PruneVAR: Training-Free Acceleration of Visual Autoregressive Modeling via Two-Stage Redundancy Reduction

Jiajian Xie*
Zhejiang University
Hangzhou, China
xiejiajian@zju.edu.cn

Jie Yang*
Zhejiang University
Hangzhou, China
j7738157@gmail.com

Shengyu Zhang†
Zhejiang University
Hangzhou, China
sy_zhang@zju.edu.cn

Figure 1: Our PruneVAR achieves efficient acceleration while preserving high-frequency details without reducing token inputs, whereas existing token pruning methods introduce noticeable artifacts and texture degradation.

Abstract

Visual Autoregressive (VAR) modeling has attracted attention for its coarse-to-fine generation via next-scale prediction. However, higher scale resolutions lead to rapid token growth and substantially increased latency. Existing methods accelerate the late high-resolution stages by reusing converged tokens, but may discard high-frequency details in unupdated regions. To address this challenge, we propose PruneVAR, a training-free VAR acceleration framework that reduces redundant model computations without decreasing token inputs to preserve fine-grained texture coherence. Our approach is motivated by the finding that the redundancy in VAR transitions from stage-level stability to layer-wise functional overlap across the multi-scale generation process. Based on this insight, we introduce a corse-to-fine layer pruning strategy. Specifically, we first prune all layers at specific stages by skipping model forward while restoring pruned outputs using cached image increments from previous scale steps. As generation proceeds to higher resolutions, we pre-identify redundant layers by analyzing the overlap of core update regions and skip only these layers in the late stages. Experiments show that PruneVAR achieves up to 2× speedup while maintaining competitive visual quality. More visualization results are available at https://jiajianxie.github.io/prunevar/.

CCS Concepts

• **Computing methodologies** → **Computer vision**.

Keywords

visual autoregressive modeling, training-free acceleration

ACM Reference Format:
Jiajian Xie, Jie Yang, and Shengyu Zhang. 2026. PruneVAR: Training-Free Acceleration of Visual Autoregressive Modeling via Two-Stage Redundancy Reduction. In *Proceedings of the 34th ACM International Conference on Multimedia(MM '26), November 10–November 14, 2026, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3767308.3835902

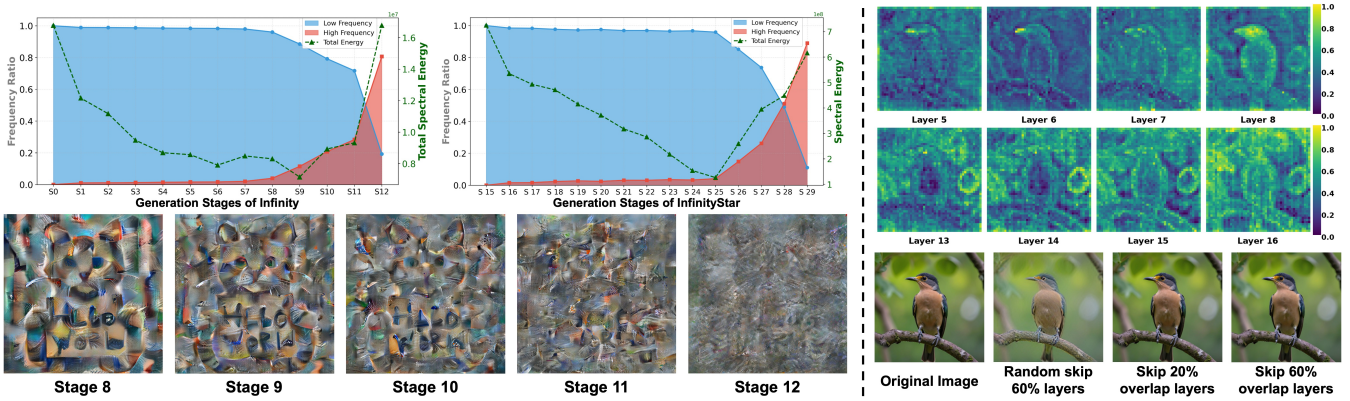
1 Introduction

Visual Autoregressive (VAR) modeling [23] introduces a next-scale autoregressive paradigm for image generation. In stead of conventional next-token prediction [13, 21, 26], VAR represents an image as a sequence of token maps at multiple scales. During generation, the model first predicts a low-resolution map that captures global

*Equal contribution.
†Corresponding author.

MM '26, Rio de Janeiro, Brazil

ACM ISBN 978-1-4503-XXXX-X/26/11
https://doi.org/10.1145/3767308.3835902



(a) Stage-level stability.

(b) Layer-wise functional overlap.

Figure 2: Visualization of the observed redundancy patterns in visual autoregressive generation. (a) Upper panels present spectral energy and frequency components of cross-scale increments, while lower panels show relatively stable cross-scale increments at intermediate stages. (b) Upper panels present similar modification regions across selected layers, while lower panels compare random layer pruning with overlap-based layer pruning.

semantics, and then progressively refines it. This method significantly reduces the number of autoregressive steps. However, in the later high-resolution stages, the quadratic growth in token numbers leads to substantial computation and long inference time [5], which remains a major bottleneck for real-time generation.

To accelerate VAR inference, existing studies mainly adopt token pruning strategies [2, 5, 33], which aim to identify and skip redundant token computations. Specifically, these methods detect stable regions and reuse tokens in these regions from previous stages or neighborhoods to avoid repeated computation. Although effective in improving efficiency, these approaches introduce limitations that compromise generation quality. They fail to sufficiently model stable regions (typically background textures or smooth surfaces), which often results in blurring and visible seams that degrade visual coherence, as shown in Figure 1. Moreover, once tokens are incorrectly classified as stable, the errors accumulate in subsequent steps and ultimately degrade the output quality.

To address these limitations, we propose a different acceleration paradigm that retains full token participation while skipping redundant computations. Our empirical analysis of VAR inference reveals that the redundancy transitions from global structural stability to localized layer functional overlap across stages. **1) Cross-scale increments exhibit structural stability at intermediate stages.** Frequency analysis in Figure 2a (Top) shows low energy and steady growth high-frequency components during stages 7–10. We further visualize the increments in Figure 2a (Bottom) and find that structural patterns can be observed at stages 8–9. These results indicate a stage-level redundancy at intermediate stages. **2) Certain layers at high-resolution stages have limited impact on the final output.** As shown in Figure 2(a), consecutive layers often concentrate on highly overlapping spatial regions, indicating repeated local refinement. An experiment of manually skipping certain layers at the last two stages still preserves high image fidelity, as shown in Figure 2(b). This indicates that redundancy shifts to a layer-wise granularity in the last stages, where certain layers have limited contributions to the final output.

Based on the above observations, we propose PruneVAR, a two-stage acceleration framework for VAR models that reduces redundant computation without decreasing token inputs. In the structure-stabilization stage, we introduce a **cache-then-prune** strategy that leverages cached image increments from preceding scales to approximate subsequent updates, allowing approximately half of the forward passes in this stage to be completely pruned, while periodically incorporating newly computed increments to refine the approximation. In the quality refinement stage, we introduce a **cross-scales layer pruning** strategy tailored for high-resolution scales. We first quantify the per-layer modification regions by measuring feature changes between inputs and outputs, emphasizing tokens with the most significant updates. Layers with highly overlapping modification regions across consecutive layers are then pre-selected for pruning. Leveraging the consistency of layer behaviors across adjacent high-resolution scales, redundant layers are determined once before this stage and reused throughout subsequent inference. Overall, our contributions are as follows:

- We propose a cache-then-prune strategy for structure-stabilization stages, which leverages cached image increments to skip intermittent model forward passes while periodically refining with newly computed increments.
- We introduce a cross-scales layer pruning strategy for high-resolution stages, identifying layers with overlapping high-impact modification regions and reusing the pruning configuration across adjacent scales to reduce redundant computation.
- Extensive experiments show that PruneVAR achieves up to 2× speedup while maintaining competitive global image quality.

2 Related Work

2.1 Autoregressive Visual Generation

Autoregressive models (AR) [9, 19, 24, 31] typically treat image generation as a sequence modeling task, where pixels or visual tokens are flattened into a sequence. These models predict the next token

conditioned on the previously generated tokens. However, the inference speed is limited by the strictly sequential generation [27]. Different from traditional next-token prediction, Visual Autoregressive (VAR) modeling [23] introduces a next-scale prediction paradigm. VAR reformulates image generation as a hierarchical, multi-scale autoregressive process, generating low-resolution token maps in the initial stages and progressively refining them into higher resolutions. Building upon this, several work has proposed high-quality text-to-image generation. HART [22] improves generation quality and efficiency by decomposing the continuous latent variables of an autoencoder into discrete tokens representing the global structure and continuous residual tokens capturing fine-grained details. Infinity [6] moves beyond fixed-vocabulary constraints by employing bit-wise visual tokenization, which theoretically enables an infinite vocabulary size for synthesizing high-fidelity images. Although VAR significantly reduces the number of autoregressive steps compared to standard AR models, the computational overhead in the later high-resolution stages remains a critical bottleneck for real-time inference [5, 10].

2.2 Visual Autoregressive Acceleration

Existing efforts to improve the inference efficiency of the VAR models include both training-based [1, 3, 10, 14, 32] and training-free [12, 15, 29] approaches. Training-based methods reduces computational redundancy by modifying model architectures [1, 25] or introducing specialized training strategies [14]. However, they incur substantial training costs and lack cross-model adaptability. In contrast, training-free methods have become a research focus, due to plug-and-play deployment without. These training-free techniques generally follow two directions. KV cache compression methods [11, 17, 30] reduce memory and computation by allocating cache budgets based on the sensitivity of different scales and layers. Token pruning methods [2, 5, 33] focus on dynamically identifying and skipping redundant tokens during high-resolution stages, and reusing neighboring or previous tokens. Despite improving efficiency, token pruning–based approaches still face limitations. Reuse of tokens often leads to blurred or missing fine-grained details and introduces visible seams at the boundaries between updated and reused regions. Moreover, once tokens are incorrectly classified as stable, the error cannot be corrected in subsequent steps. Such errors accumulate over stages and degrade the quality of the final output. In this paper, we focus on training-free acceleration without reducing tokens, and instead improve efficiency by reducing redundant computations within the model.

3 Method

3.1 Preliminary

The Visual Autoregressive (VAR) modeling [23] reformulates autoregressive generation by shifting from a "next-token prediction" to a "next-scale prediction". Instead of predicting individual tokens sequentially, VAR treats each autoregressive unit as a token map at a specific resolution. For a given image feature map, VAR first quantizes it into K multi-scale token maps $\mathcal{R} = \{r_1, r_2, \ldots, r_K\}$, where each token map r_k is associated with a predefined resolution (h_k, w_k) that increases progressively with k. The joint distribution over these token maps is factorized autoregressively as:

$$p(r_1, r_2, \ldots, r_K) = \prod_{k=1}^{K} p(r_k \mid r_1, \ldots, r_{k-1}). \tag{1}$$

To improve training stability, existing methods [6, 16, 22] usually adopt a residual prediction mechanism. At each scale k, the model predicts a residual term f_k based on previously generated token maps. The current prediction is obtained by adding this residual to the upsampled result from the previous scale:

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

where the "interpolate$(\cdot, (h_k, w_k))$" denotes upsample a token map to the size of (h_k, w_k). By unrolling the recursive formulation, the prediction at scale k can be expressed as the accumulation of all previous residuals:

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

During inference, generation proceeds sequentially from coarse to fine scales. Although KV cache can reuse intermediate computations, the computational cost still increases significantly as the resolution grows.

3.2 Motivation

In this work, we aim to improve the efficiency of VAR inference by reducing redundant model computations. To this end, we investigate the stage-wise and layer-wise behaviors of a pre-trained VAR model [6]. Specifically, we analyze the evolution of image increments across stages and examine the functional roles of layers at different resolutions. Conclusions are summarized as follows.

Observation 1: Cross-scale increments are structural stable during the intermediate stages. As shown in Figure 2a (Top), the spectral energy of cross-scale increments decreases rapidly after the early stages, remains at low level during the intermediate stages (e.g. stages 6–10), and rises again in the final stages. Meanwhile, the high-frequency components of the increments increase consistently during stages 7-11, showing a notably stable trend. We also find that the global structure remains recognizable at stage 8 and stage 9, but becomes difficult to discern at the last two stages, as shown in Figure 2a (Bottom). These results suggest that VAR generation transitions through distinct redundancy phases. In the intermediate stages, the foundational global structure has been established, and the model enters a state of structural stability where cross-scale increments become stable and predictable. This high degree of consistency implies a maximal redundancy at the stage level, providing a foundation for 100% layer pruning (i.e., stage-level skipping) through increment reuse without compromising the established layout. However, as the generation proceeds to high-resolution stages, the token granularity becomes too fine to support coarse-grained stage-level reuse.

Observation 2: Certain layers at high-resolution stages have limited contributions on the final output. Since the scale-wise stability does not exists in the last stages, we instead analyze redundancy from a layer-wise perspective. As shown in Figure 2(a), we find that many consecutive layers repeatedly focus on highly overlapping regions for local refinement, suggesting that their functions

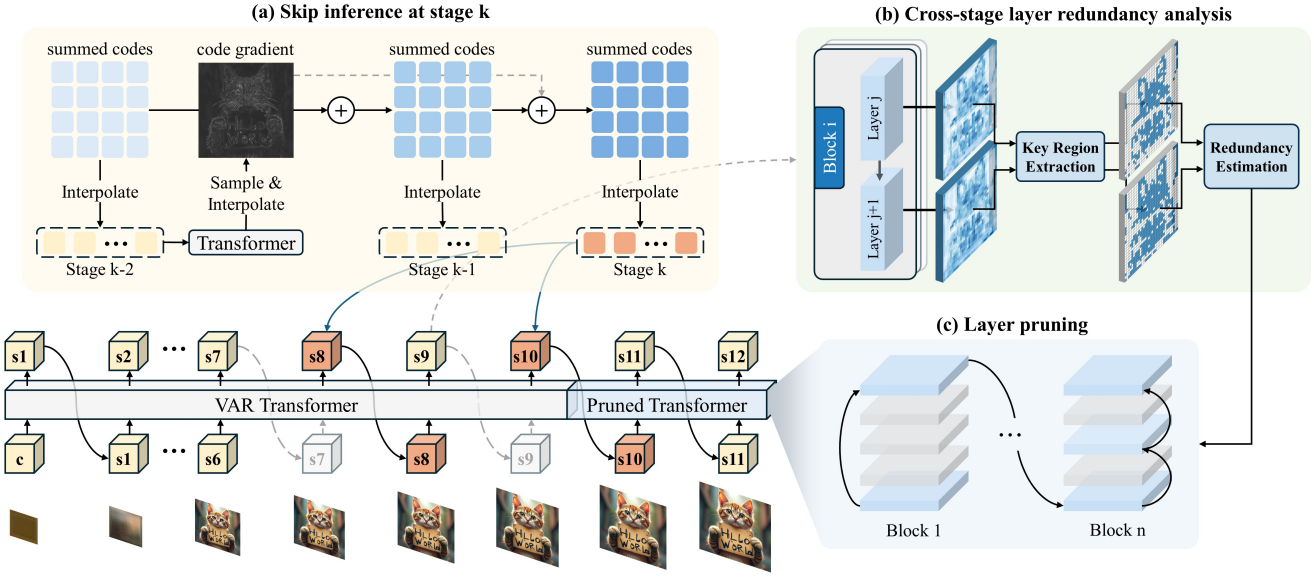


Figure 3: Overview of the proposed PruneVAR method. Subfigure (a) illustrates the cache-then-prune strategy in the structure-stabilization stage ($\mathcal{SS}$), where model inference alternates with reuse of cached outputs. Subfigure (b) shows the process of identifying redundant layers by analyzing overlapping modification regions across adjacent layers. Subfigure (c) demonstrates the application of the identified redundant layers to perform layer pruning during the quality refinement stage ($\mathcal{QR}$).

are redundant. We further manually select a subset of layers with highly overlapping focus regions and skip them during inference. This simple strategy produces results that remain highly consistent with the original outputs, as shown in Figure 2(b). These results indicate that the skipped layers contribute marginally to the final visual quality. In the late stages, the model shifts to fine-grained local refinement, where redundancy appears among certain groups of layers that repeatedly refine similar regions. As a result, layer redundancy decreases from a global pattern to a more localized one, meaning that part of the layers can be pruned without reducing generation quality.

3.3 Two-Stage Redundancy Reduction

Building upon the above observations, we propose PruneVAR to reduce redundant computations in VAR models. As shown in Figure 3, denote the semantic-planning stage as $\mathcal{SP} = \{1, 2, \ldots, N\}$, the structure-stabilization stage as $\mathcal{SS} = \{N+1, \ldots, M\}$, and the quality-refinement stage as $\mathcal{QR} = \{M+1, \ldots, K\}$. Due to the variability in the semantic planning stage, which determines the global structure, we retain the standard VAR process in $\mathcal{SP}$. For the set $\mathcal{SS}$, we intermittently skip model forward and restore the skipped outputs using cached image increments from previous stages. For the set $\mathcal{QR}$, we apply layer pruning by pre-identifying redundant layers and performing inference with the pruned model. More details are given below.

Cache-Then-Prune Strategy. In the structure-stabilization stage ($\mathcal{SS}$), model predictions at each scale step produce stable image increments. Leveraging this observation, we propose a cache-then-prune strategy that alternates between full model inference and reuse of cached outputs. As shown in Figure 3(a), we cache the model output f_k at scale step k. In the following step, instead of performing full model inference, the cached output f_k is upsampled to the next scale (h_{k+1}, w_{k+1}) and used as a proxy for f_{k+1}. This process repeats across $\mathcal{SS}$, alternately caching newly computed outputs and reusing cached increments for skipped steps:

$$f_{k+1} = \begin{cases} \text{interpolate}(f_k, (h_{k+1}, w_{k+1})), & \text{if step } k \text{ is computed,} \\ \text{modelforward}(\tilde{r}_{k+1}), & \text{if step } k \text{ is skipped.} \end{cases} \tag{4}$$

Assuming the structure-stabilization stage spans from step $N+1$ to M, the proposed cache-then-prune strategy reduces the number of model forward passes from $(M-N)$ to approximately $\frac{M-N}{2}$ under an alternating scheme.

Redundant Layer Identification. In the quality refinement stage ($\mathcal{QR}$), we further accelerate inference by pruning redundant layers. As shown in Figure 3(b), the key idea is to identify layers with highly overlapping modification regions for pruning. To quantify these regions, we measure the magnitude of feature updates between a layer's input and output. Given the input and output of the l-th layer at scale step k, denoted as $x_k^{(l)} \in \mathbb{R}^{B\times T\times C}$ and $y_k^{(l)} \in \mathbb{R}^{B\times T\times C}$, we compute the token-wise update magnitude as:

$$\Delta_k^{(l)} = \sum_{c=1}^{C} \left| y_k^{(l)} - x_k^{(l)} \right|, \tag{5}$$

where B is the batch size, T is the number of tokens, C is the feature dimension and $\Delta_k^{(l)} \in \mathbb{R}^{B\times T}$ represents the aggregated feature changes for each token. To normalize the update magnitude, we divide $\Delta_k^{(l)}$ by its maximum value:

$$\hat{\Delta}_k^{(l)} = \frac{\Delta_k^{(l)}}{\max(\Delta_k^{(l)})}. \tag{6}$$

Table 1: Quantitative results of T2I and T2V tasks. The best and second-best results are in bold and underlined.

T2I Model	Quality						Efficiency	
	CLIP↑	SSIM↑	GenEval↑	T2I-CompBench			Latency(s)↓	SpeedUp↑
				B-VQA↑	UniDet↑	S-CoT↑		
Infinity	0.32108	-	0.68122	0.61729	0.40087	77.661	1.72771	-
+FastVAR	0.32149	0.71698	<u>0.67562</u>	**0.61867**	<u>0.39786</u>	77.575	0.88176	<u>1.959×</u>
+SparseVAR	<u>0.32168</u>	<u>0.72471</u>	0.67065	0.61589	0.39779	<u>77.607</u>	1.33477	1.294×
+PruneVAR	**0.32171**	**0.86966**	**0.67686**	<u>0.61856</u>	**0.39992**	**77.611**	**0.87609**	**1.972×**
HART	0.31364	-	0.51528	0.50823	0.37567	72.4526	0.90295	-
+FastVAR	0.31103	0.57599	**0.51602**	0.50611	<u>0.36598</u>	71.9802	<u>0.65131</u>	<u>1.386×</u>
+SparseVAR	<u>0.31305</u>	<u>0.64477</u>	0.51541	<u>0.50735</u>	0.36332	<u>72.0388</u>	0.68201	1.323×
+PruneVAR	**0.31348**	**0.84231**	<u>0.51553</u>	**0.50796**	**0.36921**	**72.2612**	**0.65097**	**1.387×**
T2V Model	Quality						Efficiency	
	CLIP↑	SSIM↑	VBench↑	T2V-CompBench			Latency(s)↓	SpeedUp↑
				MLLM↑	Detection↑	Tracking↑		
InfinityStar	0.30198	-	0.83739	0.42304	0.41084	0.24987	95.77793	-
+PruneVAR	0.30105	0.84348	0.83082	0.41552	0.40856	0.24483	59.03892	1.62×

Based on the normalized update map, we define the core update region mask for layer l as:

$$M_k^{(l)} = \mathbb{I}\left(\hat{\Delta}_k^{(l)} > \tau\right), \tag{7}$$

where τ is a predefined threshold and $\mathbb{I}(\cdot)$ is the indicator function. The mask indicates tokens with relatively large feature updates, representing the regions modified by the layer.

Given the masks of adjacent layers, we then measure their overlap to quantify similarity:

$$\mathrm{Sim}_k^{(l,l+1)} = \frac{\left|M_k^{(l)} \cap M_k^{(l+1)}\right|}{\left|M_k^{(l)} \cup M_k^{(l+1)}\right|}. \tag{8}$$

If the overlap exceeds a threshold γ, the two layers are considered to have highly overlapping modification regions. We then mark the latter layer as redundant for pruning:

$$p_k^{(l+1)} = \begin{cases} 1, & \text{if } \mathrm{Sim}_k^{(l,l+1)} > \gamma, \\ 0, & \text{otherwise}, \end{cases} \tag{9}$$

where $p_k^{(l+1)} = 1$ indicates that the $(l+1)$-th layer at scale step k is redundant and can be pruned.

Cross-Stage Layer Pruning. We observe that at high resolutions, the functional behaviors of layers remain consistent across adjacent stages. Therefore, we perform redundancy analysis at the final step of the structure-stabilization stage, denoted as k_{last}, and record the layers to be pruned using:

$$r_{\text{QR}}^{(l)} = r_{k_{\text{last}}}^{(l)}, \tag{10}$$

where $r_{\text{QR}}^{(l)} = 1$ indicates that the l-th layer is marked for pruning. During the subsequent stages in $\mathcal{QR}$, we reuse this pruning configuration and perform inference using the pruned model, skipping layers with $r_{\text{QR}}^{(l)} = 1$ during forward passes, as shown in Figure 3(c). To preserve generation quality, we do not prune the initial blocks and the first and last layers within each block, as these are critical for maintaining global structure and stable feature transformations.

4 Experiments

4.1 Experimental Setup

Models and Evaluations. We apply PruneVAR to two text-to-image VAR models, Infinity [6] and HART [22] (1024 × 1024 resolution), the text-to-video model InfinityStar [16] (720P, 81 frames). For comparison, we use FastVAR [5] and SparseVAR [2] on the text-to-image models. Since these baselines do not provide acceleration code for InfinityStar, we do not include them in the video experiments. For evaluation metrics, we compare both in terms of generation quality and inference efficiency. For generation quality, text-to-image results are evaluated on GenEval [4] and T2I-CompBench [7], while text-to-video results are assessed on VBench [8] and T2V-CompBench [20]. We also evaluate all generation results with CLIP [18] scores for textual alignment and SSIM [28] for structural consistency. For efficiency evaluation, we consider both inference speed and computational cost. Inference speed is measured by latency and speedup, while computational cost is evaluated using TFLOPs, total KV-cache tokens, KV-cache memory, and peak GPU memory.

Implementation Details. For Infinity, $\mathcal{SS}$ spans stages 7 to 10, while for InfinityStar, it spans stages 24 to 27. For HART, no clear $\mathcal{SS}$ is observed, and thus the cache-then-prune strategy is not applied. For threshold selection, we perform a grid search on Infinity using 10 prompts. Each candidate setting is evaluated by $0.5 \times \text{SSIM} + (1 - \text{Latency}/\text{Latency}_{\text{orig}})$ to jointly account for structural consistency and inference efficiency. We then adopt $\tau = 0.5$ and $\gamma = 0.8$ as the default settings across all models. For baselines, FastVAR follows its default settings, while SparseVAR uses a varying threshold of 0.6 with other settings unchanged. Following FastVAR, latency is measured without including the VAE decoding time, as it is shared across all methods. All experiments are conducted on a single NVIDIA A100.

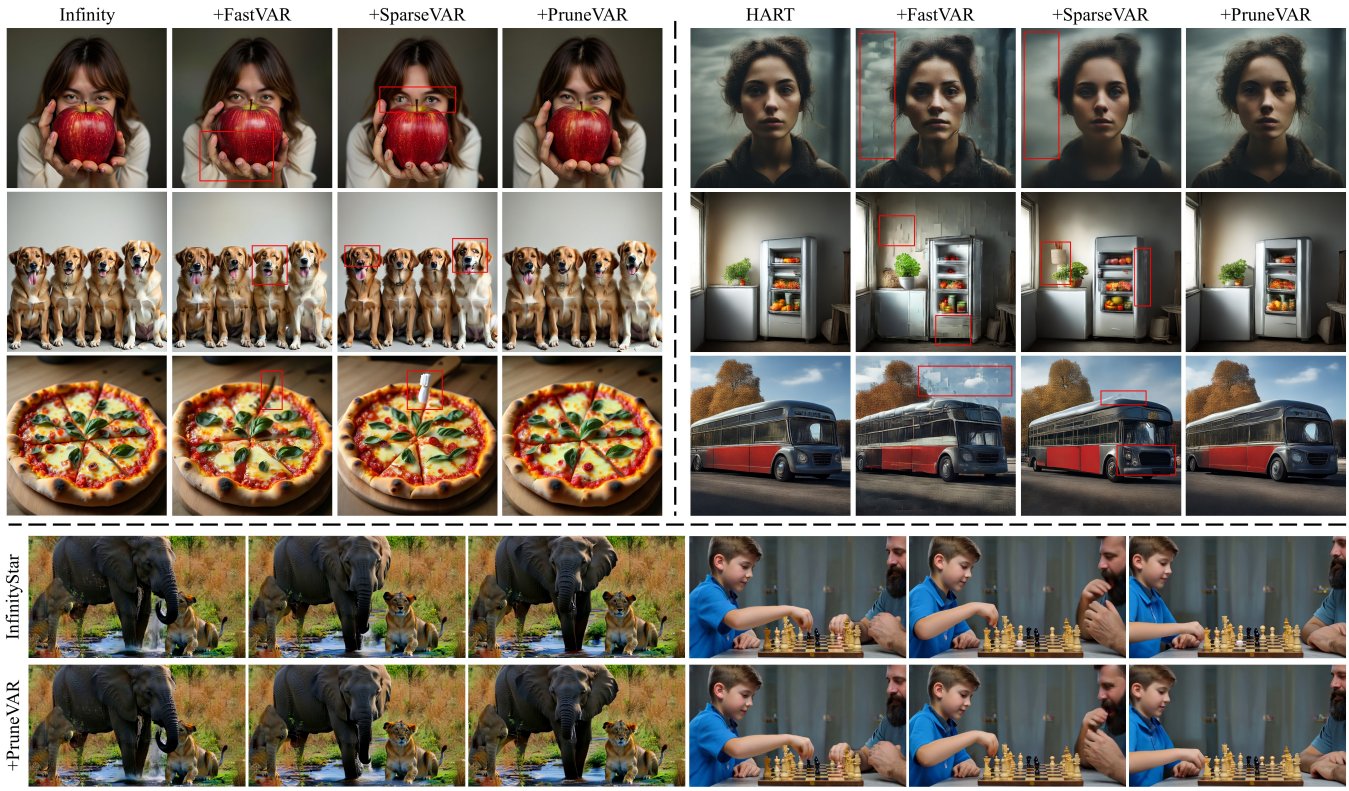


Figure 4: Qualitative results of T2I and T2V generation task using different models.

4.2 Quantitative Results

As shown in Table 1, PruneVAR achieves a better quality-efficiency trade-off. On text-to-image generation, PruneVAR obtains the highest speedup among the compared methods, reaching 1.97× on Infinity and 1.39× on HART. Moreover, PruneVAR also preserves better generation quality, achieving higher SSIM scores than FastVAR and SparseVAR on both Infinity (0.86966 vs. 0.71698 and 0.72471) and HART (0.84231 vs. 0.57599 and 0.64477), while maintaining competitive GenEval scores. On text-to-video generation, PruneVAR achieves a 1.62× speedup on InfinityStar with a VBench score close to the original model (0.83082 vs. 0.83739), suggesting that the proposed redundancy reduction strategy generalizes consistently across image and video generation.

Table 2 further shows that PruneVAR effectively reduces computational and memory costs. On Infinity, PruneVAR reduces TFLOPs from 98.05 to 23.64, slightly lower than FastVAR (23.66) and SparseVAR (24.26), while decreasing peak GPU memory from 15.74 GB to 12.73 GB. Although token-pruning methods achieve lower KV-cache memory by directly reducing tokens, the memory overhead of PruneVAR remains modest while it provides better structural consistency and competitive task-level quality.

4.3 Qualitative Results

To further evaluate the visual fidelity of our approach, we provide a qualitative comparison between PruneVAR and token-pruning baselines in Figure 4. As highlighted by the red boxes, token-pruning methods suffer from insufficient modeling of fine-grained details, as well as structural artifacts in background regions such as visible seams and blurring. The loss of fine-grained textures and degraded structural coherence significantly compromise the visual quality of the generated images. In contrast, PruneVAR maintains high consistency with the original model's outputs. Specifically, it preserves delicate textures and fine details while ensuring that background regions remain free of artifacts. When extended to video generation, PruneVAR maintains visual consistency with the original outputs. By employing the two-stage redundancy reduction instead of token-pruning, our framework improves inference efficiency while maintaining competitive generative quality.

Table 2: Computational cost of different methods.

Model	TFLOPs↓	KV/k↓	KV/GB↓	Peak/GB.↓
Infinity	98.05	336.67	7.89	15.74
+FastVAR	23.66	101.64	2.38	11.61
+SparseVAR	24.26	111.42	2.61	11.19
+PruneVAR	23.64	203.04	4.75	12.73
HART	42.62	234.05	2.74	17.05
+FastVAR	21.39	132.62	1.55	13.95
+SparseVAR	21.56	134.11	1.57	14.04
+PruneVAR	21.32	185.21	1.91	14.74
InfinityStar	1754.01	2722.64	21.27	58.28
+PruneVAR	1095.21	1694.48	13.24	48.47

Table 3: Ablation of cache-then-prune and layer-pruning. "Score" denotes GenEval for T2I and VBench for T2V.

Model	Variant	SSIM↑	Score↑	Latency↓	SpeedUp↑
Infinity	PruneVAR	0.869	0.676	0.876	1.972×
	w/o layer-pruning	0.885	0.677	1.387	1.245×
	w/o cache-then-reuse	0.903	0.678	1.176	1.467×
InfinityStar	PruneVAR	0.843	0.831	59.038	1.620×
	w/o layer-pruning	0.863	0.832	84.903	1.128×
	w/o cache-then-reuse	0.891	0.836	68.319	1.408×

Table 4: Ablation experiments on stage-skipping.

Model	Skip at	SSIM↑	Score↑	Latency↓	SpeedUp↑
Infinity	[8,10]	0.870	0.677	0.876	1.97×
	[5,7]	0.687	0.670	0.953	1.81×
	[7,9]	0.775	0.680	0.927	1.86×
	[9,11]	0.818	0.666	0.853	2.02×
InfinityStar	[24-27]	0.843	0.831	59.039	1.62×
	[18-24]	0.653	0.821	67.293	1.42×
	[20-26]	0.825	0.829	60.259	1.58×
	[27-29]	0.728	0.827	56.663	1.69×

4.4 Ablation Study

Component contribution. Table 3 shows that both cache-then-reuse and layer pruning contribute to inference acceleration. When applied independently, layer pruning achieves higher speedups on both Infinity (1.467× vs. 1.245×) and InfinityStar (1.408× vs. 1.128×), indicating that redundant layers in high-resolution stages are a major source of computational overhead. Meanwhile, cache-then-reuse also provides consistent acceleration by reusing cross-scale computations. Although it leads to a slightly larger SSIM drop than layer pruning alone, the task-level scores remain nearly unchanged when the two components are combined. This suggests that the reused intermediate representations still preserve the structural cues required for effective layer selection, without interfering with the subsequent pruning process.

Impact of stage-skipping configurations in $\mathcal{SS}$. Table 4 shows that the skipped stage range affects the quality-efficiency trade-off. Skipping too early causes a clear drop in structural consistency, e.g., SSIM decreases from 0.870 to 0.687 on Infinity and from 0.843 to 0.653 on InfinityStar. Skipping later stages can provide slightly higher speedup, but it also reduces fidelity. In contrast, our selected ranges achieve near-best acceleration while preserving higher SSIM and stable task-level scores. This trend is consistent with the observation in Figure 2a: cross-scale increments are more stable in the intermediate stages. If cache-then-reuse is applied too early, the global structure is still being formed, making the cached increment an unreliable proxy for the next update. If applied too late, the generation process has shifted to fine-grained refinement, where direct reuse may skip local corrections. Therefore, the selected intermediate ranges align with the stable increment region, leading to a better quality-efficiency trade-off.

Table 5: Ablation experiments on mask threshold τ and overlap threshold γ in $\mathcal{QR}$.

Model	τ	γ	SSIM↑	Score↑	Latency↓	SpeedUp↑
Infinity	0.5	0.8	0.870	0.677	0.876	1.972×
	0.8	0.8	0.874	0.677	0.979	1.764×
	0.2	0.8	0.809	0.676	0.773	2.235×
	0.5	0.5	0.824	0.686	0.818	2.100×
HART	0.5	0.8	0.842	0.516	0.651	1.387×
	0.8	0.8	0.861	0.515	0.701	1.288×
	0.2	0.8	0.782	0.510	0.591	1.527×
	0.5	0.5	0.809	0.516	0.633	1.426×
InfinityStar	0.5	0.8	0.843	0.831	59.039	1.620×
	0.8	0.8	0.850	0.831	72.932	1.310×
	0.2	0.8	0.760	0.827	53.198	1.800×
	0.5	0.5	0.788	0.827	54.668	1.750×

Table 6: Thresholds (τ, γ) comparison. "Total score" denotes the value of the objective function in grid search.

Method	Setting	SSIM	Latency	Total-score
HART	Best $(0.6, 0.8)$	0.840	0.645	0.7056
	Default setting	0.842	0.651	0.7015
InfinityStar	Best $(0.5, 0.7)$	0.831	57.911	0.81055
	Default setting	0.843	59.039	0.80508

Sensitivity of mask threshold τ and overlap threshold γ in $\mathcal{QR}$. Table 5 shows these two thresholds jointly control the aggressiveness of the pruning criterion. A more conservative setting, such as $(\tau, \gamma) = (0.8, 0.8)$, preserves slightly higher SSIM but yields lower speedups, since fewer layers are identified as redundant. In contrast, lowering either τ or γ enables more aggressive pruning and further improves acceleration, but also leads to a larger drop in SSIM. Nevertheless, the task-level scores remain relatively stable across different threshold settings, indicating that the pruned layers mainly affect fine-grained structural consistency rather than the overall semantic quality.

Since the default setting is obtained by grid search on Infinity and then applied to all models, we further examine whether model-specific threshold tuning is necessary. We conduct additional grid search on HART and InfinityStar and compare the searched thresholds with the default setting in Table 6. The searched settings, $(0.6, 0.8)$ for HART and $(0.5, 0.7)$ for InfinityStar, are close to the default $(0.5, 0.8)$, and their total-score gains over the default setting are marginal. This suggests that the performance is not sensitive to exact threshold values within a reasonable range. Since redundant layers exhibit consistently high overlap in their modification regions, moderate changes in τ or γ mainly affect borderline cases rather than altering the overall pruning decision, allowing the default threshold to generalize across different models.

Effectiveness of overlap-based layer pruning. To evaluate the effectiveness of the proposed overlap criterion in identifying redundant layers, we disable cache-then-reuse and compare overlap-based layer pruning with random layer pruning under different pruning rates. Table 7 shows that random pruning causes a much faster quality drop, especially at high pruning rates, suggesting

Table 7: Comparison between random layer pruning and our overlap-based layer pruning under different pruning rates.

Model	Strategy	30%		50%		80%	
		SSIM↑	Score↑	SSIM↑	Score↑	SSIM↑	Score↑
Infinity	Random	0.939	0.679	0.848	0.676	0.810	0.675
	Overlap-based	**0.951**	**0.680**	**0.922**	**0.678**	**0.908**	**0.677**
HART	Random	0.880	0.515	0.828	0.514	0.779	0.511
	Overlap-based	**0.920**	**0.515**	**0.873**	**0.515**	**0.844**	**0.516**
InfinityStar	Random	0.919	0.832	0.845	0.831	0.793	0.828
	Overlap-based	**0.950**	**0.837**	**0.912**	**0.832**	**0.896**	**0.831**

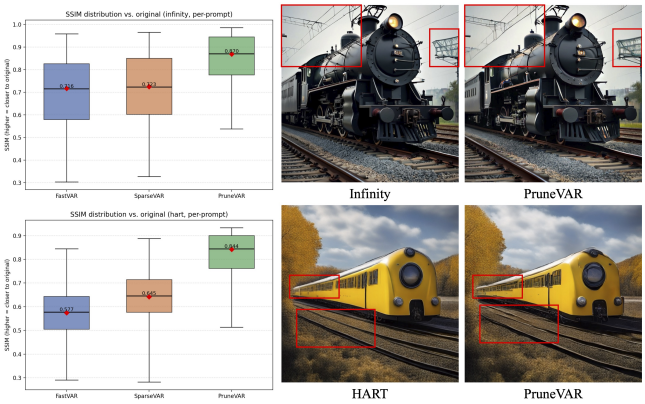


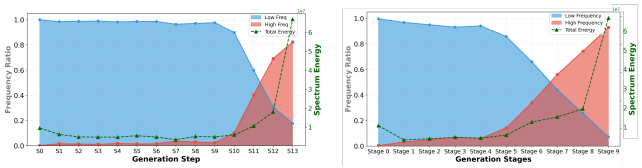
Figure 5: Box plots and failure modes.

that the redundancy in $\mathcal{QR}$ is layer-specific rather than uniformly distributed. Simply removing layers at random is likely to discard layers that still contribute to local refinement. In contrast, overlap-based pruning remains robust as more layers are pruned: the task-level scores stay nearly unchanged from 30% to 80% pruning, and the SSIM remains consistently higher than random pruning across Infinity, HART, and InfinityStar. This confirms that highly overlapping modification regions provide a reliable indicator of layer redundancy, allowing PruneVAR to maximize layer pruning while preserving generation quality.

Generalization to conditional image generation. We further evaluate PruneVAR on conditional image generation with VAR(d=30) [23] to examine whether the proposed acceleration strategy generalizes beyond text-conditioned generation. Since FastVAR and SparseVAR do not provide public code for this setting, we re-implement them on the same VAR backbone for comparison. As shown in Table 8, PruneVAR achieves the best performance among all accelerated methods, with the highest SSIM of 0.795, the lowest FID of 2.16, and the largest speedup of 1.32×. These results indicate that the proposed two-stage redundancy reduction is also effective for conditional image generation, providing a better quality-efficiency trade-off than token-level acceleration strategies.

Table 8: Comparison on VAR.

Method	SSIM↑	FID↓	Speed↑	Lat.↓
VAR(d=30)	-	2.11	-	1.531
+FastVAR	0.651	2.28	1.25×	1.218
+SparseVAR	0.708	2.21	1.12×	1.353
+PruneVAR	**0.795**	**2.16**	**1.32×**	**1.153**



(a) HART. **(b) VAR.**

Figure 6: Frequency analysis of HART and VAR.

Per-prompt robustness. We further analyze the distribution of SSIM scores across different prompts to evaluate the robustness of generation quality. As shown in Figure 5, PruneVAR consistently achieves a higher median and mean SSIM than FastVAR and SparseVAR on both Infinity and HART. More importantly, the SSIM distribution of PruneVAR is more concentrated and has a higher lower bound, indicating that its performance is less sensitive to prompt variations. In contrast, token-pruning baselines show larger fluctuations and more low-SSIM cases, suggesting that their token-level decisions may introduce prompt-dependent structural degradation. These results demonstrate that PruneVAR preserves structural consistency more reliably across diverse prompts, leading to more stable generation quality.

5 Conclusion

In this work, we introduce PruneVAR, a training-free acceleration framework for Visual Autoregressive modeling without decreasing token input. Our key observation is that redundancy in VAR shifts from global stage-level stability to localized layer-wise overlap. Based on this insight, we propose a corse-to-fine layer pruning strategy. By implementing stage-level skipping during intermediate phases and selective layer-wise pruning at late stages, PruneVAR effectively reduces computational overhead while preserving fine-grained texture coherence. Extensive experiments demonstrate that PruneVAR achieves efficient acceleration and maintains high generation fidelity in both text-to-image and text-to-video tasks.

Although PruneVAR achieves consistent acceleration, its two components still have limitations. For cache-then-reuse, this strategy is not universally applicable to all VAR backbones. As shown in Figure 6, HART and VAR do not show a clear stable low-frequency update pattern, possibly because smaller models contain less exploitable cross-stage redundancy. Thus, cache-then-reuse still requires model-dependent identification, and future work could develop adaptive criteria to determine when it should be activated. For layer-pruning, visualizing lower-bound cases in Figure 5 shows that aggressive layer removal may introduce subtle detail distortions, such as twisted lines or local artifacts. This suggests the need to adaptively control the number of pruned layers.

6 Acknowledgments

This work is supported by National Natural Science Foundation of China (No. U24A20326, 62402429, 62441236), Key Research and Development Program of Zhejiang Province (No. 2025C01026), Ningbo Yongjiang Talent Introduction Programme (2023A-397-G), Young Elite Scientists Sponsorship Program by CAST (2024QNRC001) and Zhejiang University Education Foundation Qizhen Scholar Foundation.

References

[1] Xiaoyue Chen, Yuling Shi, Kaiyuan Li, Huandong Wang, Yong Li, Xiaodong Gu, Xinlei Chen, and Mingbao Lin. 2025. Progressive Supernet Training for Efficient Visual Autoregressive Modeling. *arXiv preprint arXiv:2511.16546* (2025).

[2] Zhuokun Chen, Jugang Fan, Zhuowei Yu, Bohan Zhuang, and Mingkui Tan. 2025. Frequency-aware autoregressive modeling for efficient high-resolution image synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 17140–17149.

[3] Zigeng Chen, Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2025. Collaborative decoding makes visual auto-regressive modeling efficient. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 23334–23344.

[4] Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. 2023. Geneval: An object-focused framework for evaluating text-to-image alignment. *Advances in Neural Information Processing Systems* 36 (2023), 52132–52152.

[5] Hang Guo, Yawei Li, Taolin Zhang, Jiangshan Wang, Tao Dai, Shu-Tao Xia, and Luca Benini. 2025. Fastvar: Linear visual autoregressive modeling via cached token pruning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 19011–19021.

[6] Jian Han, Jinlai Liu, Yi Jiang, Bin Yan, Yuqi Zhang, Zehuan Yuan, Bingyue Peng, and Xiaobing Liu. 2025. Infinity: Scaling bitwise autoregressive modeling for high-resolution image synthesis. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 15733–15744.

[7] Kaiyi Huang, Kaiyue Sun, Enze Xie, Zhenguo Li, and Xihui Liu. 2023. T2i-compbench: A comprehensive benchmark for open-world compositional text-to-image generation. *Advances in Neural Information Processing Systems* 36 (2023), 78723–78747.

[8] Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. 2024. Vbench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 21807–21818.

[9] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. 2022. Autoregressive image generation using residual quantization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 11523–11532.

[10] Jiajun Li, Yue Ma, Xinyu Zhang, Qingyan Wei, Songhua Liu, and Linfeng Zhang. 2025. Skipvar: Accelerating visual autoregressive modeling via adaptive frequency-aware skipping. *arXiv preprint arXiv:2506.08908* (2025).

[11] Kunjun Li, Zigeng Chen, Cheng-Yen Yang, and Jenq-Neng Hwang. 2025. Memory-efficient visual autoregressive modeling with scale-aware kv cache compression. *arXiv preprint arXiv:2505.19602* (2025).

[12] Senmao Li, Kai Wang, Salman Khan, Fahad Shahbaz Khan, Jian Yang, and Yaxing Wang. 2025. StageVAR: Stage-Aware Acceleration for Visual Autoregressive Models. *arXiv preprint arXiv:2512.16483* (2025).

[13] Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. 2024. Autoregressive image generation without vector quantization. *Advances in Neural Information Processing Systems* 37 (2024), 56424–56445.

[14] Ying Li, Huan Wang, et al. [n. d.]. FreqExit: Enabling Early-Exit Inference for Visual Autoregressive Models via Frequency-Aware Guidance. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

[15] Zekun Li, Ning Wang, Tongxin Bai, Changwang Mei, Peisong Wang, Shuang Qiu, and Jian Cheng. 2026. SparVAR: Exploring Sparsity in Visual AutoRegressive Modeling for Training-Free Acceleration. *arXiv preprint arXiv:2602.04361* (2026).

[16] Jinlai Liu, Jian Han, Bin Yan, Hui Wu, Fengda Zhu, Xing Wang, Yi Jiang, Bingyue Peng, and Zehuan Yuan. 2025. Infinitystar: Unified spacetime autoregressive modeling for visual generation. *arXiv preprint arXiv:2511.04675* (2025).

[17] Ziran Qin, Youru Lv, Mingbao Lin, Hang Guo, Zeren Zhang, Danping Zou, and Weiyao Lin. 2026. Head-aware kv cache compression for efficient visual autoregressive modeling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 40. 24982–24990.

[18] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*. PmLR, 8748–8763.

[19] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. 2019. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems* 32 (2019).

[20] Kaiyue Sun, Kaiyi Huang, Xian Liu, Yue Wu, Zihan Xu, Zhenguo Li, and Xihui Liu. 2025. T2v-compbench: A comprehensive benchmark for compositional text-to-video generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 8406–8416.

[21] Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. 2024. Autoregressive model beats diffusion: Llama for scalable image generation. *arXiv preprint arXiv:2406.06525* (2024).

[22] Haotian Tang, Yecheng Wu, Shang Yang, Enze Xie, Junsong Chen, Junyu Chen, Zhuoyang Zhang, Han Cai, Yao Lu, and Song Han. 2024. Hart: Efficient visual generation with hybrid autoregressive transformer. *arXiv preprint arXiv:2410.10812* (2024).

[23] Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. 2024. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *Advances in neural information processing systems* 37 (2024), 84839–84865.

[24] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. 2016. Conditional image generation with pixelcnn decoders. *Advances in neural information processing systems* 29 (2016).

[25] Jort Vincenti, Metod Jazbec, and Guoxuan Xia. 2025. Dynamic Mixture-of-Experts for Visual Autoregressive Model. *arXiv preprint arXiv:2510.08629* (2025).

[26] Xinlong Wang, Xiaosong Zhang, Zhengxiong Luo, Quan Sun, Yufeng Cui, Jinsheng Wang, Fan Zhang, Yueze Wang, Zhen Li, Qiying Yu, et al. 2024. Emu3: Next-token prediction is all you need. *arXiv preprint arXiv:2409.18869* (2024).

[27] Yuqing Wang, Shuhuai Ren, Zhijie Lin, Yujin Han, Haoyuan Guo, Zhenheng Yang, Difan Zou, Jiashi Feng, and Xihui Liu. 2025. Parallelized autoregressive visual generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12955–12965.

[28] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13, 4 (2004), 600–612.

[29] Rui Xie, Tianchen Zhao, Zhihang Yuan, Rui Wan, Wenxi Gao, Zhenhua Zhu, Xuefei Ning, and Yu Wang. 2024. Litevar: Compressing visual autoregressive modelling with efficient attention and quantization. *arXiv preprint arXiv:2411.17178* (2024).

[30] Boxun Xu, Yu Wang, Zihu Wang, and Peng Li. 2026. AMS-KV: Adaptive KV Caching in Multi-Scale Visual Autoregressive Transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 40. 27206–27214.

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

[32] Kaixin Zhang, Ruiqing Yang, Yuan Zhang, Shan You, and Tao Huang. 2025. ActVAR: Activating Mixtures of Weights and Tokens for Efficient Visual Autoregressive Generation. *arXiv preprint arXiv:2511.12893* (2025).

[33] Yu Zhang, Jingyi Liu, Feng Liu, Duoqian Miao, Qi Zhang, Kexue Fu, Changwei Wang, and Longbing Cao. 2026. Adaptive Visual Autoregressive Acceleration via Dual-Linkage Entropy Analysis. *arXiv preprint arXiv:2602.01345* (2026).