LineAR: Autoregressive Image Generation Needs Only a Few Lines of Cached Tokens

Ziran Qin[*1], Youru Lv[*1], Mingbao Lin[2], Zeren Zhang[3], Chaofan Gan[1], Tieyuan Chen[1], Liquan Shen[4], Junhui Hou[5], Chern Hong Lim[6], Fei Wen[1], and Weiyao Lin[✉1]

[1] Shanghai Jiao Tong University [2] Rakuten [3] Peking University
[4] Shanghai University [5] City University of Hong Kong [6] Monash University

{qinziran,wylin}@sjtu.edu.cn
Project Page

Abstract. Autoregressive (AR) visual generation has emerged as a powerful paradigm for image and multimodal synthesis, owing to its scalability and generality. However, existing AR image generation suffers from severe memory bottlenecks due to the need to cache all previously generated visual tokens during decoding, leading to both high storage requirements and low throughput. In this paper, we introduce **LineAR**, a novel, training-free progressive key-value (KV) cache compression pipeline for AR image generation. By exploiting the intrinsic characteristics of visual attention, LineAR manages the cache from a 2D line-level perspective, preserving the visual dependency regions while progressively evicting less-informative tokens under inter-line attention guidance, leveraging the spatial continuity prior of images to ensure that each eviction step is harmless to subsequent line generation. Experiments across seven AR image generation models validate that LineAR achieves lossless or even improved generation quality with only a few cached lines. It improves ImageNet FID from 2.77 to 2.68 on LlamaGen-XL and COCO FID from 23.85 to 22.86 on Janus-Pro-1B at 1/6 budget ratio, and also improves DPG and HPSv2.1 scores on Lumina-mGPT-768 with just 1/8 KV cache. Additionally, LineAR achieves significant memory and throughput gains across devices, e.g., up to **67.06%** memory reduction and **4.17**× speedup on LlamaGen-XL, and **73.65%** memory reduction and **4.34**× speedup on Janus-Pro-1B, evaluated on NVIDIA RTX PRO 6000 GPUs.

Keywords: Visual autoregressive model · Image generation · Efficiency

1 Introduction

Autoregressive (AR) models with the "next-token" prediction paradigm have achieved remarkable success in large language models (LLMs) [1, 28] and have

[*] Equal contribution.
[✉] Corresponding author.

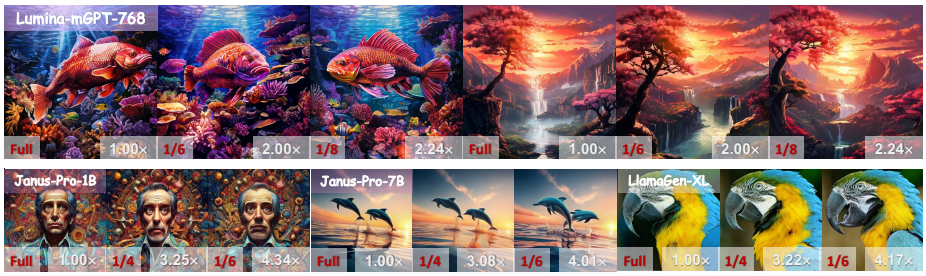

Fig. 1: LineAR enables efficient autoregressive image generation, preserving only lines of KV cache, achieving up to 2.24×, 4.34×, and 4.17× speedup on Lumina-mGPT, Janus-Pro, and LlamaGen, with improved or comparable generation quality.

recently been extended to visual generation, producing high-quality results comparable to state-of-the-art diffusion models [17, 40]. Their inherent scalability and generality position AR models as a promising foundation for unified multimodal understanding and generation [29, 49]. Despite these advances, AR image generation suffers from severe inference inefficiencies. The greedy accumulation of a heavy key-value (KV) cache from all previously generated tokens leads to a memory overhead that scales linearly with sequence length. For instance, generating a 1024×1024 image with Lumina-mGPT [29] requires caching over 4K tokens, resulting in excessive memory consumption and significantly degraded throughput, which severely complicates practical deployment.

A natural remedy is to constrain the continuous growth of the KV cache during decoding, yet directly transplanting KV cache compression techniques from LLMs [10, 11, 34, 61] to visual generation is non-trivial. The core difficulty lies in two fundamental gaps. ***Stage mismatch***: KV cache compression in LLMs mainly targets redundant long textual contexts in the *prefill* phase via one-shot truncation, whereas AR visual models build their cache progressively during *decoding*, with token importance shifting dynamically as spatial generation proceeds. ***Modality gap***: treating the visual cache as a flat 1D sequence, as in text, neglects the two-dimensional structure of images. As illustrated in Fig. 2, 1D token-level eviction [53, 61] greedily retains globally "informative" tokens but shatters spatial proximity, easily causing structure collapse (Fig. 2(a)); 1D local-window eviction [54] preserves recent context but discards distant anchors, incurring semantic loss (Fig. 2(b)). Neither strategy is satisfactory.

To bridge these gaps, we systematically investigate how attention behaves inside AR visual models and uncover three key observations (Fig. 3). ***(i) Visual attention dilution***: as decoding progresses, per-token visual attention steadily dilutes while conditional tokens absorb an increasing share, revealing heavy redundancy in the cached visual tokens. ***(ii) Locally concentrated visual dependency***: the generation process relies heavily on adjacent spatial contexts and initial visual anchors, leaving the mid-region of the cache largely dispensable. ***(iii) Inter-line attention consistency***: adjacent raster lines share

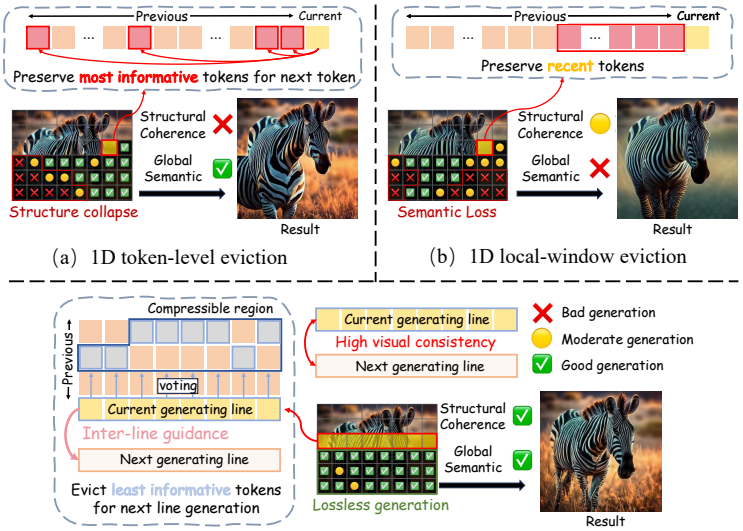

Fig. 2: Comparison of KV cache eviction strategies for AR image generation. (a) 1D token-level eviction preserves isolated informative tokens but destroys spatial proximity, causing structure collapse; (b) 1D local-window eviction retains recent context but discards important anchors, leading to semantic loss; (c) Our LineAR adopts a 2D line-level perspective with inter-line guided eviction, preserving both structural coherence and global semantics for lossless generation.

highly similar attention distributions, so the current line's attention serves as a reliable proxy for identifying which cached tokens the next line will need.

Building on these insights, we propose LineAR, a novel training-free progressive KV cache compression pipeline tailored for autoregressive visual models. LineAR departs from traditional 1D sequential processing by managing the KV cache from a *2D line-level* perspective, explicitly dividing the image generation process into rasterized line stages. This enables efficient cache management that fully exploits the locality and inter-line consistency inherent in visual generation. LineAR progressively removes less informative tokens for next line generation under inter-line guidance (Fig. 2(c)). As a result, LineAR liberates AR visual models from heavy KV cache dependencies, enabling high-quality image generation using only a few lines of cached tokens. It achieves substantial memory reductions and throughput accelerations while maintaining, and in some cases improving, generation quality, all without requiring any model retraining. The contributions of our paper include:

1. We conduct an in-depth analysis of AR visual models, revealing inefficiencies in visual attention during the decoding process and identifying inherent visual characteristics, including locally concentrated visual dependencies and inter-line attention consistency.

2. We introduce LineAR, a simple yet highly effective decoding-time KV cache compression pipeline. By performing progressive compression leveraging interline attention guidance, LineAR empowers efficient AR image generation with only a few lines of cache.
3. We extensively validate LineAR on seven AR visual models spanning class-conditional and text-to-image generation. Experiments demonstrate that LineAR achieves significant memory and throughput gains while maintaining lossless or even improved generation quality.

2 Related Work

Autoregressive Visual Generation. Autoregressive (AR) models [1, 2, 8, 28, 57] have shown great success in text generation, leading to their extension into image synthesis. Pioneering works [17, 37, 40, 48, 60] paved the way by quantizing images into discrete tokens [9, 45, 59], decoded autoregressively in a GPT-style manner. Building on this, recent advances [5, 29, 42, 49, 51, 55] adopt architectures closely aligned with text generation, enabling a unified framework for multimodal understanding and generation. However, next-token prediction suffers from severe inference inefficiency, as each token must be generated sequentially. To alleviate this, alternative designs explore more efficient paradigms, such as next-scale [16, 39, 41, 44, 46] and masked autoregressive modeling [4, 24, 25]. Despite these, all methods still require caching previously generated tokens, causing significant storage overhead. Our work focuses on the next-token paradigm, the most general and widely adopted across modalities.

Efficient Autoregressive Visual Generation. Recent efforts have focused on improving the efficiency of visual AR generation. Parallel decoding [18, 19, 22, 33, 43, 50, 62] approaches aim to reduce decoding steps by predicting multiple visual tokens simultaneously, thereby accelerating generation. Despite their effectiveness, they often require retraining AR models to learn alternative decoding orders [19, 33, 50, 62], or trade additional computation [18, 43] for reduced latency. More importantly, they fail to reduce the memory footprint of cached tokens during generation, resulting in nearly unchanged peak memory usage. Other strategies, such as token sparsification [6, 15, 35] and cache reuse [56], optimize generation efficiency but are typically tailored to specific paradigms such as next-scale prediction [44] or masked autoregressive modeling [25], making them unsuitable for the next-token prediction approach that our work targets.

KV Cache Compression. KV cache compression [23, 31, 47, 61], particularly eviction-based methods [10–12, 26, 30, 32, 34, 38, 58, 61], effectively mitigates KV cache memory bottlenecks by enforcing capacity budgets. While successful in LLMs, they mostly focus on the prefill phase KV cache compression, truncating redundant tokens from long inputs in a one-shot manner. In contrast, AR visual models generate long token sequences progressively during decoding, where token importance evolves dynamically. Although decoding-time methods like StreamingLLM [54] and H2O [61] employ local-window or greedy per-token updates, and recent works like R-KV [3] optimize reasoning LLMs [14] by considering contextual redundancy, they uniformly operate from a 1D sequence-based

perspective. Consequently, these methods prove suboptimal when directly applied to AR image generation, as they neglect the critical spatial continuity of images and fail to exploit inherent visual characteristics. To fill this gap, our work pioneers a 2D-aware decoding-time KV cache compression pipeline explicitly tailored for visual AR models.

3 Method

3.1 Preliminaries

Autoregressive Visual Generation. Autoregressive visual generation follows next-token paradigm to synthesize visual content in raster order. Given a textual prompt (or class label) tokenized into a sequence of condition tokens $\mathbf{c} \in \mathbb{R}^C$, the model generates a sequence of visual tokens $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$, where each token is predicted based on the conditional prefix and previously generated tokens:

$$p(\mathbf{x}_{1:N} \mid \mathbf{c}) = \prod_{i=1}^{N} p(\mathbf{x}_i \mid \mathbf{c}, \mathbf{x}_{<i}), \tag{1}$$

where $N = h \times w$ denotes the number of visual tokens on the latent feature grid. After decoding, the tokens are mapped back to pixel space by a pretrained visual decoder and rearranged to form an image of size $H \times W$.

KV Cache in Visual Generation. Autoregressive visual generation typically involves two phases: (1) *prefill phase*: the model encodes text prompts or class labels into C conditional tokens, forming the conditional key–value (KV) states $\{\mathbf{K}_{\text{cond}}, \mathbf{V}_{\text{cond}}\} \in \mathbb{R}^{C \times d}$. (2) *decoding phase*: the model autoregressively generates subsequent visual tokens in multiple decoding steps. For a single attention head at decoding step i, the attention is computed as:

$$\text{Attention}(\mathbf{q}_i, \mathbf{K}^{(i)}, \mathbf{V}^{(i)}) = \text{Softmax}\left(\frac{\mathbf{q}_i {\mathbf{K}^{(i)}}^\top}{\sqrt{d}}\right)\mathbf{V}^{(i)}, \tag{2}$$

where $\mathbf{q}_i \in \mathbb{R}^{1 \times d}$ is the query of the i-th visual token, and $\{\mathbf{K}^{(i)}, \mathbf{V}^{(i)}\} \in \mathbb{R}^{(C+i) \times d}$ denote the current KV states consisting of both conditional and visual tokens:

$$\mathbf{K}^{(i)} = [\mathbf{K}_{\text{cond}}, \mathbf{K}_{1:i}], \qquad \mathbf{V}^{(i)} = [\mathbf{V}_{\text{cond}}, \mathbf{V}_{1:i}]. \tag{3}$$

During inference, these KV pairs are cached to avoid recomputing projections for previous tokens. At any decoding step i, the visual KV cache is updated by concatenating the previous cache $\{\mathbf{K}_{1:i-1}, \mathbf{V}_{1:i-1}\}$ with the newly generated KV pair $\{\mathbf{k}_i, \mathbf{v}_i\}$, forming:

$$\mathbf{P}_{1:i} = \{\mathbf{K}_{1:i}, \mathbf{V}_{1:i}\} = \{[\mathbf{K}_{1:i-1}, \mathbf{k}_i], [\mathbf{V}_{1:i-1}, \mathbf{v}_i]\}. \tag{4}$$

For simplicity, we denote the visual KV cache at step i by $\mathbf{P}_{1:i}$. While KV cache effectively eliminates redundant computation, its memory footprint grows linearly with generated visual tokens N, resulting in $\mathcal{O}(N)$ storage that dominates inference cost at high resolutions and severely limits decoding efficiency. This underscores the urgent need for decoding-time KV cache optimization.

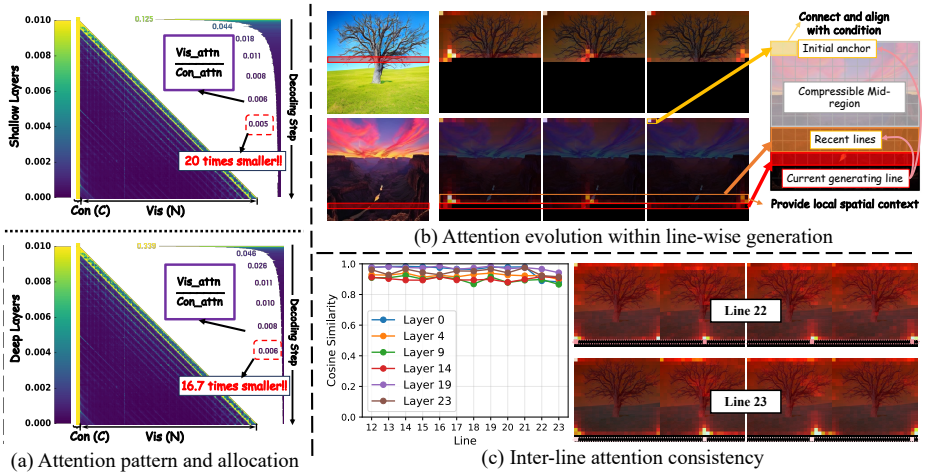

Fig. 3: (a)Visualization of attention pattern and allocation. The attention persistently shifts toward the conditional tokens, while visual attention gradually dilutes as decoding progresses. **(b)Attention dynamics within line-wise generation.** The evolving attention demonstrates that generating a line heavily depends on the initial anchor and neighboring recent lines, thereby leaving the mid-region highly redundant and compressible. **(c) Inter-line attention consistency.** (Left) Attention similarity between adjacent lines. (Right) Attention visualization for past generated regions across two adjacent lines. Inter-line attention shows high consistency.

3.2 Revisiting Visual Autoregressive Attention

We first empirically analyze the attention mechanism using Janus-Pro [5] on DPG prompts [21] to address three fundamental questions: (1) *Is it necessary to cache all historical visual tokens?* (2) *If not, which specific regions of the visual cache should be the primary targets for eviction?* (3) *How can we execute this progressive eviction without degrading subsequent generations?*

Observation 1: Visual Attention Dilution. Fig. 3(a) illustrates the attention patterns and the attention allocation ratio between visual and conditional tokens in autoregressive visual generation. In early decoding steps, visual tokens closely align with conditional tokens to establish the overall visual style and color tone of the generated image [53], thereby receiving a comparable attention allocation. However, as generation proceeds, visual attention becomes increasingly diluted, persistently sinking toward the conditional tokens (whose average attention allocation grows up to 20× higher than visual tokens in shallow layers). In models such as Lumina-mGPT [29], the number of cached visual tokens can exceed the conditional ones by more than 200×, making the vanilla KV cache mechanism highly inefficient for autoregressive visual generation.

Observation 2: Locally Concentrated Visual Dependency. We further study how visual attention evolves by visualizing the attention heatmap in line-by-line generation, which naturally aligns with the raster-scan order. As shown

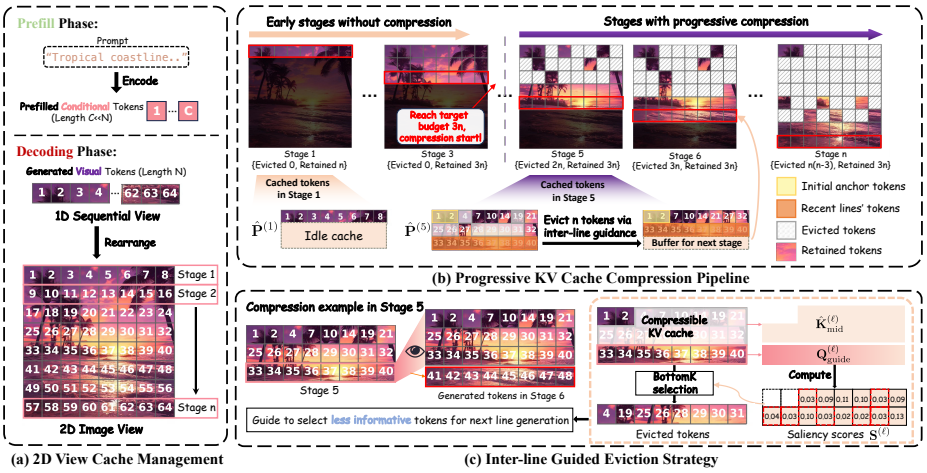

Fig. 4: Overview of the proposed LineAR framework. We illustrate a toy example of an 8×8 image grid with a budget ratio $\rho = 3/8$ in one attention head, featuring **(a) 2D View Visual KV Cache Management** to spatially organize the cache by horizontal lines (where each line serves as a natural generating and management unit), **(b) Progressive KV Cache Compression Pipeline** to progressively reduce cached visual tokens, and **(c) Inter-line Guided Eviction Strategy** to safely discard the least informative tokens leveraging inter-line attention consistency.

in Fig. 3(b), attention is primarily focused on spatially adjacent tokens and the initial visual tokens that are strongly linked to the conditional tokens. This highlights that visual generation relies heavily on local spatial context and persistent dependence on the initial anchor tokens that encode the global style and conditional alignment. Consequently, the generation of the current line is mainly dependent on neighboring lines and initial anchor tokens, while distant visual tokens from the mid-region contribute little to the prediction of subsequent tokens. This redundancy makes it intuitive and feasible to progressively discard less informative tokens from the mid-region during the generation process.

Observation 3: Inter-Line Attention Consistency. Unlike text generation, where adjacent sentences often undergo abrupt semantic shifts, images exhibit a strong spatial continuity that directly manifests in the attention space. As shown in Fig. 3(c), the attention distributions of consecutive raster lines overlap substantially: both the heatmap visualization and the cosine similarity curve confirm that adjacent lines attend to nearly identical regions of the cached history, with this consistency holding across all layers and generation stages. This finding is crucial for progressive compression. It indicates that the attention pattern observed during the current line's generation serves as a faithful preview of what the next line will require. We exploit this inter-line consistency as the core guiding principle of our compression strategy, ensuring that each eviction step is safely and reliably informed by the intrinsic generation process itself, rather than relying on sub-optimal heuristic proxies.

3.3 LineAR

We present *LineAR*, a decoding-time KV cache compression pipeline that translates our analytical insights to enable efficient autoregressive image generation.
Visual KV Cache Management in 2D Line-level View. Departing from the vanilla 1D sequential view, we manage the visual KV cache from a *2D view*, which fundamentally respects the structural integrity of images. Let the total number of visual tokens be $N = h \times w = n^2$ for simplicity. As illustrated in Fig. 4(a), the full KV cache is conceptualized as an $n \times n$ matrix following the raster order, where each position corresponds to a latent image feature. Under this view, the generation process can be segmented into a sequence of n raster-line stages, each corresponding to the generation of one horizontal line consisting of n visual tokens. We denote by $\mathbf{P}^{(\ell)} \triangleq \{\mathbf{K}^{(\ell)} = \mathbf{K}_{1:n\ell}, \mathbf{V}^{(\ell)} = \mathbf{V}_{1:n\ell}\}$ the state of the KV cache after the ℓ-th line has been generated. This 2D view introduces a natural management unit, the raster line, which allows the model to enforce spatial coherence during progressive compression.
Cache Budget and Budget Ratio. We set the cache budget B as the upper bound on the number of cached KV pairs and assume that it is an integer multiple of n, *i.e.*, $n \mid B$, so that the cache can be managed at the granularity of complete raster lines. We further define the budget ratio $\rho = B/N$.
Progressive KV Cache Compression Pipeline. Fig. 4(b) illustrates the overall pipeline. Given a target budget ratio ρ, we maintain a cache space upper bounded by $|\hat{\mathbf{P}}| \leq \rho N = B$ during generation. The cache is regulated in a stable and orderly manner, using the completion of each raster line as a synchronization point. Recall that we split AR image generation into n stages, in the early stages' generation ($\ell/n < \rho$), no compression is applied, allowing the model to first establish a consistent global style and remain fully guided by the conditional context. Once the generated context exceeds the predefined cache bound, progressive KV compression is activated in the following stages.

At any end of line ℓ when compression triggers, $\hat{\mathbf{P}}^{(\ell)}$ can be always partitioned into three parts: (1) Initial anchor tokens $\hat{\mathbf{P}}^{(\ell)}_{\text{init}} = \mathbf{P}_{1:N_{\text{init}}}$ that maintain the global style and conditional alignment; (2) Recent lines' tokens $\hat{\mathbf{P}}^{(\ell)}_{\text{rec}} = \mathbf{P}_{(\ell-r)n+1:\ell n}$ that provide local visual context; and (3) The compressible mid-region tokens $\hat{\mathbf{P}}^{(\ell)}_{\text{mid}}$, which lie between the initial and recent lines. For $\hat{\mathbf{P}}^{(\ell)}_{\text{mid}}$, we then perform the inter-line guided eviction strategy $\Phi_{\text{inter}}(\cdot)$, which evicts the least n informative tokens for next line generation, $\hat{\mathbf{P}}^{(\ell)}_{\text{evict}} \in \mathbb{R}^{n\times d}$. We compact mid-region cache $\hat{\mathbf{P}}^{(\ell)}_{\text{mid}}$ by purging these evicted tokens:

$$\hat{\mathbf{P}}^{(\ell)}_{\text{evict}} = \Phi_{\text{inter}}(\hat{\mathbf{P}}^{(\ell)}_{\text{mid}}, n), \quad \hat{\mathbf{P}}^{(\ell)}_{\text{mid}} \leftarrow \hat{\mathbf{P}}^{(\ell)}_{\text{mid}} \setminus \hat{\mathbf{P}}^{(\ell)}_{\text{evict}}. \tag{5}$$

After compression, the cache $\hat{\mathbf{P}}^{(\ell)}$ is updated by:

$$\hat{\mathbf{P}}^{(\ell)} \leftarrow [\hat{\mathbf{P}}^{(\ell)}_{\text{init}}, \hat{\mathbf{P}}^{(\ell)}_{\text{mid}}, \hat{\mathbf{P}}^{(\ell)}_{\text{rec}}], \quad |\hat{\mathbf{P}}^{(\ell)}| = B - n. \tag{6}$$

Note that, each compression operation guarantees an n-token buffer for the next line's generation. At the subsequent stage ℓ+1, the cache dynamically absorbs the newly generated line ℓ+1:

$$\hat{\mathbf{P}}^{(\ell+1)} = \big[\hat{\mathbf{P}}^{(\ell)},\ \mathbf{P}_{\ell n+1:(\ell+1)n}\big], \tag{7}$$

and then the compression operation is repeated iteratively. This completes the progressive KV cache compression pipeline, maintaining anchors and locality, enforcing the budget, and leaving a one-line buffer, thus balancing compression frequency and generation stability.

Inter-line Guided Eviction Strategy Φ_{inter}. In the progressive compression pipeline, the initial anchor $\hat{\mathbf{P}}^{(\ell)}_{\text{init}}$ is fixed and the recent window $\hat{\mathbf{P}}^{(\ell)}_{\text{rec}}$ advances line by line. The key challenge is to prudently evict tokens from the mid-region without affecting the synthesis of the next line. Leveraging the strong visual consistency between adjacent image lines, we design the inter-line guided eviction strategy (shown in Fig. 4(c)), which utilizes similarity information from the adjacent line to guide token selection.

For the uncompressed key-value states of the remaining mid tokens $\hat{\mathbf{P}}^{(\ell)}_{\text{mid}} = \{\hat{\mathbf{K}}^{(\ell)}_{\text{mid}}, \hat{\mathbf{V}}^{(\ell)}_{\text{mid}}\}$, our goal is to evict n tokens that are least informative for the next line's generation. To this end, we utilize the attention queries of the currently generated line as an active proxy to evaluate the mid-region's importance for the upcoming line. Specifically, we maintain a guide queue of queries from the current generating line ℓ:

$$\mathbf{Q}^{(\ell)}_{\text{guide}} = [\,\mathbf{q}_{(\ell-1)n+1}, \ldots, \mathbf{q}_{\ell n}\,] \in \mathbb{R}^{n\times d}. \tag{8}$$

We then compute saliency scores for the mid-region by averaging the attention cast by current line's queries:

$$\mathbf{S}^{(\ell)} = \frac{1}{n}\,\mathbf{1}^{\top}\text{Softmax}\Big(\frac{\mathbf{Q}^{(\ell)}_{\text{guide}}\hat{\mathbf{K}}^{(\ell)\top}_{\text{mid}}}{\sqrt{d}}\Big) \in \mathbb{R}^{|\hat{\mathbf{P}}^{(\ell)}_{\text{mid}}|}, \tag{9}$$

which ensures that each token from the current line equally votes on the eviction decision. This prevents any single query token from dominating the selection, thereby avoiding biased eviction that could compromise the spatial coherence of the upcoming line. We then select the n least salient tokens to be evicted:

$$\mathcal{E}^{(\ell)} = \text{BottomK}\big(\mathbf{S}^{(\ell)}, n\big), \quad \hat{\mathbf{P}}^{(\ell)}_{\text{evict}} = \hat{\mathbf{P}}^{(\ell)}_{\text{mid}}[\mathcal{E}^{(\ell)}]. \tag{10}$$

In this way, we preserve the most important information, achieving lossless compression for the next line generation.

4 Experimentation

4.1 Experimental Setup

Evaluated Models. We evaluate LineAR on seven state-of-the-art autoregressive image generation models, covering three representative architectures. (1) Janus-Pro-1B/7B [5], unified multimodal models generating 384×384 images with 576 visual tokens; (2) Lumina-mGPT-768/1024 [29], text-to-image models

generating 768×768 and 1024×1024 images, with 2352 and 4160 visual tokens; (3) LlamaGen-L/XL/XXL [40], class-conditional models generating 384×384 images with 576 visual tokens.

Evaluation Metrics. For text-to-image generation, we employ three widely used benchmarks: GenEval [13] and DPG [21] to assess high-level semantic alignment and compositional consistency, and HPSv2.1 [52] for perceptual quality. We further evaluate image quality and prompt adherence using Fréchet Inception Distance [20] (FID) and CLIP score [36] on 30k MS-COCO [27] validation captions. We denote FID for MS-COCO as FID-30k, which remains the de facto standard metric for benchmarking overall image synthesis quality. For class-conditional image generation, we follow standard ImageNet evaluation protocols and report FID-50k, Inception Score (IS), Precision (Prec.), and Recall, computed from 50k generated samples.

Implementation Details. For all evaluated models, we use their official codebases with default inference configurations and FlashAttention-2 [7] enabled. For LineAR, we retain the two most recent lines ($r = 2$, reduced to $r = 1$ under the aggressive 1/8 budget ratio) and maintain a queue of queries from the current line for the inter-line guided eviction strategy. We report results under different budget ratios ($\rho = B/N$) in the following evaluations.

4.2 Main Results

Text-to-Image Generation. *Quantitative results.* Tab. 1 presents the results of applying LineAR to four models spanning different architectures, model sizes, and generation resolutions. Across all cases, LineAR consistently achieves lossless or even improved performance compared to the full-cache baseline. For instance, on Lumina-mGPT-768, LineAR achieves higher DPG and HPSv2.1 scores with only 1/8 visual cache retained. Beyond its strong performance on high-resolution generation, LineAR also performs effectively on Janus-Pro models, which generate low-resolution images with only 576 visual tokens. Even when the cache is compressed to a ratio of 1/6, LineAR maintains nearly identical generation quality. Specifically, it exhibits only marginal drops in GenEval (0.02 and 0.01) and DPG (0.08 and 0.25) on Janus-Pro-1B and 7B, respectively, while achieving even higher HPSv2.1 scores than the full-cache baseline.

Qualitative results. Fig. 5 presents images generated with Lumina-mGPT-768 and Janus-Pro-7B using LineAR under different budget ratios. Across models, LineAR preserves high visual quality that remains well aligned with textual prompts, maintaining similar style and color tones to the full cache, even under high compression ratios. Along with the quantitative results, this demonstrates the effectiveness of LineAR. Designed with the characteristics of visual generation in mind, it effectively preserves conditional semantics and achieves superior perceptual quality through attention regularization, overcoming the inefficiencies of standard full-cache attention.

Class-conditional Image Generation. We further evaluate LineAR on LlamaGen families of varying sizes from L to XXL on the ImageNet dataset. As shown in Tab. 2, LineAR consistently achieves comparable FID performance

Table 1: Quantitative results of text-to-image generation on Janus-Pro-1B, Janus-Pro-7B, Lumina-mGPT-768 and Lumina-mGPT-1024 with LineAR under different budget ratios. We evaluate using GenEval, DPG and HPSv2.1 benchmarks.

Models	ρ	GenEval↑					DPG↑				HPSv2.1↑
		Single Obj.	Two Obj.	Counting	Colors	Overall	Entity	Relation	Attribute	Overall	
Janus-Pro-1B ($N = 576$)	Full	0.98	0.83	0.50	0.90	0.73	89.19	93.42	87.19	82.88	21.59
	1/4	0.99	0.86	0.49	0.89	0.72	89.14	93.19	87.63	82.76	22.22
	1/6	0.99	0.81	0.51	0.89	0.71	89.34	92.46	87.59	82.80	22.70
Janus-Pro-7B ($N = 576$)	Full	0.99	0.86	0.57	0.91	0.79	90.21	93.81	87.89	84.72	23.76
	1/4	0.99	0.89	0.58	0.93	0.80	90.05	93.93	87.83	84.84	23.88
	1/6	0.99	0.88	0.52	0.91	0.78	89.84	93.42	87.61	84.47	24.09
Lumina-mGPT-768 ($N = 2352$)	Full	1.00	0.73	0.25	0.82	0.52	82.31	87.16	80.67	74.91	28.53
	1/6	0.99	0.74	0.28	0.84	0.52	81.94	87.81	79.74	75.58	28.56
	1/8	0.99	0.73	0.24	0.86	0.52	82.99	87.85	80.51	75.24	28.56
Lumina-mGPT-1024 ($N = 4160$)	Full	0.98	0.76	0.29	0.83	0.56	86.74	90.87	84.06	79.54	28.93
	1/8	0.99	0.75	0.30	0.82	0.55	86.58	90.75	83.96	79.50	28.83

Fig. 5: Qualitative results of text-to-image generation by LineAR on Lumina-mGPT-768 (left) and Janus-Pro-7B (right).

to the full-cache baseline, even when the cache budget ratio is reduced to 1/8 on all models. Notably, LineAR achieves lower FID scores (2.68 vs. 2.77) on LlamaGen-XL with a 1/6 budget ratio and shows improvements in both FID and Inception Score on LlamaGen-XXL with a 1/4 budget ratio. These results demonstrate that LineAR not only significantly improves the efficiency of autoregressive image generation but also reduces visual redundancy in cached tokens, thereby enhancing generation quality.

4.3 Comparison with Baselines

We compare LineAR against representative baselines adapted for AR image generation: StreamingLLM [54] (local-window updating), H2O [61] (attention-based eviction), R-KV [3] (redundancy-aware selection), and the sparse attention method ADSA [53] (diversity-preserving eviction via value cosine similarity).

Quantitative Comparison. We first conduct quantitative evaluations under various cache budgets on both class-conditional (ImageNet) and text-to-image

Table 2: Quantitative results of class-conditional generation on LlamaGen-L/XL/XXL with LineAR under different budget ratios.

Models	ρ	FID-50k↓	IS↑	Prec.↑	Recall↑
LlamaGen-L ($N = 576$)	Full	3.13	257.43	0.83	0.52
	1/4	3.11	256.70	0.83	0.51
	1/6	3.22	260.12	0.83	0.51
	1/8	3.37	264.13	0.82	0.51
LlamaGen-XL ($N = 576$)	Full	2.77	282.41	0.83	0.54
	1/4	2.64	279.95	0.83	0.55
	1/6	2.68	279.17	0.82	0.55
	1/8	2.95	282.91	0.82	0.54
LlamaGen-XXL ($N = 576$)	Full	2.47	290.82	0.83	0.56
	1/4	2.45	292.47	0.83	0.56
	1/6	2.54	290.14	0.82	0.56
	1/8	2.64	292.92	0.82	0.56

Fig. 6: Qualitative comparison with baselines on Janus-Pro-1B with $\rho = 1/6$.

Table 3: Quantitative comparison of compression methods on class-conditional and text-to-image generation. The efficiency comparison is conducted on RTX PRO 6000.

Methods	ρ	LlamaGen-XL					Janus-Pro-1B				
		Latency ↓	Through. ↑	Speedup↑	FID-50k ↓	IS↑	Latency ↓	Through. ↑	Speedup↑	FID-30k↓	CLIP ↑
Full	1	71.27s	3.59it/s	1.00×	2.77	282.41	87.36s	2.93it/s	1.00×	23.85	0.233
Streaming	1/6	16.73s	15.30it/s	4.26×	3.09	307.69	19.62s	13.05it/s	4.45×	24.80	0.236
	1/8	14.53s	17.62it/s	4.91×	4.37	297.08	16.63s	15.40it/s	5.25×	32.17	0.228
H2O	1/6	18.49s	13.85it/s	3.85×	3.06	287.67	21.99s	11.64it/s	3.97×	25.87	0.225
	1/8	15.83s	16.17it/s	4.50×	3.72	279.05	18.51s	13.83it/s	4.72×	29.46	0.214
ADSA	1/6	17.17s	14.91it/s	4.15×	3.34	270.89	20.71s	12.36it/s	4.22×	23.63	0.236
	1/8	14.93s	17.15it/s	4.77×	4.90	247.70	17.22s	14.86it/s	5.07×	24.98	0.236
R-KV	1/6	17.47s	14.65it/s	4.08×	2.77	276.56	20.47s	12.51it/s	4.27×	24.63	0.232
	1/8	15.13s	16.92it/s	4.71×	3.45	271.29	17.70s	14.46it/s	4.94×	25.55	0.232
LineAR	1/6	17.10s	14.97it/s	4.17×	2.68	279.17	20.12s	12.72it/s	4.34×	22.86	0.236
	1/8	14.85s	17.24it/s	4.80×	2.95	282.91	16.96s	15.09it/s	5.15×	24.03	0.237

(MS-COCO) generation, utilizing LlamaGen-XL and Janus-Pro-1B, respectively. As shown in Tab. 3, LineAR consistently outperforms the baselines in generation fidelity, achieving the best FID scores across all tasks and budget ratios. At a moderate budget ratio of 1/6, most methods maintain relatively stable performance by preserving essential tokens to some extent. However, under a more aggressive 1/8 ratio, all baselines suffer severe quality degradation, exposing the inherent limitations of their 1D token-level or window-level eviction paradigms in capturing visual characteristics. In stark contrast, LineAR remains highly robust with minimal performance drops. Beyond achieving superior generation quality, LineAR maintains inference efficiency comparable to StreamingLLM. It incurs negligible overhead via lightweight, once-per-line eviction. Conversely, H2O lags behind the others in throughput due to the continuous computational burden of per-step attention accumulation.

Qualitative Comparison. We further present qualitative comparisons in Fig. 6. In contrast to the significant loss of detail and structural distortion exhibited by baseline methods under low budget ratios, LineAR demonstrates superior robustness. Across diverse scenarios, including portraits, landscapes, and intricate

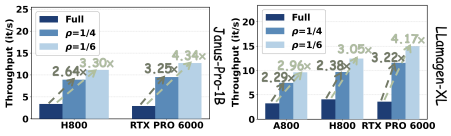

Fig. 7: Efficiency evaluation on different NVIDIA GPU devices.

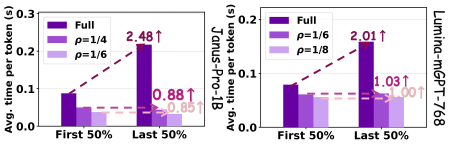

Fig. 8: Comparison of decoding speed in the first and last 50% of the image.

Table 4: Efficiency analysis on LlamaGen, Janus-Pro, and Lumina-mGPT models with lossless compression using LineAR, evaluated on NVIDIA RTX PRO 6000 GPUs.

Models	Method	ρ	Budget	Batch	Memory	Mem. Saving	Latency	Through.	Speedup
LlamaGen-XL ($N = 576$)	Full	1	576	256	63.64GB	-	71.27s	3.59it/s	1.00×
	LineAR	1/4	144	256	25.49GB	59.94%	22.14s	11.56it/s	3.22×
		1/6	96	256	20.96GB	67.06%	17.10s	14.97it/s	4.17×
Janus-Pro-1B ($N = 576$)	Full	1	576	256	60.25GB	-	87.36s	2.93it/s	1.00×
	LineAR	1/4	144	256	20.89GB	65.32%	26.88s	9.52it/s	3.25×
		1/6	96	256	15.88GB	73.65%	20.12s	12.72it/s	4.34×
Janus-Pro-7B ($N = 576$)	Full	1	576	128	85.00GB	-	111.49s	1.15it/s	1.00×
	LineAR	1/4	144	128	34.80GB	59.06%	36.25s	3.53it/s	3.08×
		1/6	96	128	28.68GB	66.26%	27.80s	4.60it/s	4.01×
Lumina-mGPT-768 ($N = 2352$)	Full	1	2352	18	76.69GB	-	309.66s	0.06it/s	1.00×
	LineAR	1/6	392	18	25.54GB	66.70%	154.69s	0.12it/s	2.00×
		1/8	294	18	22.87GB	70.18%	138.14s	0.13it/s	2.24×

objects, LineAR consistently synthesizes images with visual quality on par with the full-cache baseline. Notably, in portrait scenes, LineAR yields highly realistic and aesthetically pleasing results, proving its efficacy in maintaining strict structural integrity, global semantics, and textural fidelity, even under constrained memory budgets.

4.4 Efficiency Analysis

LineAR's efficient management yields substantial memory savings and robust inference speedups across diverse architectures and hardware devices.

Efficiency Evaluation across Architectures. Tab. 4 presents the results under lossless compression settings. Specifically, at a 1/6 budget ratio, LineAR achieves up to 67.06% memory savings on LlamaGen-XL while significantly increasing throughput by up to 4.17× compared to the full-cache baseline. Furthermore, on Janus-Pro-1B and Lumina-mGPT-768, LineAR yields 73.65% and 70.18% memory savings, respectively, accompanied by 4.34× and 2.24× speedups at 1/6 and 1/8 budget ratios, respectively.

Efficiency Evaluation across Devices. Fig. 7 illustrates the speedup ratios achieved by LineAR across three NVIDIA GPUs with distinct memory bandwidths and compute profiles: A800, H800, and RTX PRO 6000. Despite these architectural differences, LineAR consistently delivers around 3–4× inference ac-

Table 5: Component ablation on Janus-Pro-1B with $\rho = 1/6$.

$\hat{\mathbf{P}}_{\text{init}}$	$\hat{\mathbf{P}}_{\text{rec}}$	$\hat{\mathbf{P}}_{\text{mid}}$	FID-30k↓	CLIP↑	GenEval↑
	✔	✔	203.38	0.16	0.04
✔		✔	79.96	0.19	0.13
✔	✔		23.81	0.24	0.67
✔	✔	✔	**22.86**	**0.24**	**0.71**

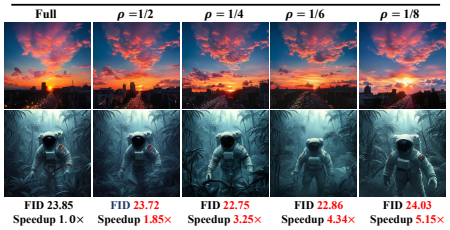

Fig. 9: Comparison results under different budget ratios.

Table 6: Ablation study of $\hat{\mathbf{P}}_{\text{mid}}$ selection strategies on Janus-Pro-1B. K/V-Sim, AttAcc, and IL Guid. denote K/V-Similarity, attention accumulation, and inter-line guided eviction, respectively.

LineAR	ρ	FID-30k↓	CLIP↑	GenEval↑
w. Random	1/6	34.34	0.19	0.36
w. K-Sim	1/6	23.23	0.24	0.68
w. V-Sim	1/6	23.57	0.24	0.67
w. AttAcc	1/6	23.27	0.24	0.68
w. IL Guid.	1/6	**22.86**	**0.24**	**0.71**
w. Random	1/8	46.62	0.17	0.22
w. K-Sim	1/8	24.18	0.24	0.63
w. V-Sim	1/8	24.98	0.24	0.62
w. AttAcc	1/8	24.30	0.24	0.61
w. IL Guid.	1/8	**24.03**	**0.24**	**0.65**

celeration across all devices on LlamaGen-XL and Janus-Pro-1B, confirming that the speedup stems from the reduced KV cache footprint rather than hardware-specific optimizations.

Stable Decoding Throughput. LineAR effectively mitigates the degradation of autoregressive decoding throughput over time. Fig. 8 compares per-token generation speeds between the first and second halves of decoding: while full-cache models suffer severe slowdowns as the KV cache accumulates, LineAR maintains highly stable speeds by strictly bounding the cache footprint. This constant throughput makes it exceptionally suited for real-world deployments.

4.5 Ablation Study

Impact of Compression Components. We evaluate each component of our progressive compression framework in Tab. 5. Removing either the initial anchors ($\hat{\mathbf{P}}_{\text{init}}$) or the recent lines' cache ($\hat{\mathbf{P}}_{\text{rec}}$) severely degrades quality, proving their necessity for global alignment and local spatial continuity. While keeping only these boundaries works acceptably, selectively retaining informative mid-region tokens ($\hat{\mathbf{P}}_{\text{mid}}$) crucially reduces FID-30k from 23.81 to 22.86. This confirms that adaptively compressing the mid-region via visual guidance is far superior to rigid truncation, validating our design tailored to visual generation characteristics.

Effectiveness of Inter-line Guided Eviction. To isolate the effectiveness of our Inter-line Guided Eviction (IL Guid.), we compare it against representative heuristics adapted for $\hat{\mathbf{P}}_{\text{mid}}$ selection under identical settings: Random, Key/Value-Similarity (K/V-Sim, inspired by R-KV and ADSA), and Attention Accumulation (AttAcc, from H2O). Tab. 6 presents the results. Although these baseline heuristics outperform random selection by preserving important information, they remain suboptimal. In contrast, IL Guid. demonstrates superior

performance across all budget ratios. By explicitly exploiting the spatial continuity prior of visual generation, our approach guarantees that the eviction process is fundamentally less harmful to subsequent lines, leading to a safer and more effective progressive compression.

Impact of Budget Ratios. Fig. 9 illustrates the performance across varying cache budgets. While moderate compression optimally removes attention redundancy to benefit both speed and FID scores, aggressive compression leads to a slight quality drop. Nevertheless, LineAR consistently maintains strong prompt alignment and preserves overall visual fidelity even under extreme constraints.

Additional Ablation Study. More ablations on hyperparameter sensitivity and more analysis are deferred to the *Appendix*.

5 Conclusion

We present LineAR, a novel, training-free KV cache compression method designed for efficient autoregressive image generation. LineAR manages the visual KV cache at the line level and enables lossless progressive compression through inter-line guidance. It allows autoregressive image generation with only a few lines cached, achieving lossless performance. Extensive experiments across multiple AR image generation models demonstrate its effectiveness.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grants 62595733, 62325109, and 62561160155; in part by the Research Grants Council of the Hong Kong Special Administrative Region, China, under Grant N_CityU1114/25; and in part by the Shanghai "Belt and Road" Young Scholar Exchange Grant under Grant 24510742000.

References

1. Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F.L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al.: Gpt-4 technical report. arXiv preprint arXiv:2303.08774 (2023) 1, 4
2. Anthropic: The claude 3 model family: Opus, sonnet, haiku (2024), https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf 4
3. Cai, Z., Xiao, W., Sun, H., Luo, C., Zhang, Y., Wan, K., Li, Y., Zhou, Y., Chang, L.W., Gu, J., et al.: R-kv: Redundancy-aware kv cache compression for training-free reasoning models acceleration. arXiv preprint arXiv:2505.24133 (2025) 4, 11
4. Chang, H., Zhang, H., Jiang, L., Liu, C., Freeman, W.T.: Maskgit: Masked generative image transformer. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2022) 4
5. Chen, X., Wu, Z., Liu, X., Pan, Z., Liu, W., Xie, Z., Yu, X., Ruan, C.: Janus-pro: Unified multimodal understanding and generation with data and model scaling. arXiv preprint arXiv:2501.17811 (2025) 4, 6, 9

6. Chen, Z., Fan, J., Yu, Z., Zhuang, B., Tan, M.: Frequency-aware autoregressive modeling for efficient high-resolution image synthesis. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (2025) 4
7. Dao, T.: FlashAttention-2: Faster attention with better parallelism and work partitioning. In: International Conference on Learning Representations (ICLR) (2024) 10
8. Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al.: The llama 3 herd of models. arXiv preprint arXiv:2407.21783 (2024) 4
9. Esser, P., Rombach, R., Ommer, B.: Taming transformers for high-resolution image synthesis. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 12873–12883 (2021) 4
10. Feng, Y., Lv, J., Cao, Y., Xie, X., Zhou, S.K.: Ada-kv: Optimizing kv cache eviction by adaptive budget allocation for efficient llm inference. arXiv preprint arXiv:2407.11550 (2024) 2, 4
11. Fu, Y., Cai, Z., Asi, A., Xiong, W., Dong, Y., Xiao, W.: Not all heads matter: A head-level kv cache compression method with integrated retrieval and reasoning. arXiv preprint arXiv:2410.19258 (2024) 2, 4
12. Ge, S., Zhang, Y., Liu, L., Zhang, M., Han, J., Gao, J.: Model tells you what to discard: Adaptive kv cache compression for llms. arXiv preprint arXiv:2310.01801 (2023) 4
13. Ghosh, D., Hajishirzi, H., Schmidt, L.: Geneval: An object-focused framework for evaluating text-to-image alignment. Advances in Neural Information Processing Systems pp. 52132–52152 (2023) 10
14. Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al.: Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. arXiv preprint arXiv:2501.12948 (2025) 4
15. Guo, H., Li, Y., Zhang, T., Wang, J., Dai, T., Xia, S.T., Benini, L.: Fastvar: Linear visual autoregressive modeling via cached token pruning. arXiv preprint arXiv:2503.23367 (2025) 4
16. Han, J., Liu, J., Jiang, Y., Yan, B., Zhang, Y., Yuan, Z., Peng, B., Liu, X.: Infinity: Scaling bitwise autoregressive modeling for high-resolution image synthesis. arXiv preprint arXiv:2412.04431 (2024) 4
17. He, W., Fu, S., Liu, M., Wang, X., Xiao, W., Shu, F., Wang, Y., Zhang, L., Yu, Z., Li, H., et al.: Mars: Mixture of auto-regressive models for fine-grained text-to-image synthesis. arXiv preprint arXiv:2407.07614 (2024) 2, 4
18. He, Y., Chen, F., He, Y., He, S., Zhou, H., Zhang, K., Zhuang, B.: Zipar: Parallel autoregressive image generation through spatial locality. In: Forty-second International Conference on Machine Learning 4
19. He, Y., He, Y., He, S., Chen, F., Zhou, H., Zhang, K., Zhuang, B.: Neighboring autoregressive modeling for efficient visual generation. arXiv preprint arXiv:2503.10696 (2025) 4
20. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. Advances in neural information processing systems **30** (2017) 10
21. Hu, X., Wang, R., Fang, Y., Fu, B., Cheng, P., Yu, G.: Ella: Equip diffusion models with llm for enhanced semantic alignment. arXiv preprint arXiv:2403.05135 (2024) 6, 10
22. Huang, Y., Chen, W., Zheng, W., Duan, Y., Zhou, J., Lu, J.: Spectralar: Spectral autoregressive visual generation. arXiv preprint arXiv:2506.10962 (2025) 4

23. Kang, H., Zhang, Q., Kundu, S., Jeong, G., Liu, Z., Krishna, T., Zhao, T.: Gear: An efficient kv cache compression recipe for near-lossless generative inference of llm. arXiv preprint arXiv:2403.05527 (2024) 4
24. Li, T., Chang, H., Mishra, S.K., Zhang, H., Katabi, D., Krishnan, D.: Mage: Masked generative encoder to unify representation learning and image synthesis. arXiv preprint arXiv:2211.09117 (2022) 4
25. Li, T., Tian, Y., Li, H., Deng, M., He, K.: Autoregressive image generation without vector quantization. arXiv preprint arXiv:2406.11838 (2024) 4
26. Li, Y., Huang, Y., Yang, B., Venkitesh, B., Locatelli, A., Ye, H., Cai, T., Lewis, P., Chen, D.: Snapkv: Llm knows what you are looking for before generation. arXiv preprint arXiv:2404.14469 (2024) 4
27. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision. pp. 740–755. Springer (2014) 10
28. Liu, A., Feng, B., Xue, B., Wang, B., Wu, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., et al.: Deepseek-v3 technical report. arXiv preprint arXiv:2412.19437 (2024) 1, 4
29. Liu, D., Zhao, S., Zhuo, L., Lin, W., Qiao, Y., Li, H., Gao, P.: Lumina-mgpt: Illuminate flexible photorealistic text-to-image generation with multimodal generative pretraining (2024), https://arxiv.org/abs/2408.02657 2, 4, 6, 9
30. Liu, Z., Desai, A., Liao, F., Wang, W., Xie, V., Xu, Z., Kyrillidis, A., Shrivastava, A.: Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. Advances in Neural Information Processing Systems **36** (2024) 4
31. Liu, Z., Yuan, J., Jin, H., Zhong, S., Xu, Z., Braverman, V., Chen, B., Hu, X.: Kivi: A tuning-free asymmetric 2bit quantization for kv cache. arXiv preprint arXiv:2402.02750 (2024) 4
32. Oren, M., Hassid, M., Adi, Y., Schwartz, R.: Transformers are multi-state rnns. arXiv preprint arXiv:2401.06104 (2024) 4
33. Pang, Z., Zhang, T., Luan, F., Man, Y., Tan, H., Zhang, K., Freeman, W.T., Wang, Y.X.: Randar: Decoder-only autoregressive visual generation in random orders. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 45–55 (2025) 4
34. Qin, Z., Cao, Y., Lin, M., Hu, W., Fan, S., Cheng, K., Lin, W., Li, J.: Cake: Cascading and adaptive kv cache eviction with layer preferences. arXiv preprint arXiv:2503.12491 (2025) 2, 4
35. Qin, Z., Lv, Y., Lin, M., Zhang, Z., Zou, D., Lin, W.: Head-aware kv cache compression for efficient visual autoregressive modeling. arXiv preprint arXiv:2504.09261 (2025) 4
36. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: International conference on machine learning. pp. 8748–8763. PmLR (2021) 10
37. Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., Sutskever, I.: Zero-shot text-to-image generation. In: International conference on machine learning. pp. 8821–8831. Pmlr (2021) 4
38. Ren, S., Zhu, K.Q.: On the efficacy of eviction policy for key-value constrained generative language model inference. arXiv preprint arXiv:2402.06262 (2024) 4
39. Ren, S., Yu, Y., Ruiz, N., Wang, F., Yuille, A., Xie, C.: M-var: Decoupled scale-wise autoregressive modeling for high-quality image generation. arXiv preprint arXiv:2411.10433 (2024) 4

40. Sun, P., Jiang, Y., Chen, S., Zhang, S., Peng, B., Luo, P., Yuan, Z.: Autoregressive model beats diffusion: Llama for scalable image generation. arXiv preprint arXiv:2406.06525 (2024) 2, 4, 10
41. Tang, H., Wu, Y., Yang, S., Xie, E., Chen, J., Chen, J., Zhang, Z., Cai, H., Lu, Y., Han, S.: Hart: Efficient visual generation with hybrid autoregressive transformer. arXiv preprint arXiv:2410.10812 (2024) 4
42. Team, C.: Chameleon: Mixed-modal early-fusion foundation models (2025), https://arxiv.org/abs/2405.09818 4
43. Teng, Y., Shi, H., Liu, X., Ning, X., Dai, G., Wang, Y., Li, Z., Liu, X.: Accelerating auto-regressive text-to-image generation with training-free speculative jacobi decoding. arXiv preprint arXiv:2410.01699 (2024) 4
44. Tian, K., Jiang, Y., Yuan, Z., Peng, B., Wang, L.: Visual autoregressive modeling: Scalable image generation via next-scale prediction. Advances in neural information processing systems **37**, 84839–84865 (2024) 4
45. Van Den Oord, A., Vinyals, O., et al.: Neural discrete representation learning. Advances in neural information processing systems **30** (2017) 4
46. Voronov, A., Kuznedelev, D., Khoroshikh, M., Khrulkov, V., Baranchuk, D.: Switti: Designing scale-wise transformers for text-to-image synthesis (2024) 4
47. Wan, Z., Wu, Z., Liu, C., Huang, J., Zhu, Z., Jin, P., Wang, L., Yuan, L.: Look-m: Look-once optimization in kv cache for efficient multimodal long-context inference. arXiv preprint arXiv:2406.18139 (2024) 4
48. Wang, J., Tian, Z., Wang, X., Zhang, X., Huang, W., Wu, Z., Jiang, Y.G.: Simplear: Pushing the frontier of autoregressive visual generation through pretraining, sft, and rl. arXiv preprint arXiv:2504.11455 (2025) 4
49. Wang, X., Zhang, X., Luo, Z., Sun, Q., Cui, Y., Wang, J., Zhang, F., Wang, Y., Li, Z., Yu, Q., et al.: Emu3: Next-token prediction is all you need. arXiv preprint arXiv:2409.18869 (2024) 2, 4
50. Wang, Y., Ren, S., Lin, Z., Han, Y., Guo, H., Yang, Z., Zou, D., Feng, J., Liu, X.: Parallelized autoregressive visual generation. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 12955–12965 (2025) 4
51. Wu, C., Chen, X., Wu, Z., Ma, Y., Liu, X., Pan, Z., Liu, W., Xie, Z., Yu, X., Ruan, C., et al.: Janus: Decoupling visual encoding for unified multimodal understanding and generation. arXiv preprint arXiv:2410.13848 (2024) 4
52. Wu, X., Hao, Y., Sun, K., Chen, Y., Zhu, F., Zhao, R., Li, H.: Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis. arXiv preprint arXiv:2306.09341 (2023) 10
53. Xiang, X., Fan, Q.: Make it efficient: Dynamic sparse attention for autoregressive image generation (2025), https://arxiv.org/abs/2506.18226 2, 6, 11
54. Xiao, G., Tian, Y., Chen, B., Han, S., Lewis, M.: Efficient streaming language models with attention sinks. arXiv preprint arXiv:2309.17453 (2023) 2, 4, 11
55. Xin, Y., Yan, J., Qin, Q., Li, Z., Liu, D., Li, S., Huang, V.S.J., Zhou, Y., Zhang, R., Zhuo, L., et al.: Lumina-mgpt 2.0: Stand-alone autoregressive image modeling. arXiv preprint arXiv:2507.17801 (2025) 4
56. Yan, F., Wei, Q., Tang, J., Li, J., Wang, Y., Hu, X., Li, H., Zhang, L.: Lazymar: Accelerating masked autoregressive models via feature caching. arXiv preprint arXiv:2503.12450 (2025) 4
57. Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al.: Qwen3 technical report. arXiv preprint arXiv:2505.09388 (2025) 4

58. Yang, D., Han, X., Gao, Y., Hu, Y., Zhang, S., Zhao, H.: Pyramidinfer: Pyramid kv cache compression for high-throughput llm inference. arXiv preprint arXiv:2405.12532 (2024) 4
59. Yu, J., Li, X., Koh, J.Y., Zhang, H., Pang, R., Qin, J., Ku, A., Xu, Y., Baldridge, J., Wu, Y.: Vector-quantized image modeling with improved vqgan. arXiv preprint arXiv:2110.04627 (2021) 4
60. Yue, X., Wang, Z., Wang, Y., Zhang, W., Liu, X., Ouyang, W., Bai, L., Zhou, L.: Understand before you generate: Self-guided training for autoregressive image generation. arXiv preprint arXiv:2509.15185 (2025) 4
61. Zhang, Z., Sheng, Y., Zhou, T., Chen, T., Zheng, L., Cai, R., Song, Z., Tian, Y., Ré, C., Barrett, C., et al.: H2o: Heavy-hitter oracle for efficient generative inference of large language models. Advances in Neural Information Processing Systems **36** (2024) 2, 4, 11
62. Zhang, Z., Huang, L.J., Wu, C., Yang, S., Peng, K., Lu, Y., Han, S.: Locality-aware parallel decoding for efficient autoregressive image generation. arXiv preprint arXiv:2507.01957 (2025) 4