AMS-KV: Adaptive KV Caching in Multi-Scale Visual Autoregressive Transformers

Boxun Xu, Yu Wang, Zihu Wang, Peng Li

University of California, Santa Barbara
{boxunxu, yu95, zihu_wang, lip}@ucsb.edu

Abstract

Visual autoregressive modeling (VAR) via next-scale prediction has emerged as a scalable image generation paradigm. While Key and Value (KV) caching in large language models (LLMs) has been extensively studied, next-scale prediction presents unique challenges, and KV caching design for next-scale based VAR transformers remains largely unexplored. A major bottleneck is the excessive KV memory growth with the increasing number of scales—severely limiting scalability. Our systematic investigation reveals that: (1) Attending to tokens from local scales significantly contributes to generation quality (2) Allocating a small amount of memory for the coarsest scales, termed as condensed scales, stabilizes multi-scale image generation (3) Strong KV similarity across finer scales is predominantly observed in cache-efficient layers, whereas cache-demanding layers exhibit weaker inter-scale similarity. Based on the observations, we introduce AMS-KV, a scale-adaptive KV caching policy for next-scale prediction in VAR models. AMS-KV prioritizes storing KVs from condensed and local scales, preserving the most relevant tokens to maintain generation quality. It further optimizes KV cache utilization and computational efficiency identifying cache-demanding layers through inter-scale similarity analysis. Compared to the vanilla next-scale prediction-based VAR models, AMS-KV reduces KV cache usage by up to 84.83% and self-attention latency by 60.48%. Moreover, when the baseline VAR-d30 model encounters out-of-memory failures at a batch size of 128, AMS-KV enables stable scaling to a batch size of 256 with improved throughput.

Introduction

Autoregressive (AR) models have demonstrated remarkable success in natural language processing (NLP), underpinning the development of large language models (LLMs). Scaling laws (Achiam et al. 2023; Kaplan et al. 2020) reveal that AR models benefit significantly from increased model size, larger datasets, and expanded computational resources, making them the foundation of the state-of-the-art NLP systems. However, their efficiency remains a critical challenge, particularly due to the large memory overhead introduced by storing key-value (KV) caches during inference.

Encouraged by the success of AR models in NLP, efforts have been made to extend AR modeling to vision tasks, leading to promising results in image and video generation. Among these, VQ-VAE (van den Oord, Vinyals, and Kavukcuoglu 2017) introduced quantized latent tokens to greatly improve computational efficiency over pixel-level AR modeling (van den Oord, Kalchbrenner, and Kavukcuoglu 2016; van den Oord et al. 2016; Chen et al. 2020). Follow-up works have enhanced quantization quality via hierarchical structures (Razavi, van den Oord, and Vinyals 2019), residual quantization (Lee et al. 2022), or generative adversarial frameworks (Esser, Rombach, and Ommer 2020). Nevertheless, token-level AR models have historically underperformed compared to diffusion models (Sohl-Dickstein et al. 2015; Song and Ermon 2019; Peebles and Xie 2023; Esser et al. 2024), which however require multiple expensive iterative sampling steps.

The recently developed visual autoregressive (VAR) models (Tian et al. 2024) have adopted GPT-style next-token prediction through next-scale prediction, enabling a new coarse-to-fine progressive image generation paradigm and surpassing diffusion-based generative models in performance. VAR models offer a more structured and controllable generative process (Tian et al. 2024; Xie et al. 2024; Chen et al. 2024b), and have been successfully generalized to tasks like text-to-image synthesis (Voronov et al. 2024; Zhang et al. 2024; Ma et al. 2025; Han et al. 2024), 3D reconstruction (Zhang, Xiong, and Xu 2024; Chen et al. 2025), and multi-modal generation (Qu et al. 2024). However, practical scalability of VAR remains a challenge.

Model efficiency has been extensively studied in NLP, with techniques such as pruning (Ma, Fang, and Wang 2023; Muralidharan et al. 2024; Zhu et al. 2024), quantization (Han, Mao, and Dally 2016; Lin et al. 2024; Gao et al. 2024; Rokh, Azarpeyvand, and Khanteymoori 2023), and cache optimization (Xiao et al. 2024; Zhang et al. 2023; Wu and Tu 2024; Sun et al. 2024; Zirui Liu et al. 2023; Hooper et al. 2024; Brandon et al. 2024; Liu et al. 2024; Chen et al. 2024a). In vision, techniques of improving diffusion model efficiency have also been well developed, including reducing the number of sampling steps (Luo et al. 2023; Salimans and Ho 2022) and accelerating solvers (Lu et al. 2022, 2023). However, similar optimizations for VAR remain relatively scarce (Xie et al. 2024; Chen et al. 2024b).

Notably, the progressive nature of VAR significantly extends the overall sequence length compared to conventional

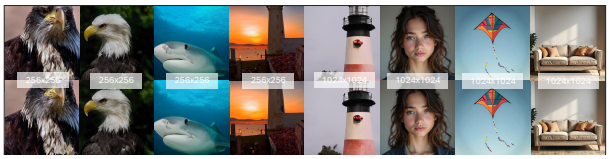

Figure 1: **Top:** The 256×256 images generated by VAR-d30(Tian et al. 2024) and 1024×1024 images generated Infinity-2B(Han et al. 2024). **Bottom:** generated using tuning-free AMS-KV with 4.7× less KV Cache Memory consumption.

AR models. This results in excessive memory usage, primarily due to the KV cache, a fundamental bottleneck limiting the scalability of VAR models. Unlike LLMs, which only maintain a single sequence of KV cache storage, VAR models operate across multiple scales, requiring progressively larger cache storage as the generation process unfolds. This multi-scale dependency significantly amplifies the KV cache footprint, making VAR models memory-intensive and challenging to scale to higher resolutions or more complex scenes.

However, addressing KV cache efficiency in VAR models has received little attention, despite its critical role in enabling scalable deployment. In this work, we bridge this gap by systematically investigating KV cache redundancy in VAR models and proposing an Adaptive Multi-Scale KV caching strategy, called AMS-KV, to significantly reduce memory consumption while maintaining model performance. Our main contributions are:

- We conduct in-depth studies of KV caching behavior across different **scales** and **layers** in VAR models, revealing key insights into non-uniform cache requirements, scale-wise importance and layer-wise cache preference.
- We propose a novel KV caching policy to dynamically allocate cache storage based on scale types across layers with heterogeneous cache demands. This enables substantial memory savings as shown in Figure 1 and supports larger batches while keeping generation quality.
- We perform extensive experiments and ablation studies to validate the performance of AMS-KV, demonstrating significantly improved memory efficiencies.

By addressing KV cache inefficiencies in VAR models, AMS-KV facilitates the scalable deployment of AR-based vision models, enabling broader applicability in real-world scenarios.

Related Works

Autoregressive Vision Modeling

Autoregressive (AR) models have achieved remarkable success in natural language processing (NLP), leading to the development of influential foundation models (Radford et al. 2019; Touvron and et al. 2023; Jiang and et al. 2023; DeepSeek-AI and et al. 2025). Recently, significant effort has been dedicated to adapting AR modeling to vision tasks.

Early attempts focused on pixel-level AR modeling (van den Oord, Kalchbrenner, and Kavukcuoglu 2016; van den Oord et al. 2016; Chen et al. 2020), demonstrating feasibility but facing scalability challenges in generating high-resolution images. To overcome this, VQ-VAE (van den Oord, Vinyals, and Kavukcuoglu 2017) introduced quantized latent tokens, greatly improving computational efficiency. Follow-up studies have enhanced quantization quality via hierarchical structures (Razavi, van den Oord, and Vinyals 2019), residual quantization (Lee et al. 2022), or generative adversarial frameworks (Esser, Rombach, and Ommer 2020). Parallel developments directly leverage transformer architectures for visual modeling (Dosovitskiy et al. 2020; Yu et al. 2021), opening opportunities for downstream tasks such as text-to-image generation (Yu et al. 2022; Ramesh et al. 2021; Li et al. 2024; He et al. 2024).

Despite these advancements, token-level AR models have typically underperformed diffusion models (Sohl-Dickstein et al. 2015; Song and Ermon 2019; Peebles and Xie 2023; Bao et al. 2023; Esser et al. 2024). Until recently, VAR (Tian et al. 2024) introduced a novel AR paradigm based on next-scale prediction, achieving superior performance compared to diffusion models. This approach has successfully generalized to tasks like text-to-image synthesis (Voronov et al. 2024; Zhang et al. 2024; Ma et al. 2025; Han et al. 2024), 3D reconstruction (Zhang, Xiong, and Xu 2024; Chen et al. 2025), and multi-modal generation (Qu et al. 2024).

Efficient Language and Vision Models

The success of autoregressive (AR) models in NLP has driven extensive research on scaling laws (Achiam et al. 2023; Kaplan et al. 2020), revealing that performance scales effectively with increased model size, data, and computation. However, the computational cost remains a major challenge for practical deployment.

To address redundancy in over-parameterized networks for language modeling, compression techniques such as pruning (Ma, Fang, and Wang 2023; Muralidharan et al. 2024; Zhu et al. 2024), quantization (Han, Mao, and Dally 2016; Lin et al. 2024; Gao et al. 2024; Rokh, Azarpeyvand, and Khanteymoori 2023), and knowledge distillation have been explored. For AR models, attention mechanisms introduce additional bottlenecks due to large key-value (KV) caches. Recent works mitigate this by discarding less critical cache entries (Xiao et al. 2024; Zhang et al. 2023; Wu and Tu 2024; Sun et al. 2024), applying quantization (Zirui Liu et al. 2023; Hooper et al. 2024), or enabling cache sharing across layers (Brandon et al. 2024; Liu et al. 2024; Chen et al. 2024a).

In vision models, research primarily focuses on diffusion-based architectures by optimizing sampling efficiency via reduced steps (Luo et al. 2023; Salimans and Ho 2022) and faster solvers (Lu et al. 2022, 2023). In contrast, efficiency improvements for VAR models remain largely underexplored, with only a few initial efforts (Xie et al. 2024; Chen et al. 2024b).

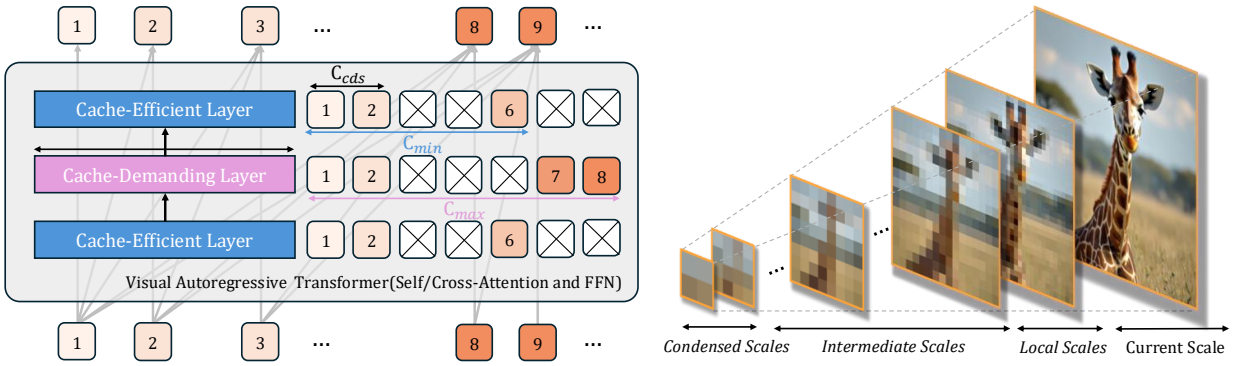

Figure 2: Overview of Adaptive Multi-Scale KV Caching (AMS-KV) for Visual Autoregressive Modeling.

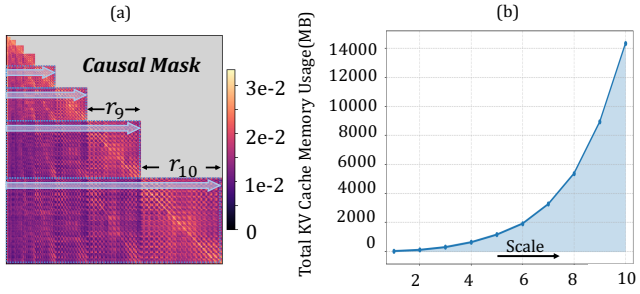

Figure 3: (a) The growth of attention map. (b) Memory usage of VAR across scales during unconditional image generation on an NVIDIA A100-80G (Batch Size=50)

Preliminaries

Visual Autoregressive Modeling (VAR)

Visual Autoregressive (VAR) models build upon the principles introduced by VQ-VAE (van den Oord, Vinyals, and Kavukcuoglu 2017), but further extends the standard next-token prediction approach to next-scale prediction. Specifically, an input image is first encoded into a multi-scale hierarchy of discrete tokens using a Vector-Quantized Variational Autoencoder (VQ-VAE):

$$f = \mathcal{E}(im), \quad R := (r_1, r_2, \ldots, r_K) = \mathcal{Q}(f), \qquad (1)$$

where $f \in \mathbb{R}^{h \times w \times d}$ denotes the encoded feature map. Each discrete scale $r^k \in [V]^{h^k \times w^k}$[1] is obtained through a residual quantizer $\mathcal{Q}$ (Lee et al. 2022) that utilizes a shared codebook.

A decoder-only transformer architecture factorizes the joint likelihood of these scales autoregressively:

$$p(r_1, r_2, \ldots, r_K) = \prod_{k=1}^{K} p(r_k \mid r_{<k}), \qquad (2)$$

which sequentially predicts coarse scales with smaller spatial dimensions to larger fine-grained scales. This next scale prediction allows VAR to progressively capture information from general structures to finer details, and achieves outstanding performance in visual representation learning and image synthesis tasks.

[1]For simplicity, we let $h^k = w^k$ in later sections, e.g., we abbreviate the spatial size of a scale of 4×4 to 4.

Memory Bottleneck in Efficient VAR

Despite its strong performance, VAR models encounter significant memory bottlenecks during next-scale prediction. Standard VAR implementations typically use scales with spatial dimensions that grow non-linearly, such as $[1 \times 1], [2 \times 2], \ldots, [16 \times 16]$. This quadratic increase in spatial dimension results in rapidly growth of computational complexity for the attention mechanism, as illustrated in Figure 3 (a). To mitigate this complexity, VAR adopts the Key-Value (KV) cache technique, widely used in large language models (LLMs), which efficiently caches previously computed key and value matrices to avoid redundant computations (Touvron and et al. 2023). Nevertheless, due to the quadratic growth of each subsequent scale's spatial size, the size of newly computed keys and values also expands quadratically, ultimately resulting in a cubic growth of the overall KV cache size. This rapid growth can quickly surpass memory limitations, particularly during predictions at later scales, as illustrated in Figure 3(b).

Methodology

The expanding KV cache in VAR models presents a significant computational and memory challenge. This necessitates an efficient KV cache management strategy that can prune redundant information without compromising generation quality. A critical step toward such a strategy is to understand the roles of different scales existing in different layers. We have observed that due to the sequential nature of scale generation in VAR, a query at a certain stage often exhibits similar attention patterns across multiple distinct scales as shown in Figure 3. This suggests a high degree of semantic similarity, particularly in later stages, and reveals substantial redundancy in the information stored across scales.

To address this, we identify two key properties: (1) Not all scales need to be stored and they do not contribute equally to generation, a property we refer to as **scale-wise importance**. (2) Storing even the most important scales is still challenging due to the quadratic size complexity. However, we ob-

serve that KV cache requirements are heterogeneous across different layers. We term this as **layer-dependent cache preference** and classify layers as either cache-efficient or cache-demanding.

Not All Scales Are Equally Important

Inspired by the notion of token importance in LLMs, we hypothesize that different scales contribute unequally to the generation process. To analyze this non-uniformity, we examine the cross-scale attention—the extent to which the final scale (e.g., r_{10} in VAR models) attends to tokens from preceding scales across different heads. We observe that scales influence the generation process differently, contributing unevenly to next-scale predictions.

As we concern scale-wise patterns and token counts vary across different scales, directly comparing raw accumulated cross-scale attention values for each previous scale is not meaningful. To account for this, we introduce *attention density*, defined as the average cross-scale attention score per token within a previous scale, to compare how attention is distributed across scales.

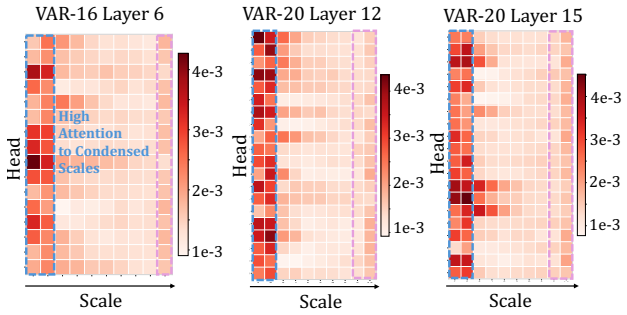

Figure 4: Visualization of the attention density across heads and scales when generating the last scale r_{10}.

Interestingly, as illustrated in Figure 4, VAR models exhibit a characteristic pattern: the initial scales, despite having fewer tokens, exhibit the highest attention density. This implies that storing such scales are most high cost performance. The attention density decreases across intermediate scales before rising again at the local scales. This trend suggests that both early and late scales play a critical role in generation, while intermediate scales contribute less significantly. Based on this observation, we categorize scales into three groups: Condensed scales, referring to the first two scales; Local scales, comprising the most recent scales preceding the current scale; and Intermediate scales which include all remaining scales not covered by the above two groups.

To further assess the impact of these scale groups, we conduct an experiment that selectively removes KV caches for different scale groups during image generation. The results reveal distinct effects, as shown in Figure 5. Firstly, condensed scales serve as an anchor for modeling the global structural foundation, removing these scales disrupts the overall composition of the generated images. Secondly, local scales refine the details. Without them, synthesized images become blurry and lack sharpness. Finally, intermediate scales have minimal impact when excluded, indicating redundancy in the KV cache for these scales. We also compare the image generation quality between the above three settings in Table 1, which further conforms our conclusion.

Config.	FID↓	IS↑	Precision↑	Recall↑
Baseline(No removal)	2.07	336.15	0.82	0.58
Remove Intermediate	2.12	329.60	0.82	0.59
Remove Local	3.04	295.14	0.76	0.60
Remove Condensed	2.48	298.35	0.80	0.59

Table 1: Comparison of Generation Quality When Removing KV Caches from Different Scales.

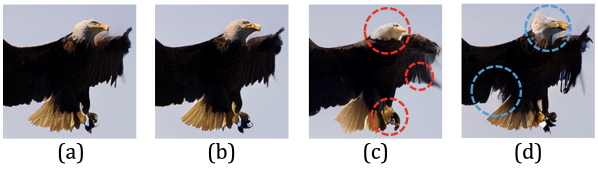

Figure 5: An eagle generated by VAR-d30 with (a) full KV cache; (b) intermediate scales removed; (c) local scales removed; (d) condensed scales removed. Red circles highlight missing fine details; blue circles indicate distorted regions.

These findings highlight that memory allocation for KV caches in VAR models can be optimized by prioritizing the condensed and local scales while reducing storage for intermediate scales, thereby reducing overall memory overhead without degrading generation quality.

Not All Layers are Cache-Demanding

In decoder-only transformer architectures with KV cache support (Touvron and et al. 2023; Han et al. 2024), the cache budget is typically allocated uniformly across all decoder layers—each layer is assigned the same maximum cache size. However, this default strategy is suboptimal for VAR models. Our analysis reveals that cache-demanding layers—those with low inter-scale KV similarity—play a critical role in feature expansion and structural propagation, and thus require larger KV caches to preserve essential generation context. In contrast, cache-efficient layers exhibit high KV similarity across scales and mainly perform refinement over already-formed representations, making them more resilient to reduced cache allocation.

Moreover, uniformly storing local scales across all layers becomes impractical, as the token count within each scale grows quadratically. Under a fixed total cache budget, we observe that not all layers should be treated equally in cache allocation. Specifically, prioritizing cache resources for cache-demanding layers, to store more local scales, helps preserve generation quality. In contrast, assigning minimal cache to cache-efficient layers, limiting them to a few local or even sub-local scales can further reduce memory usage without harming performance. Here, "sub-local" scales refer

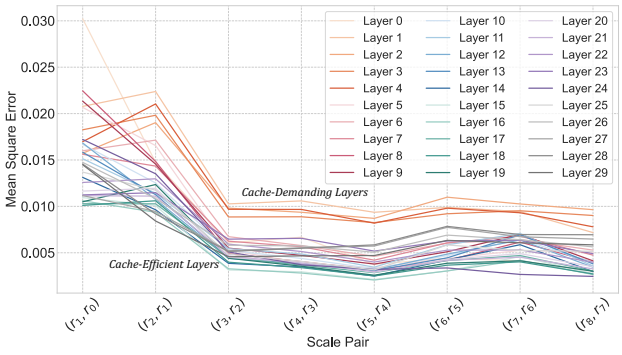

Figure 6: Discrepancy between adjacent-scale keys across layers in the VAR-d30 model.

to earlier scales that are even smaller than the local scales. This motivates a non-uniform, similarity-aware cache allocation strategy that aligns cache resources with the functional roles of different decoding layers in VAR models.

To validate the observation, we conduct an illustrative experiment under a constrained total cache budget. This total budget allows one-sixth of layers assigned with a large layer cache capacity of $C_{\max}$, sufficient to store the two most recent scales and condensed scales while assigning a smaller layer capacity of $C_{\min}$ to the remaining five-sixths layers, which can exactly store the second most recent and condensed scales. These caches are distributed across the decoding layers of a VAR-d30 model. We compare the following two cache allocation strategies under the same total cache budget with the same number of layers using larger caches as described above. In the uniform allocation, `S1`, the layers with larger caches are uniformly distributed across model depth. In contrast, in the similarity-aware allocation, `S2`, the larger caches are assigned to the layers exhibiting the lowest inter-scale similarity, as defined in Equation 3 and identified in Figure 6.

We evaluate the impact of these two strategies on image generation quality as summarized in Table 2. The results further confirm that low-similarity layers, as cache-demanding layers, are crucial for feature construction and require larger cache capacity. Reducing the cache size in these layers leads to a significant degradation in performance, whereas cache-efficient layers exhibit greater robustness to cache compression.

Policy	FID↓	IS↑	Prec.↑	Rec.↑
`S1` (Uniform)	6.65	223.65	0.71	0.60
`S2` (Sim-aware)	3.37	267.71	0.76	0.61

Table 2: Comparison of generation quality under two allocation strategies, `S1` (uniform) and `S2` (similarity-aware).

`AMS-KV`: Adaptive Multi-Scale KV Caching

The proposed KV caching policy,`AMS-KV`, is motivated by the aforementioned observations of scale-wise importance and layer-dependent cache preference. First, `AMS-KV` retains the condensed scales while rolling local scales. Second, it employs a similarity-driven approach to classify layers into **cache-demanding** and **cache-efficient** layers, enabling cache-demanding layers to store larger and more local scales.

As illustrated in Figure 2, each layer's KV cache in `AMS-KV` operates independently and is governed by three key hyperparameters: $C_{\min}$, $C_{\max}$ and C_{cds}. Specifically, $C_{\max}$ represents the expanded capacity allocated to cache-demanding layers, which is the maximum cache size for a layer; $C_{\min}$ defines the reserved capacity of cache-efficient layers which is the minimum cache size for a layer; C_{cds} specifies the number of condensed scales retained within the KV cache during rolling.

The update rule is as illustrated in Algorithm 1. Each layer is initialized as cache-efficient layer with its cache budget C_{bgt} set to $C_{\min}$. During finer-grained rolling of KV pairs, the KV cache always prioritizes storing the condensed scales, which are locked due to their high cross-scale attention, as illustrated in Figure 4. In addition, the KV cache dynamically appends local finer scales by evicting one or multiple older intermediate scales. To manage this process, we introduce the Condensed Least Recently Used (`CLRU`) policy, a FIFO-based eviction strategy that removes older scales while ensuring that condensed scales are retained, thereby maintaining the cache size within the budget C_{bgt}.

If the initial cache budget $C_{bgt} = C_{\min}$ is exhausted, `AMS-KV` invokes a similarity-guided mechanism to determine whether the current layer should be classified as a **cache-demanding** layer. To quantify inter-scale similarity, the keys from the $(i-1)$-th scale, denoted as k_{i-1}, are resized via 2D interpolation to match the spatial resolution of k_i, resulting in $\hat{k}_{i-1}$. The similarity score is then computed as the negative ℓ_2 distance:

$$\mathrm{S}(k_i, \hat{k}_{i-1}) = -\|k_i - \hat{k}_{i-1}\|_2. \tag{3}$$

If this similarity falls below a predefined threshold θ, the layer is classified as cache-demanding.

The cache extends its cache budget C_{bgt} only when two conditions are satisfied. First, the current cache size budget C_{bgt} can be further increased to $C_{\max}$. Second, the layer is identified as a cache-demanding layer. If the both conditions hold, the layer's cache size budget is expanded from $C_{\min}$ to $C_{\max}$ by allocating the additional memory $(C_{\max} - C_{\min})$. Otherwise, the cache budget remains fixed at $C_{\min}$.

In this manner, the cache-efficient layers in `AMS-KV` are processed with $C_{\min}$, while the detected cache-demanding layers are assigned a cache capacity of $C_{\max}$, particularly as the cache size of scale r_i increases.

Experiments

End-to-End Results

Experimental Setup We conduct experiments using pretrained Visual Autoregressive (VAR) models ($d = 30, 36$) obtained from (Tian et al. 2024) and Infinity-2B(Han et al. 2024). The primary objective is to compare the image generation performance of standard VAR models against those

Model Config.	Image Res.	#Para.	#Dim	FID↓	IS↑	Prec.↑	Rec.↑	KV Cache Size
VAR-d30(Tian et al. 2024)	256x256	2.0B	1920	2.07	336.15	0.82	0.58	22.41GB
VAR-d30[1](Tian et al. 2024)	256x256	2.0B	1920	2.03	305.19	0.82	0.59	22.41GB
VAR-d30 (+AMS-KV)	256x256	2.0B	1920	2.09	330.50	0.82	0.58	4.77GB (78.72%↓)
VAR-d30[1](+AMS-KV)	256x256	2.0B	1920	2.09	301.19	0.82	0.58	4.77GB (78.72%↓)
VAR-d36[2](Tian et al. 2024)	512x512	2.4B	2304	3.26	331.13	0.82	0.34	53.15GB
VAR-d36(Tian et al. 2024)	512x512	2.4B	2304	-	-	-	-	OOM$^{\diamondsuit}$
VAR-d36[2] (+AMS-KV)	512x512	2.4B	2304	3.31	331.18	0.82	0.34	15.26GB (71.29%↓)
VAR-d36 (+AMS-KV)	512x512	2.4B	2304	-	-	-	-	30.52GB

Table 3: Performance Comparison between different model configurations in VAR model families. [1] indicates a configuration of CFG=1.5 and top-k=900; [2] indicates a batch size of 25 on specific models. $\diamondsuit$ denotes an approximately KV Cache requirement of 117GB, which causes Out-of-Memory(OOM).

Algorithm 1: Adaptive Multi-Scale KV Caching

procedure CACHE($\{(k_i, v_i)\}_{i=1}^{N_{\text{scales}}}, C_{\min}, C_{\max}, C_{\text{cds}}$)
  $\mathcal{C}_0 \leftarrow \emptyset$
  $C_{bgt} \leftarrow C_{\min}$ ▷ Initialized as Cache-Efficient layers
  for $i = 1 \rightarrow N_{\text{scales}}$ **do**
    if $|(k_i, v_i)| > C_{\max}$ **then**
      Continue
    end if
    $\mathcal{C}_i \leftarrow \mathcal{C}_{(i-1)} \cup \{(k_i, v_i)\}$ ▷ Append i-th scale
    if $|\mathcal{C}_i| > C_{bgt}$ **then**
      if $C_{bgt} == C_{\min}$ & $\text{S}(k_i, \hat{k}_{(i-1)}) < \theta$ **then**
        $C_{bgt} \leftarrow C_{\max}$ ▷ Expand Cache Budget
      end if
      if $|(k_i, v_i)| > C_{bgt}$ **then**
        $\mathcal{C}_i \leftarrow \mathcal{C}_{(i-1)}$ and Continue
      end if
      $\mathcal{C}_i \leftarrow \text{CLRU}(\mathcal{C}_i, C_{bgt}, C_{\text{cds}})$ ▷ Rolling
    end if
  end for
end procedure

integrated with AMS-KV, focusing on both visual quality, memory utilization and computation efficiency. Following (Tian et al. 2024), we use top-k=600 and CFG=2.0 by default, ensuring a balance between diversity and coherence in generated samples. Results with top-k=900 and CFG=1.5 are explicitly noted when used. We set the capacity of cache-demanding layers, $C_{\max}$ to exactly equal the size of the last two scales and condensed scales, and the capacity of cache-efficient layers, $C_{\min}$, to the size of the penultimate scale and condensed scales. This configuration enables a systematic analysis of the trade-off between memory consumption and generation quality. Meanwhile, we set C_{cds} as the first two scales.

Evaluation Metrics The evaluation is conducted on the ImageNet-1K 256×256 and 512×512 class-conditioned generation, and object-focused benchmarking framework GenEval(Ghosh, Hajishirzi, and Schmidt 2023). We adopt standard metrics to measure generation quality: Fréchet Inception Distance (FID) for measuring distributional alignment with real images, Inception Score (IS) for evaluating sample realism and diversity, and Precision and Recall to quantify the trade-off between fidelity and coverage. Memory usage and computation latency is measured on a single NVIDIA A100 (80GB) GPU, and KV cache storage requirements is obtained with PyTorch CUDA profiling tools. AMS-KV is compatible with FlashAttention (Dao et al. 2022), so we keep using FlashAttention-2 as default with all baselines.

Results and Analysis Table 3 presents a comparison of modeling performance and KV cache utilization between the baseline model and AMS-KV across different model sizes. The batch size is fixed at 50 by default. We can observe that AMS-KV significantly reduces memory consumption while preserving high generation quality with minimal impact(+0.02 to +0.06 FID) on key evaluation metrics under different configurations. Across different VAR model variants, it achieves a reduction in KV cache utilization ranging from 71.29% to 78.72% (3.48× to 4.70× compression). Notably, it enables inference for the largest VAR-d36 model, which would otherwise exceed the memory capacity of a single GPU under standard caching strategies.

These results confirm that AMS-KV is a highly effective solution for reducing memory overhead while preserving generation quality, making it particularly advantageous for scaling autoregressive vision models within limited hardware constraints.

	Two Obj.↑	Position↑	Color Attr.↑	Overall↑
baseline	**0.824**	0.277	0.592	0.711
+AMS-KV	0.821	**0.278**	**0.608**	**0.714**

Table 4: Evaluation of AMS-KV on the Text-to-Image Task Using Infinity-2B on GenEval

	SWA	H2O	STA	AMS-KV
FID↓	11.61	8.80	2.81	**2.09**
IS↑	169.94	182.8	276.92	**323.7**

Table 5: Comparison with other KV cache strategies.

Robustness of AMS-KV To demonstrate the robustness of AMS-KV beyond intuitive hyperparameters, we explore a broader range of configurations for the tunable parameters $C_{\min}$ and $C_{\max}$. We perform ablation studies by sweeping across valid combinations, as shown in Figure 7, and observe a variety of memory–performance trade-offs. Notably, most configurations achieve substantial memory savings with minimal generation quality degradation, suggesting significant redundancy in VAR and robustness of AMS-KV.

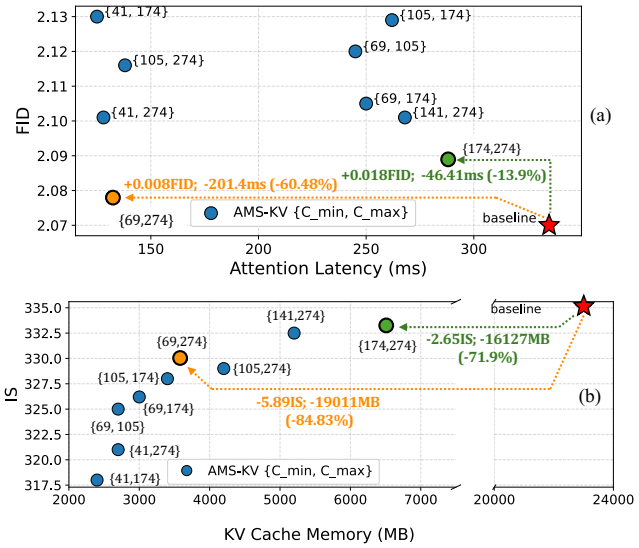

Figure 7: Robustness of AMS-KV across different cache size configurations. (a) FID vs. Attention Latency; (b) IS vs. KV Cache Memory Usage.

Evaluation on Infinity-2B We evaluate AMS-KV on Infinity-2B as shown in Table 4. AMS-KV reduces memory consumption from 28.12GB to 18.02GB(35.6% ↓) and increases throughput from 0.826 images/s to 0.885 images/s, while slightly improving overall generation quality from 0.711 to 0.714.

Comparison with Other KV Cache work We compare the generation performance of a KV cache strategy for LLMs under a fixed compression ratio of 75%. As shown in Table 5, AMS-KV significantly has better generation quality at the same memory budget over SWA (Beltagy, Peters, and Cohan 2020), H2O (Zhang et al. 2023), and STA (Xiao et al. 2024)). Meanwhile, AMS-KV operates independently of attention scores, making it fully compatible with memory-efficient attention kernels.

Reduced Memory Footprint with AMS-KV

We further analyze the memory footprint of AMS-KV compared to the baseline model as batch size increases—an important factor for the practical deployment of VAR models. Using the same setup as in Section , we evaluate the VAR-d30 model with batch sizes ranging from 16 to 256. In addition to reporting the KV cache memory footprint, we also measure the change in throughput, defined as the number of images generated per second (img/s).

Batch Size	Model	KV Mem.	Thr.(img/s)
16	VAR-d30	7.17GB	18.73
16	+AMS-KV	2.81GB	18.86
32	VAR-d30	14.34GB	21.07
32	+AMS-KV	5.62GB	21.10
64	VAR-d30	28.69GB	22.21
64	+AMS-KV	11.23GB	22.32
128	VAR-d30	OOM	OOM
128	+AMS-KV	22.47GB	23.00
256	VAR-d30	OOM	OOM
256	+AMS-KV	44.94GB	23.88

Table 6: The Optimization of VAR-d30 Memory Footprints and Throughput Across Batch Sizes.

Results and Analysis: Table 6 shows that as batch size increases, the KV cache size grows rapidly and eventually dominates overall memory utilization. Notably, at batch sizes of 128 and 256, the baseline model of VAR-d30 encounters Out-Of-Memory(OOM) failure, making inference infeasible. In contrast, AMS-KV significantly reduces KV cache requirements, allowing inference under 4× batch size.

Despite the substantial memory savings, AMS-KV yields only a modest throughput improvement of 7.5%. This is expected, as the attention computation latency is not dominant factors compared to FFNs in VAR models, and thus does not significantly constrain overall throughput as memory usage does. However, the ability of AMS-KV to support inference at high batch sizes without memory overflows highlights its practical advantage for real-world deployment, where large-batch processing is often essential for efficiency.

Conclusion

We address the challenge of KV caching inefficiency in visual autoregressive models, a major bottleneck for scalability. While VAR models achieve state-of-the-art generative performance with low latency, their growing KV cache footprint limits practical deployment. To mitigate this, we propose AMS-KV, an adaptive multi-scale KV caching strategy that significantly reduces memory consumption while preserving model performance. Experiments show that AMS-KV reduces KV cache utilization by up to 78.72%, enabling inference at 4 × batch sizes and preventing OOM failures. Despite this reduction, it maintains highly competitive image generation quality. Our analysis further reveals that early layers require larger caches, while later layers tolerate compression, offering valuable insights into efficient cache management. By alleviating KV cache overhead, AMS-KV enhances the scalability of VAR models, facilitating deployment in memory-constrained settings. We hope this work inspires future research on optimizing AR-based vision models for improved efficiency and scalability.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grants No. 2310170.

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Bao, F.; Nie, S.; Xue, K.; Cao, Y.; Li, C.; Su, H.; and Zhu, J. 2023. All are Worth Words: A ViT Backbone for Diffusion Models. arXiv:2209.12152.

Beltagy, I.; Peters, M. E.; and Cohan, A. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Brandon, W.; Mishra, M.; Nrusimha, A.; Panda, R.; and Kelly, J. R. 2024. Reducing Transformer Key-Value Cache Size with Cross-Layer Attention. arXiv:2405.12981.

Chen, M.; Radford, A.; Child, R.; Wu, J.; Jun, H.; Luan, D.; and Sutskever, I. 2020. Generative Pretraining From Pixels. In III, H. D.; and Singh, A., eds., *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, 1691–1703. PMLR.

Chen, Q.; Wang, W.; Zhang, Q.; Zheng, S.; Zhang, S.; Deng, C.; Yu, H.; Liu, J.; Ma, Y.; and Zhang, C. 2024a. Skip-Layer Attention: Bridging Abstract and Detailed Dependencies in Transformers. arXiv:2406.11274.

Chen, Y.; Lan, Y.; Zhou, S.; Wang, T.; and Pan, X. 2025. SAR3D: Autoregressive 3D Object Generation and Understanding via Multi-scale 3D VQVAE. In *CVPR*.

Chen, Z.; Ma, X.; Fang, G.; and Wang, X. 2024b. Collaborative Decoding Makes Visual Auto-Regressive Modeling Efficient. arXiv:2411.17787.

Dao, T.; Fu, D.; Ermon, S.; Rudra, A.; and Ré, C. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35: 16344–16359.

DeepSeek-AI; and et al. 2025. DeepSeek-V3 Technical Report. arXiv:2412.19437.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *CoRR*, abs/2010.11929.

Esser, P.; Kulal, S.; Blattmann, A.; Entezari, R.; Müller, J.; Saini, H.; Levi, Y.; Lorenz, D.; Sauer, A.; Boesel, F.; Podell, D.; Dockhorn, T.; English, Z.; Lacey, K.; Goodwin, A.; Marek, Y.; and Rombach, R. 2024. Scaling Rectified Flow Transformers for High-Resolution Image Synthesis. arXiv:2403.03206.

Esser, P.; Rombach, R.; and Ommer, B. 2020. Taming Transformers for High-Resolution Image Synthesis. *CoRR*, abs/2012.09841.

Gao, S.; Lin, C.-H.; Hua, T.; Tang, Z.; Shen, Y.; Jin, H.; and Hsu, Y.-C. 2024. DISP-LLM: Dimension-Independent Structural Pruning for Large Language Models. In Globerson, A.; Mackey, L.; Belgrave, D.; Fan, A.; Paquet, U.; Tomczak, J.; and Zhang, C., eds., *Advances in Neural Information Processing Systems*, volume 37, 72219–72244. Curran Associates, Inc.

Ghosh, D.; Hajishirzi, H.; and Schmidt, L. 2023. Geneval: An object-focused framework for evaluating text-to-image alignment. *Advances in Neural Information Processing Systems*, 36: 52132–52152.

Han, J.; Liu, J.; Jiang, Y.; Yan, B.; Zhang, Y.; Yuan, Z.; Peng, B.; and Liu, X. 2024. Infinity: Scaling Bitwise AutoRegressive Modeling for High-Resolution Image Synthesis. arXiv:2412.04431.

Han, S.; Mao, H.; and Dally, W. J. 2016. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. arXiv:1510.00149.

He, W.; Fu, S.; Liu, M.; Wang, X.; Xiao, W.; Shu, F.; Wang, Y.; Zhang, L.; Yu, Z.; Li, H.; Huang, Z.; Gan, L.; and Jiang, H. 2024. MARS: Mixture of Auto-Regressive Models for Fine-grained Text-to-image Synthesis. arXiv:2407.07614.

Hooper, C.; Kim, S.; Mohammadzadeh, H.; Mahoney, M. W.; Shao, S.; Keutzer, K.; and Gholami, A. 2024. KVQuant: Towards 10 Million Context Length LLM Inference with KV Cache Quantization. In Globerson, A.; Mackey, L.; Belgrave, D.; Fan, A.; Paquet, U.; Tomczak, J.; and Zhang, C., eds., *Advances in Neural Information Processing Systems*, volume 37, 1270–1303. Curran Associates, Inc.

Jiang, A. Q.; and et al. 2023. Mistral 7B. arXiv:2310.06825.

Kaplan, J.; McCandlish, S.; Henighan, T.; Brown, T. B.; Chess, B.; Child, R.; Gray, S.; Radford, A.; Wu, J.; and Amodei, D. 2020. Scaling Laws for Neural Language Models. arXiv:2001.08361.

Lee, D.; Kim, C.; Kim, S.; Cho, M.; and Han, W.-S. 2022. Autoregressive Image Generation using Residual Quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11523–11532.

Li, H.; Yang, J.; Wang, K.; Qiu, X.; Chou, Y.; Li, X.; and Li, G. 2024. Scalable Autoregressive Image Generation with Mamba. arXiv:2408.12245.

Lin, J.; Tang, J.; Tang, H.; Yang, S.; Chen, W.-M.; Wang, W.-C.; Xiao, G.; Dang, X.; Gan, C.; and Han, S. 2024. AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration. arXiv:2306.00978.

Liu, A.; Liu, J.; Pan, Z.; He, Y.; Haffari, G.; and Zhuang, B. 2024. MiniCache: KV Cache Compression in Depth Dimension for Large Language Models. arXiv:2405.14366.

Lu, C.; Zhou, Y.; Bao, F.; Chen, J.; Li, C.; and Zhu, J. 2022. DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps. arXiv:2206.00927.

Lu, C.; Zhou, Y.; Bao, F.; Chen, J.; Li, C.; and Zhu, J. 2023. DPM-Solver++: Fast Solver for Guided Sampling of Diffusion Probabilistic Models. arXiv:2211.01095.

Luo, S.; Tan, Y.; Huang, L.; Li, J.; and Zhao, H. 2023. Latent Consistency Models: Synthesizing High-Resolution Images with Few-Step Inference. arXiv:2310.04378.

Ma, X.; Fang, G.; and Wang, X. 2023. LLM-Pruner: On the Structural Pruning of Large Language Models. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems*, volume 36, 21702–21720. Curran Associates, Inc.

Ma, X.; Zhou, M.; Liang, T.; Bai, Y.; Zhao, T.; Li, B.; Chen, H.; and Jin, Y. 2025. STAR: Scale-wise Text-conditioned AutoRegressive image generation. arXiv:2406.10797.

Muralidharan, S.; Sreenivas, S. T.; Joshi, R.; Chochowski, M.; Patwary, M.; Shoeybi, M.; Catanzaro, B.; Kautz, J.; and Molchanov, P. 2024. Compact Language Models via Pruning and Knowledge Distillation. arXiv:2407.14679.

Peebles, W.; and Xie, S. 2023. Scalable Diffusion Models with Transformers. arXiv:2212.09748.

Qu, L.; Zhang, H.; Liu, Y.; Wang, X.; Jiang, Y.; Gao, Y.; Ye, H.; Du, D. K.; Yuan, Z.; and Wu, X. 2024. Tokenflow: Unified image tokenizer for multimodal understanding and generation. *arXiv preprint arXiv:2412.03069*.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language Models are Unsupervised Multitask Learners.

Ramesh, A.; Pavlov, M.; Goh, G.; Gray, S.; Voss, C.; Radford, A.; Chen, M.; and Sutskever, I. 2021. Zero-Shot Text-to-Image Generation. *CoRR*, abs/2102.12092.

Razavi, A.; van den Oord, A.; and Vinyals, O. 2019. Generating Diverse High-Fidelity Images with VQ-VAE-2. *CoRR*, abs/1906.00446.

Rokh, B.; Azarpeyvand, A.; and Khanteymoori, A. 2023. A Comprehensive Survey on Model Quantization for Deep Neural Networks in Image Classification. *ACM Transactions on Intelligent Systems and Technology*, 14(6): 1–50.

Salimans, T.; and Ho, J. 2022. Progressive Distillation for Fast Sampling of Diffusion Models. arXiv:2202.00512.

Sohl-Dickstein, J.; Weiss, E. A.; Maheswaranathan, N.; and Ganguli, S. 2015. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. *CoRR*, abs/1503.03585.

Song, Y.; and Ermon, S. 2019. Generative Modeling by Estimating Gradients of the Data Distribution. *CoRR*, abs/1907.05600.

Sun, Y.; Dong, L.; Zhu, Y.; Huang, S.; Wang, W.; Ma, S.; Zhang, Q.; Wang, J.; and Wei, F. 2024. You Only Cache Once: Decoder-Decoder Architectures for Language Models. arXiv:2405.05254.

Tian, K.; Jiang, Y.; Yuan, Z.; Peng, B.; and Wang, L. 2024. Visual Autoregressive Modeling: Scalable Image Generation via Next-Scale Prediction.

Touvron, H.; and et al. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288.

van den Oord, A.; Kalchbrenner, N.; and Kavukcuoglu, K. 2016. Pixel Recurrent Neural Networks. *CoRR*, abs/1601.06759.

van den Oord, A.; Kalchbrenner, N.; Vinyals, O.; Espeholt, L.; Graves, A.; and Kavukcuoglu, K. 2016. Conditional Image Generation with PixelCNN Decoders. *CoRR*, abs/1606.05328.

van den Oord, A.; Vinyals, O.; and Kavukcuoglu, K. 2017. Neural Discrete Representation Learning. *CoRR*, abs/1711.00937.

Voronov, A.; Kuznedelev, D.; Khoroshikh, M.; Khrulkov, V.; and Baranchuk, D. 2024. Switti: Designing Scale-Wise Transformers for Text-to-Image Synthesis.

Wu, H.; and Tu, K. 2024. Layer-Condensed KV Cache for Efficient Inference of Large Language Models. arXiv:2405.10637.

Xiao, G.; Tian, Y.; Chen, B.; Han, S.; and Lewis, M. 2024. Efficient Streaming Language Models with Attention Sinks. arXiv:2309.17453.

Xie, R.; Zhao, T.; Yuan, Z.; Wan, R.; Gao, W.; Zhu, Z.; Ning, X.; and Wang, Y. 2024. LiteVAR: Compressing Visual Autoregressive Modelling with Efficient Attention and Quantization. arXiv:2411.17178.

Yu, J.; Li, X.; Koh, J. Y.; Zhang, H.; Pang, R.; Qin, J.; Ku, A.; Xu, Y.; Baldridge, J.; and Wu, Y. 2021. Vector-quantized Image Modeling with Improved VQGAN. *CoRR*, abs/2110.04627.

Yu, J.; Xu, Y.; Koh, J. Y.; Luong, T.; Baid, G.; Wang, Z.; Vasudevan, V.; Ku, A.; Yang, Y.; Ayan, B. K.; Hutchinson, B.; Han, W.; Parekh, Z.; Li, X.; Zhang, H.; Baldridge, J.; and Wu, Y. 2022. Scaling Autoregressive Models for Content-Rich Text-to-Image Generation. arXiv:2206.10789.

Zhang, J.; Xiong, F.; and Xu, M. 2024. 3D representation in 512-Byte:Variational tokenizer is the key for autoregressive 3D generation. arXiv:2412.02202.

Zhang, Q.; Dai, X.; Yang, N.; An, X.; Feng, Z.; and Ren, X. 2024. VAR-CLIP: Text-to-Image Generator with Visual Auto-Regressive Modeling.

Zhang, Z.; Sheng, Y.; Zhou, T.; Chen, T.; Zheng, L.; Cai, R.; Song, Z.; Tian, Y.; Ré, C.; Barrett, C.; Wang, Z.; and Chen, B. 2023. H_2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models. arXiv:2306.14048.

Zhu, X.; Li, J.; Liu, Y.; Ma, C.; and Wang, W. 2024. A Survey on Model Compression for Large Language Models. arXiv:2308.07633.

Zirui Liu; Jiayi Yuan; Hongye Jin; Shaochen Zhong; Zhaozhuo Xu; Braverman, V.; Beidi Chen; and Hu, X. 2023. KIVI : Plug-and-play 2bit KV Cache Quantization with Streaming Asymmetric Quantization.