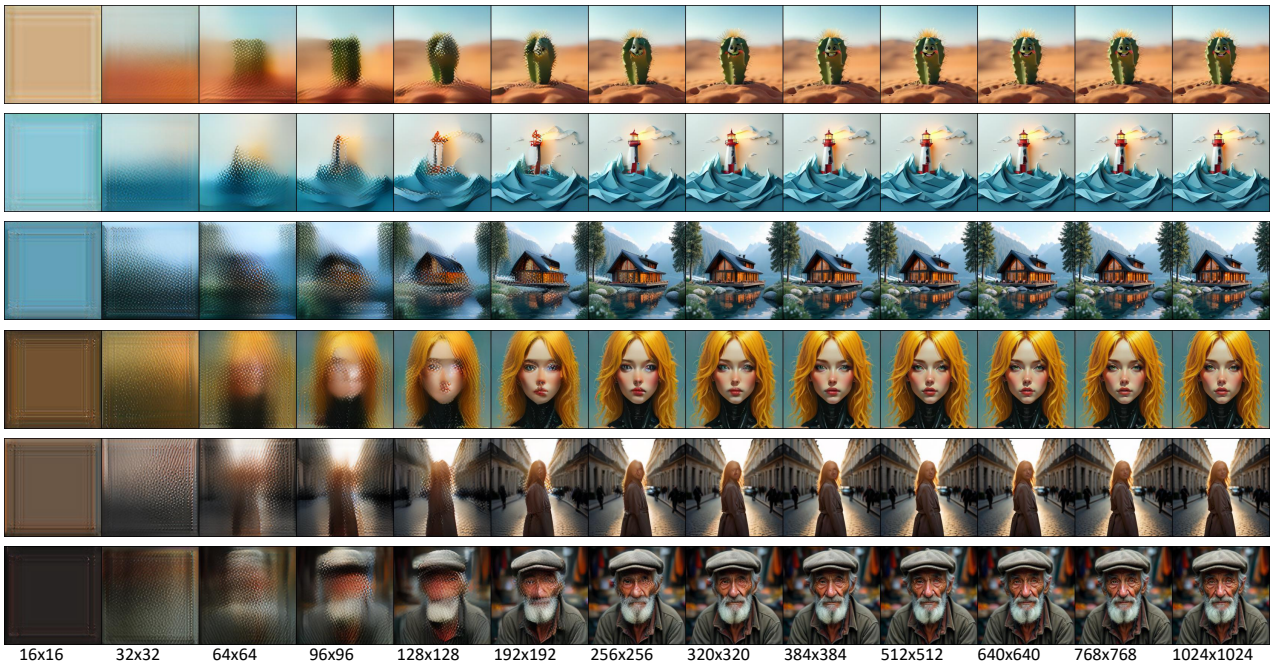


Figure A.2. More visualization results of the intermediate outputs at different scale steps of pre-trained VAR model [22].

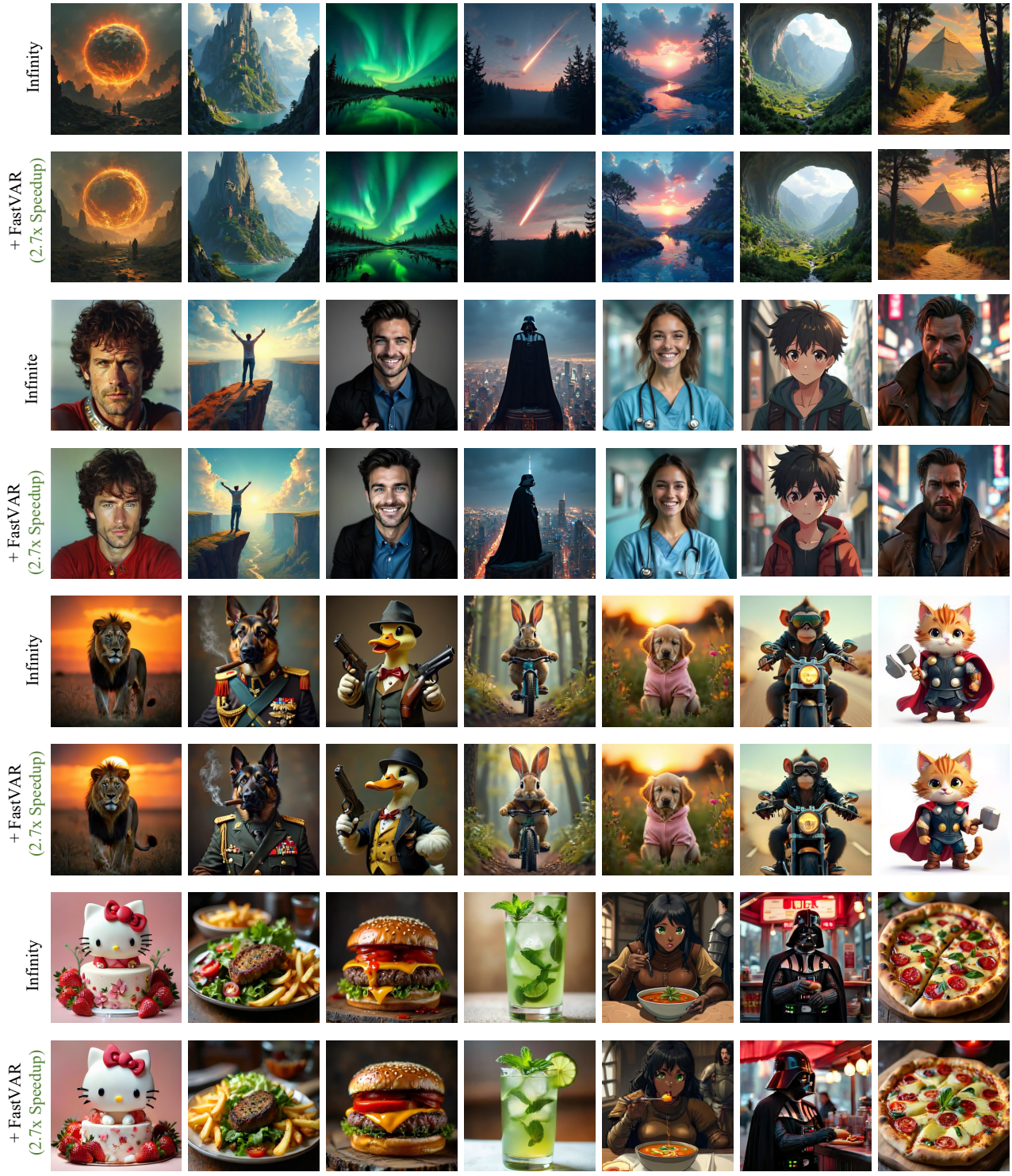


Figure A.3. More qualitative comparison with the Infinite model on the MJHQ30K dataset.

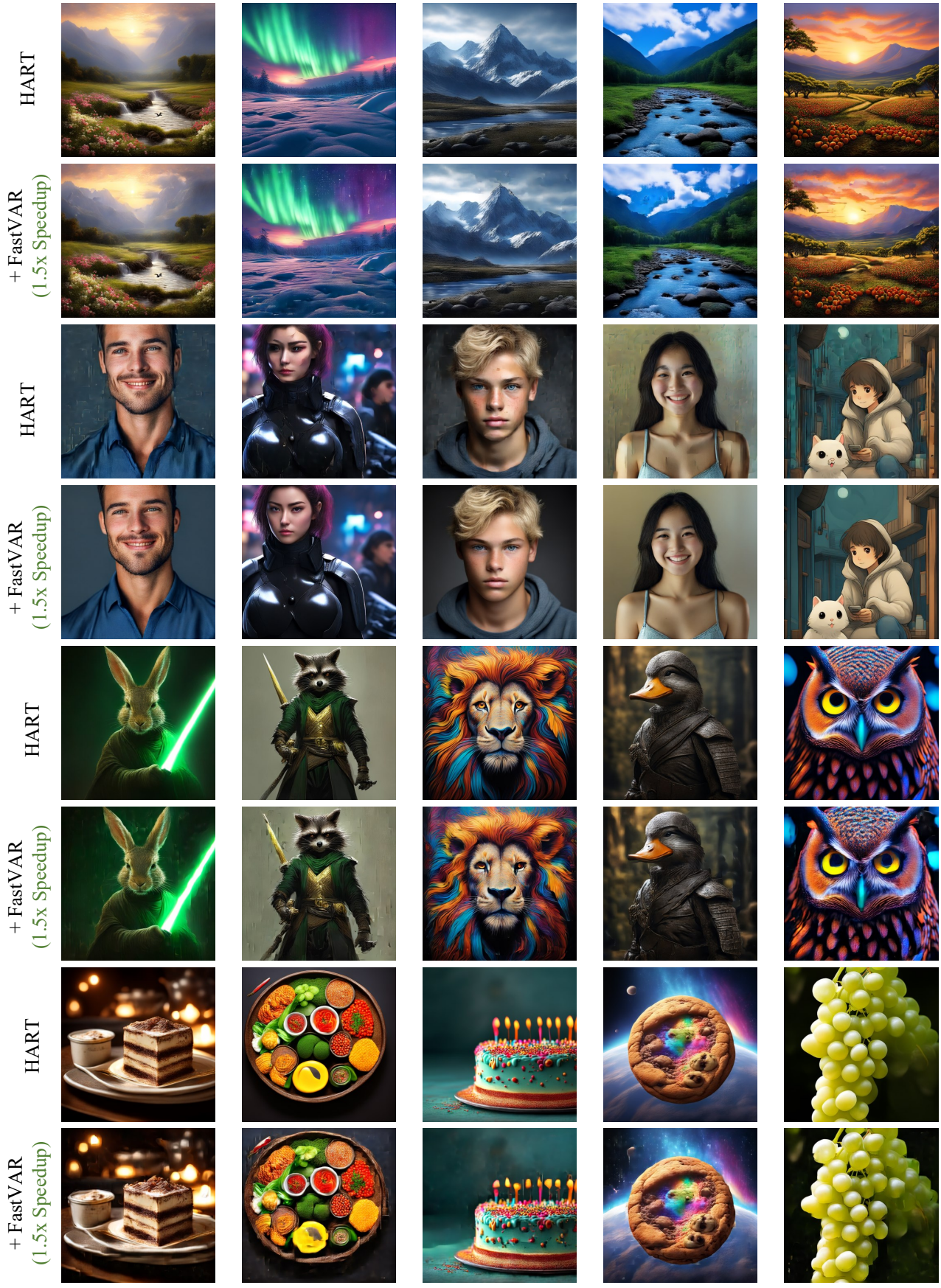


Figure A.4. More qualitative comparison with the HART backbone on the MJHQ30K dataset.

Figure A.5. Moreover generation results of the high-resolution image synthesis with Infinite+FastVAR. The images are scaled for better presentation.

Figure A.6. Moreover generation results of the high-resolution image synthesis with Infinite+FastVAR.

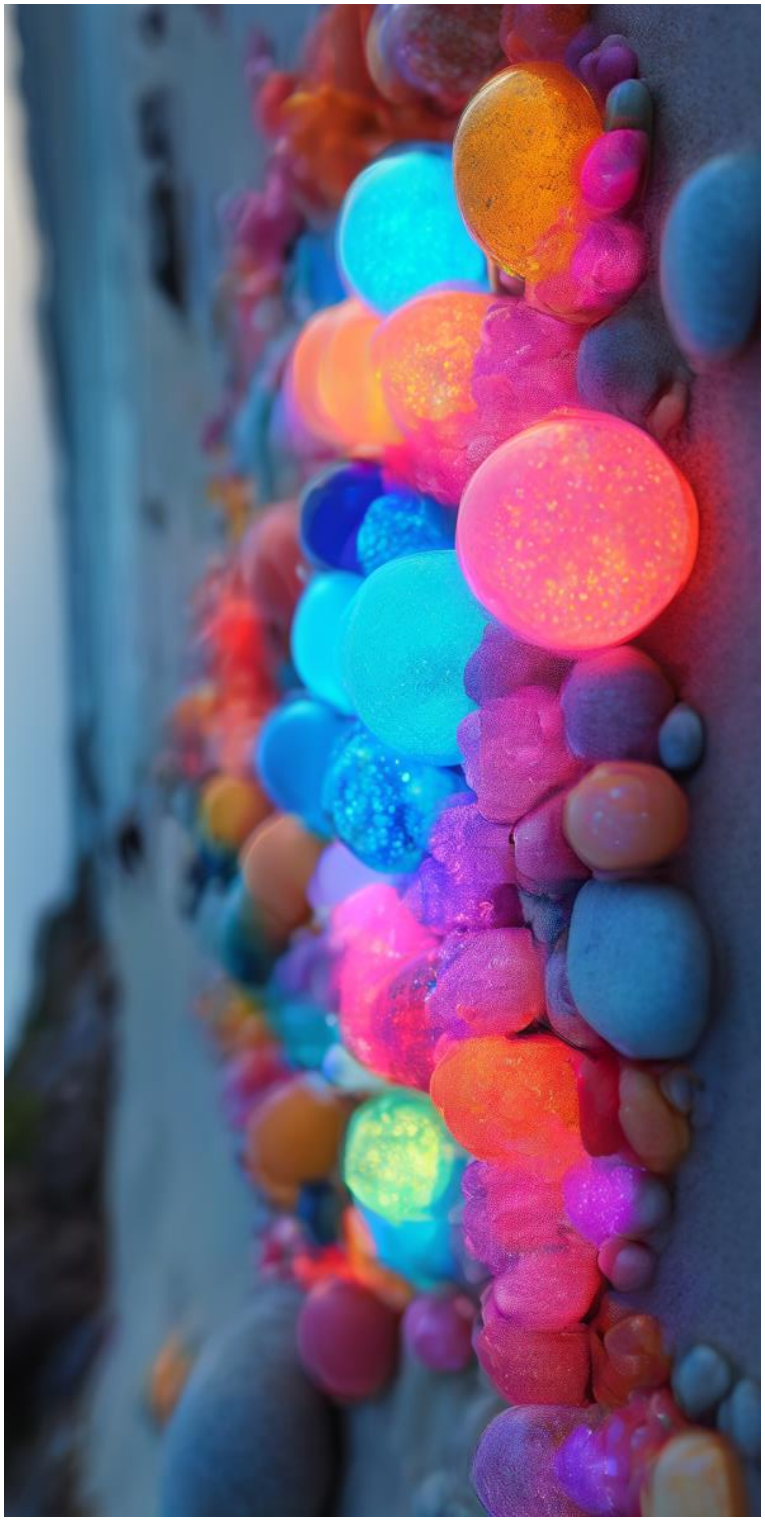


Figure A.7. Moreover generation results of the high-resolution image synthesis with Infinite+FastVAR.