SparVAR: Exploring Sparsity in Visual AutoRegressive Modeling for Training-Free Acceleration

Supplementary Material

A. Algorithmic Details of CS^4A

A.1. Hardware-Efficient Sparse Computation

As illustrated in Fig. 1, the pipeline of CS^4A comprises two distinct phases: **sparse pattern identification** and **hardware-accelerated sparse computation**. At the sparse decision scale S, we compute full dense attention to capture the global dependency structure. To avoid the computational overhead of sorting per-query token attention scores, we adopt the block-wise column-sum strategy [8]. We partition the attention map $\mathbf{P}^{(S)}$ into contiguous query blocks of size C, highlighted by the red dashed boxes in Fig. 1. Within each query block, we aggregate attention scores along the query dimension to derive a cumulative importance score for each key column. Subsequently, the sparse indices $\text{inds}^{(S)}$ are determined by performing a Top-K selection on these column sums. These indices are then projected to the target scale $k \in \{S+1, \dots, K\}$ via our sparse index mapping strategy $\mathcal{M}_{S\to k}(\cdot)$, enabling efficient sparse computation at high-resolution scales.

A primary challenge in sparse attention lies in the inefficiency of unstructured sparsity, where random sparse matrix multiplication fails to effectively leverage the Tensor Cores on modern GPUs. Furthermore, runtime overheads such as dynamic mask computation and cache maintenance can introduce significant latency. To address this, we adopt the Tile Packing technique [8], which maps sparse computation into compacted dense computation. We extend their efficient custom kernels to support our causal cross-scale attention mechanism. Specifically, guided by the mapped sparse indices $\text{inds}^{(k)}$, the kernel **gathers** discrete, non-contiguous key and value vectors from the GPU global memory (HBM). These vectors are then **packed** into contiguous dense tiles within the GPU shared memory (SRAM). Once resident in SRAM, these packed tiles constitute a logically dense matrix (as shown in the bottom-left of Fig. 1). This transformation enables the GPU to execute standard dense General Matrix Multiplications (GEMMs) using Tensor Cores, thereby achieving peak computational throughput.

A.2. Cross-Scale Sparse Index Mapping

To efficiently propagate the sparse activation patterns from the decision scale S to a higher-resolution target scale $k \in \{S+1, \dots, K\}$, we propose a two-stage mapping mechanism, denoted as $\mathcal{M}_{S\to k}(\cdot)$. This mapping addresses two fundamental challenges: the quadratic expansion of the query grid and the scale drift of historical key-value pairs. Specifically, we decompose $\mathcal{M}$ into two coupled transformations: *Query Block Homography* and *Relative Scale Alignment*.

Query Block Homography. Following the hardware-efficient Column-Sum design [8], query tokens are partitioned into contiguous blocks of size C (e.g., $C = 192$). As the spatial resolution expands from $h_S \times w_S$ to $h_k \times w_k$, the number of query blocks increases from $G_S = \lceil N_S/C \rceil$ to $G_k = \lceil N_k/C \rceil$. Let $\mathcal{G}_S = \{0, \dots, G_S - 1\}$ and $\mathcal{G}_k = \{0, \dots, G_{k-1}\}$ denote the sets of query block indices at scales S and k, respectively. We model the mapping from a target block $g_k \in \mathcal{G}_k$ to a source block $g_S \in \mathcal{G}_S$ as a nearest-neighbor interpolation within the normalized coordinate space $[0, 1]$. Formally, the mapping is defined as:

$$\phi(g_k) = \left\lfloor \frac{g_k + 0.5}{G_k} \cdot G_S - 0.5 \right\rceil_{\mathcal{G}_S},$$

where $\lfloor \cdot \rceil_{\mathcal{G}_S}$ denotes rounding to the nearest integer clipped within the bounds of $\mathcal{G}_S$. This transformation ensures that query tokens at the finer scale k inherit the sparsity pattern from their spatially corresponding region at the coarser scale S, thereby preserving the global attention structure.

Relative Scale Alignment for KVs. The KV cache in VAR is a concatenation of tokens from all historical scales, e.g., for the sparse decision scale S, its KV cache $\mathbf{K}_{\le S} = \text{Concat}(\mathbf{K}_1, \dots, \mathbf{K}_S)$. A flattened global KV index j is structurally ambiguous without scale context, i.e., j does not explicitly encode which historical scale it comes from, making direct cross-scale mapping ambiguous. To resolve this ambiguity, we propose a *Decompose–Align–Project* mechanism to map sparse KV indices from scale S to later target scales $k \in \{S+1, \dots, K\}$.

- **Decomposition.** We first decompose each global KV index j at scale S into its source scale and its within-scale offset. Let

$$C_0 = 0, \qquad C_i = \sum_{r=1}^{i} N_r, \quad i = 1, \dots, S,$$

where N_i is the number of tokens at scale i. Then the source scale l of index j is determined by

$$C_{l-1} \le j < C_l,$$

and the corresponding local offset is

$$\delta = j - C_{l-1}.$$

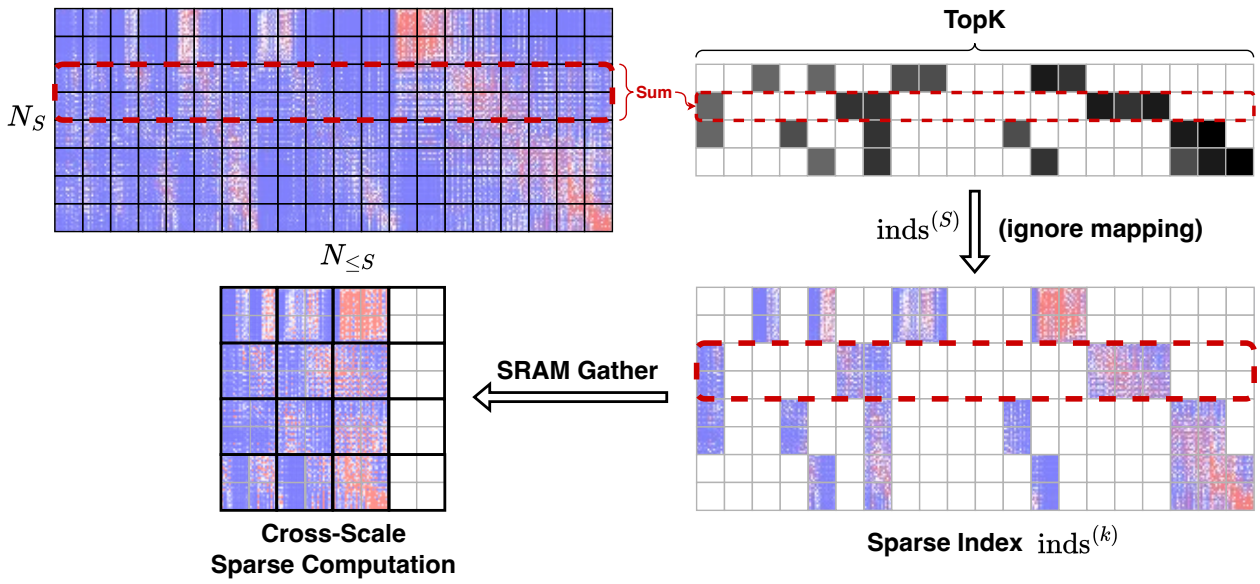

Figure 1. The illustration of CS^4A. The sparse index mapping process is omitted for clarity.

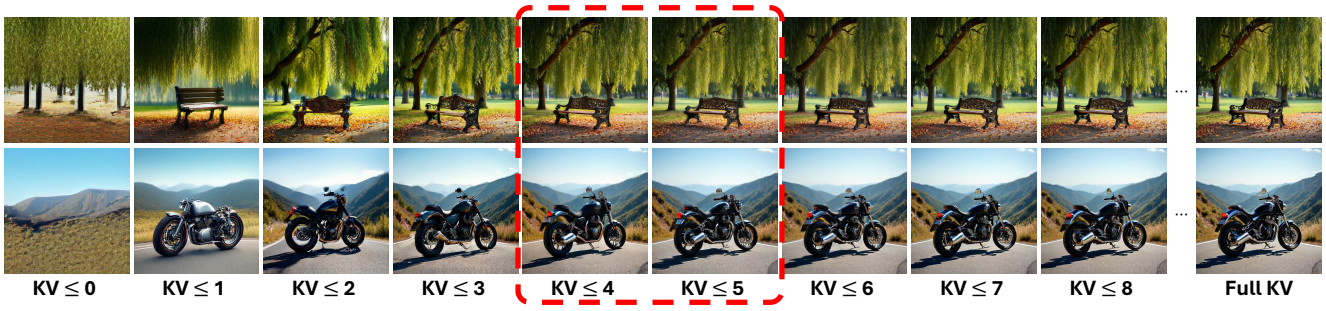

Figure 2. The influence of attention sinks in KV cache on generated images.

Thus, each flattened index is decomposed into a tuple (l, δ), where l identifies the historical source scale and δ specifies the relative position within that scale.

- **Relative Alignment.** Based on the cross-scale self-similarity observation, the sparse attention patterns depend on the *relative scale distance* between the current query scale and the historical key scales. Hence, when mapping from the sparse decision scale S to a target scale k, we preserve this relative offset. Specifically, if a KV index at scale S comes from historical scale l, then its aligned source scale at target scale k is

$$l' = k - (S - l).$$

Equivalently, this enforces

$$S - l = k - l',$$

so that the mapped interaction at scale k corresponds to the same relative cross-scale region as at scale S. For example, the sparse pattern associated with $\mathbf{A}^{(S,l)}$ is mapped to the corresponding region in $\mathbf{A}^{(k,l')}$ with identical relative distance.

- **Spatial Projection.** Finally, we project the local spatial offset δ from the resolution of scale l ($h_l \times w_l$) to the resolution of the target scale l' ($h_{l'} \times w_{l'}$). To preserve 2D spatial locality, we de-linearize δ into 2D coordinates (u, v), perform bilinear scaling, and re-linearize:

$$u' = \lfloor u \cdot \frac{h_{l'}}{h_l} \rfloor, \quad v' = \lfloor v \cdot \frac{w_{l'}}{w_l} \rfloor, \quad \delta' = u' \cdot w_{l'} + v'.$$

Finally, the mapped global KV index at target scale k is reconstructed as

$$j' = C_{l'-1} + \delta',$$

where $C_{l'-1}$ denotes the starting index of scale l' in the KV cache of target scale k.

This procedure preserves both relative cross-scale structure and within-scale locality, enabling sparse patterns identified at the sparse decision scale to be consistently reused at later high-resolution scales.

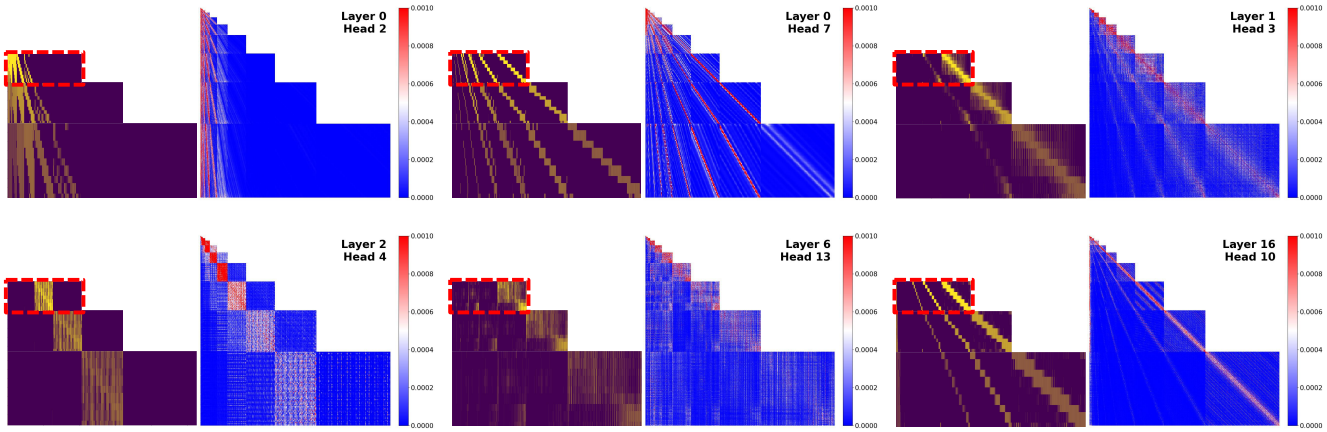

Figure 3. Visualization of cross-scale sparse index mapping. The red dashed box highlights the sparse indices identified at the sparse decision scale S, while the blocks below display the corresponding sparse indices projected onto subsequent high-resolution scales. Evidently, the mapped sparse indices precisely cover the most salient attention activation regions of the baseline model.

Attention Sink. To mitigate the loss of critical global context during sparse selection, we explicitly enforce a **sink token preservation** strategy. Let $\mathcal{A}_{\text{sink}} = \{0, \ldots, N_{\text{sink}}-1\}$ denote the set of indices corresponding to tokens from the initial scales. The final sparse index set for the target scale K is constructed as the union of the mapped dynamic indices and these static sink indices:

$$\text{inds}^{(k)} = \mathcal{A}_{\text{sink}} \cup \mathcal{M}_{S\to k}(\text{inds}^{(S)}).$$

This ensures that the accelerated sparse attention mechanism retains a stable attention sink, thereby preventing the collapse of semantic coherence during the generation process.

Effectiveness Analysis. To validate the effectiveness of the proposed sparse index mapping strategy, we visualize the alignment between the predicted sparse patterns and the full attention activations of the baseline model in Fig. 3. Specifically, we visualize the sparse indices derived from the sparse decision scale S (highlighted by red dashed boxes) alongside the mapped indices at subsequent high-resolution scales, allowing for a direct examination of the sparse patterns predicted by CS^4A. As illustrated across various layers and heads, the mapped sparse indices (depicted in gold) exhibit a remarkable spatial correspondence with the ground-truth activation hotspots (red regions) of the baseline model. Notably, this alignment persists even in the final scales where the resolution expands significantly, confirming that the structural self-similarity of cross-scale attention is robustly preserved by our mapping strategy. This coverage of salient attention activation regions ensures that the sparse computation approximates the full attention mechanism with minimal error. It fundamentally explains why CS^4A achieves substantial acceleration while maintaining the generation of high-frequency details, as the model continues to attend to the most critical contextual information during the fine-grained refinement stages.

A.3. Difference from Chipmunk.

Chipmunk achieves training-free acceleration by exploiting the temporal redundancy and the similarity of attention activations across denoising steps inherent in Diffusion Transformers (DiTs) [1, 5, 10]. However, the premise that latent variables maintain a constant spatial resolution during denoising does not hold for the unique next-scale prediction paradigm of VAR. Fundamentally, VAR is characterized by a progressive increase in token count as spatial resolution grows during inference, standing in stark contrast to DiT, where the sequence length remains invariant. The motivation behind our proposed CS^4A lies in the cross-scale self-similarity intrinsic to VAR models, namely that attention activation patterns at smaller scales exhibit similarities to those at larger scales in corresponding regions. To the best of our knowledge, SparVAR is the first work to identify and exploit this attention activation pattern redundancy for accelerating visual autoregressive generation, exploring a dimension of sparsity orthogonal to the temporal sparsity found in DiTs. Furthermore, to effectively propagate this sparsity across scales, we propose an efficient sparse index mapping strategy that projects sparse patterns from low-resolution to high-resolution scales via rigorous geometric transformations. From an engineering perspective, we extend Chipmunk's highly optimized sparse kernels—originally designed solely for standard self-attention—to support the causal cross-scale attention required by VAR. This generalization unlocks the potential of sparse computing for a broader class of autoregressive models beyond the fixed-length setting of DiTs.

Table 1. Quantitative comparison on the DPG-Bench dataset. The best results in each setting are highlighted in **bold**.

Methods	Acceleration		DPG-Bench					
	#Scales ↓	Speedup ↑	Global ↑	Entity ↑	Attribute ↑	Relation ↑	Other ↑	Overall ↑
w/o skip scales								
Infinity-2B [3]	13	1.00×	88.499	88.120	**88.957**	88.767	84.521	82.839
FastVAR [2]	13	1.01×	77.291	89.168	88.537	**91.932**	84.637	82.723
ScaleKV [7]	13	0.70×	78.088	88.550	87.495	90.229	**89.468**	**82.885**
SparVAR (Ours)	13	**1.38**×	**89.021**	**89.174**	87.274	91.015	84.405	82.858
w/ skip last 2 scales								
FastVAR [2]	11	1.33×	81.423	88.267	88.184	90.127	**89.351**	82.651
SparVAR (Ours)	11	**1.70**×	**83.864**	**88.945**	**88.274**	**90.883**	81.888	**82.663**
w/ dynamic skip last scales								
SkipVAR-2B [6]	10∼13	1.33×	85.273	**89.115**	**89.289**	**89.519**	84.836	82.877
SparVAR (Ours)	10∼13	**1.58**×	**89.216**	88.896	88.004	89.279	**89.366**	**82.891**
w/o skip scales								
Infinity-8B [3]	13	1.00×	**92.144**	89.741	90.376	92.084	91.862	**86.440**
FastVAR [2]	13	1.14×	81.784	90.195	89.311	94.084	84.496	86.343
ScaleKV [7]	13	0.67×	81.876	90.041	89.666	**94.464**	**92.675**	86.333
SparVAR (Ours)	13	**1.57**×	87.306	**91.510**	**90.862**	91.565	90.193	86.411
w/ skip last 2 scales								
FastVAR [2]	11	1.79×	87.295	**91.466**	90.048	**91.113**	**90.308**	86.332
SparVAR (Ours)	11	**2.28**×	**91.729**	91.068	**91.031**	90.354	89.072	**86.468**
w/ dynamic skip last scales								
SkipVAR-8B [6]	10∼13	1.71×	**92.648**	90.591	**91.032**	90.088	86.008	86.293
SparVAR (Ours)	10∼13	**2.05**×	88.227	**91.863**	90.703	**92.239**	**86.758**	**86.409**

Table 2. Quantitative comparison of memory consumption. "Torch Mem." refers to peak PyTorch allocated memory, and "NV Mem." refers to NVIDIA physical memory. The best results in each setting are highlighted in **bold**.

Methods	2B model				8B model			
	GenEval	Speedup ↑	**Torch Mem. ↓**	**NV Mem. ↓**	GenEval ↑	Speedup ↑	**Torch Mem. ↓**	**NV Mem. ↓**
w/o skip scales								
Infinity [3]	0.736	1.00×	15.78GB	28.63GB	**0.798**	1.00×	37.11GB	55.75GB
FastVAR [2]	0.712	1.01×	15.91GB	28.45GB	0.792	1.14×	37.41GB	54.29GB
ScaleKV [7]	0.732	0.70×	**10.98GB**	**17.01GB**	0.793	0.67×	**22.71GB**	**29.64GB**
SparVAR (Ours)	**0.738**	**1.38**×	16.02GB	23.21GB	0.796	**1.57**×	37.48GB	45.45GB
+ KV comp.	0.730	1.29×	12.74GB	18.51GB	0.793	1.31×	30.42GB	35.18GB
w/ skip last 2 scales								
FastVAR [2]	0.716	1.33×	11.65GB	19.73GB	0.79	1.79×	**26.19GB**	37.31GB
SparVAR (Ours)	**0.725**	**1.70**×	**11.12GB**	**17.04GB**	**0.800**	**2.28**×	26.42GB	**29.73GB**
w/ dynamic skip last scales								
SkipVAR [6]	0.730	1.33×	**12.50GB**	23.32GB	0.789	1.71×	30.05GB	44.00GB
SparVAR (Ours)	**0.732**	**1.58**×	**12.50GB**	**16.98GB**	**0.796**	**2.05**×	**30.00GB**	**34.10GB**

B. Complexity of Cross-Scale Sparse Attention

We analyze the time complexity of the proposed CS^4A and $CSLA$, comparing them against standard cross-scale full attention. Let N_k denote the number of tokens at the current scale k, and $N_{\leq k}$ represent the cumulative number of tokens across all preceding scales. The feature dimension is denoted by d. For a given scale k, following Eq. **??**, the computational complexity of full dense attention is given by:

$$\mathcal{O}(dN_kN_{\leq k}).$$

CS⁴A. CS^4A initially performs full dense attention at the sparse decision scale S to identify salient sparse pat-

terns. The complexity of full attention at scale S is $\mathcal{O}(dN_S N_{\leq S})$, consistent with the baseline model. Since S typically corresponds to a mid-resolution scale, this computational overhead is relatively marginal. Additionally, computing the column-sum tensor $D^{(S)}$ per query block introduces a complexity of $\mathcal{O}(dG_S N_{\leq S})$, where $G_S = \lceil N_S/C \rceil$. Consequently, the total complexity at the sparse decision scale S is given by:

$$\mathcal{O}^{(S)} = \mathcal{O}(dN_S N_{\leq S}) + \mathcal{O}(dG_S N_{\leq S}).$$

In practice, hardware-friendly block size C (e.g., 128 or 192) makes the second term negligible compared to the first.

For subsequent high-resolution scales $k > S$, we employ cross-scale sparse computation, which restricts attention calculation to the Top-K selected KV pairs. The process begins with sparse index mapping. Mapping the query blocks and KV indices incurs a time complexity of $\mathcal{O}(N_k)$, as it involves only lightweight coordinate transformations and gather operations. Let α denote the sparsity ratio (e.g., retaining the top 20% of KVs). For any given query, the number of active KVs is $\alpha \cdot N_{\leq k} \ll N_{\leq k}$. Therefore, the complexity of the sparse computation is reduced to

$$\mathcal{O}(d\alpha N_k N_{\leq k}).$$

This implies a linear relationship between total complexity and the sparsity ratio. For small α, CS^4A achieves significant acceleration. Theoretically, as N_k increases, the complexity reduction factor asymptotically approaches $1/\alpha$.

CSLA. $CSLA$ decomposes the historical KV cache into a dense attention sink region and a block-wise local sparse region. The first N_{sink} KV tokens from early scales are always attended with full dense attention, preserving their role as global structural anchors [12]. The remaining $N_{\leq k} - N_{\text{sink}}$ KV tokens are arranged as 2D grids per historical scale, where each scale $h \in \{1, \ldots, k\}$ is assigned a scale-dependent local window size w_h (with radius $r_h = \lfloor w_h/2 \rfloor$). Consequently, for any query at scale k, the number of KV tokens involved in the sparse region is upper-bounded by

$$N_{\text{local}} \leq \sum_{h \leq k} \mathrm{w}_h^2.$$

In our design, the window sizes, the window sizes w_h are typically kept small—for example, the last three scales use $\{3, 5, 7\}$—so N_{local} is effectively a constant upper bound. The computational complexity of $CSLA$ at scale k therefore becomes

$$\mathcal{O}\left(dN_k(N_{\text{sink}} + N_{\text{local}})\right).$$

Moreover, since N_{sink} covers only a few early scales (e.g., the **first 5 scales** contain just **121 KV tokens**), we have $N_{\text{sink}} + N_{\text{local}} \ll N_{\leq k}$ at late high-resolution scales. Therefore, compared with fully dense attention, $CSLA$ significantly reduces the computational cost.

C. Implementation Details.

The baseline VAR model, Infinity [3], operates across a total of 13 resolution scales, corresponding to 13 iterations of next-scale autoregressive prediction. We treat the **first 5 scales** as the **attention sink** and always preserve full attention computation over this region. The sparsity configurations are set as follows: (i) In the setting **without scale skipping**, we designate Scale 11 (with resolution 40×40) as the sparse decision scale, and apply hardware-efficient sparse computation to the last 2 scales for acceleration. (ii) When **skipping the last two scales** [2], Scale 10 serves as the sparse decision scale, with sparse computation applied exclusively to Scale 11. (iii) For the **sample-adaptive skipping strategy** [6], we similarly adopt Scale 10 as the sparse decision scale, sparsifying any subsequent scales that are not skipped. Regarding the configuration of sparse attention modules, the application ratio of CS^4A to $CSLA$ across all attention layers is approximately 6:4. Unlike Infinity, HART [9] operates over 14 scales. Following prior work [2], we do not apply scale skipping in any HART experiments, and set its sparse decision scale to Scale 11. Finally, for the acceleration performance analysis, to ensure a fair comparison with the baseline model, we measure throughput using a batch size of 1.

D. Additional Experiments

Comparison on DPG-Bench. To further validate the effectiveness of SparVAR, we provide additional quantitative results on the DPG-Bench [4], which evaluates prompt-following capabilities across multiple dimensions. The results are summarized in Tab. 1. For the 2B model, SparVAR achieves a competitive overall score of **82.858** without skipping scales, surpassing the baseline and FastVAR while delivering a superior $\mathbf{1.38}\times$ speedup. When combined with scale-skipping strategies, our method maintains robust performance, achieving the highest score of **82.891** under the dynamic skipping setting with a $\mathbf{1.58}\times$ speedup, demonstrating an effective balance between acceleration and semantic fidelity. For the larger 8B model, SparVAR achieves an overall score of **86.411** without scale skipping, comparable to the baseline (86.440) and outperforming FastVAR and ScaleKV, all while providing a substantial $\mathbf{1.57}\times$ acceleration. Notably, when integrated with scale-skipping strategies, SparVAR consistently outperforms other baselines. In the "skip last 2 scales" setting, it attains the highest score of **86.468**—exceeding even the baseline—alongside a remarkable speedup. The above results demonstrate the generality and effectiveness of our method.

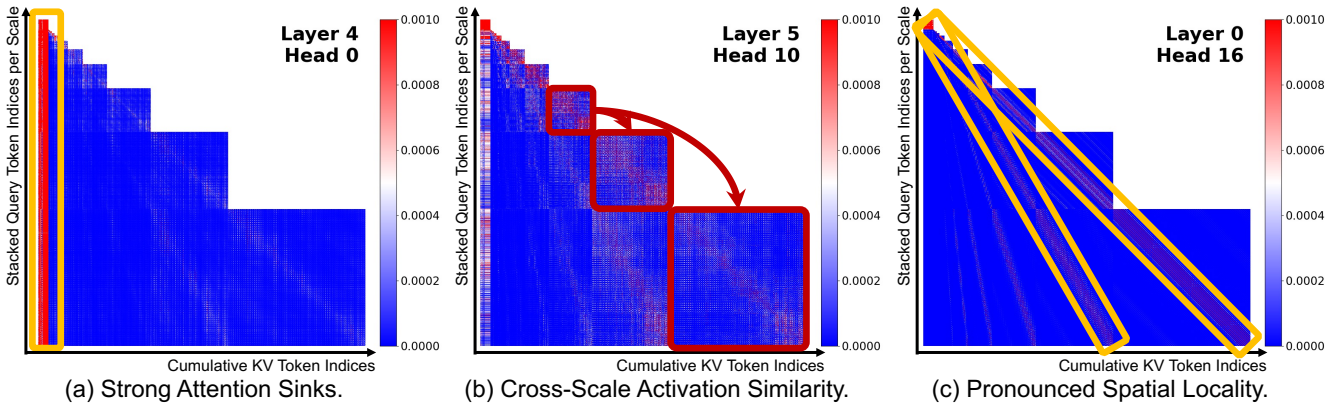

Figure 4. Visualization of attention activation patterns in the HART [9] across different layers and heads.

Table 3. Quantitative comparison on efficiency and quality on 1024×1024 image generation. Note that the efficiency of HART baseline is tested under FlashAttention.

Methods	Acceleration				GPU Memory		GenEval	DPG-Bench	HPSv2.1	ImageReward	Quality Metrics		
	#Scales ↓	Speedup ↑	Latency ↓	Throughput ↑	Torch Mem. ↓	NV Mem. ↓	Overall ↑	Overall ↑	Overall ↑	Overall ↑	PSNR ↑	SSIM ↑	LPIPS ↓
HART [9]	14	1.00×	446.30ms	2.24it/s	19.35	27.93	**0.509**	74.754	29.070	0.661	-	-	-
FastVAR [2]	14	1.11×	403.33ms	2.48it/s	19.33	**25.72**	0.504	74.762	27.680	0.602	20.850	0.704	0.316
SparVAR (Ours)	14	**1.16×**	**383.11ms**	**2.61it/s**	**19.29**	26.98	0.507	**75.625**	**29.140**	**0.680**	**21.157**	**0.735**	**0.240**

Memory Consumption. We also provide a comprehensive memory comparison for generating 1024×1024 images in the Tab. 2, including peak PyTorch active memory (Torch Mem.) and NVIDIA physical memory (NV Mem.). The memory reduction in FastVAR and SkipVAR rely heavily on scale-skipping strategies. Notably, FastVAR (w/o skip scales) exhibits memory usage nearly to the baseline, as the overhead of caching intermediate scale tokens for restoration offsets the benefits of token pruning. While ScaleKV achieves the lowest memory, its design cannot be accelerated by modern efficiency attention kernels (e.g., FlashAttention), resulting in inference speeds significantly slower than the baseline. Our sparse attention is optimized primarily for **inference latency** rather than memory minimization. Some reduction in physical memory originates from the efficient memory access strategies of the kernels. As a 'plug-and-play' module, SparVAR is compatible with KV compression methods (e.g., retaining only sinks and local scales KV Cache, denoted as '+ KV Comp.' in Tab. 2). We leave further dedicated memory optimizations to future work.

Generalizability of SparVAR. We further validated SparVAR on another representative VAR-based text-to-image model HART [9]. Our aim is to demonstrate that SparVAR is not custom-designed for a specific VAR architecture (e.g., Infinity [3]), but rather serves as a plug-and-play module for accelerating visual generation models based on the next-scale prediction paradigm. Fig. 4 illustrates the sparse activation patterns at each scale of the HART model, which exhibit three key properties similar to those observed in Infinity. It is worth noting that since the first scale in HART is not a 1×1 token map but comprises text tokens with a fixed sequence length of 300, its attention sink phenomenon is even more stronger compared to Infinity. Consequently, we adjust the configuration of the attention sink scales for our cross-scale sparse attention. Specifically, in addition to the first 5 scales, the sink region must consistently include the initial text tokens. Note that, following the setting of FastVAR, this model without scale skipping. Shown in Tab. 3, our SparVAR achieves speedup and consistently outperforms FastVAR across 4 high-level benchmarks, even exceeding the baseline. Superior low-level metrics further confirm robust detail preservation.

Qualitative Comparison. We compare SparVAR against the baseline Infinity-2B and other training-free acceleration methods across complex generation scenarios from the HPSv2.1 [11] benchmark. As shown in Fig. 5, in the setting without scale skipping, SparVAR achieves a $1.38\times$ speedup while generating images with visual fidelity nearly identical to the baseline model. This is particularly evident in the second row, where our method accurately reconstructs the fine-grained square window panes of the white building—a structural detail that both FastVAR and ScaleKV fail to preserve, resulting in blurred or distorted features. Crucially, the robustness of SparVAR extends to the aggressive skip-scale setting. When skipping the last two high-resolution scales to achieve a $\mathbf{1.70}\times$ speedup, SparVAR

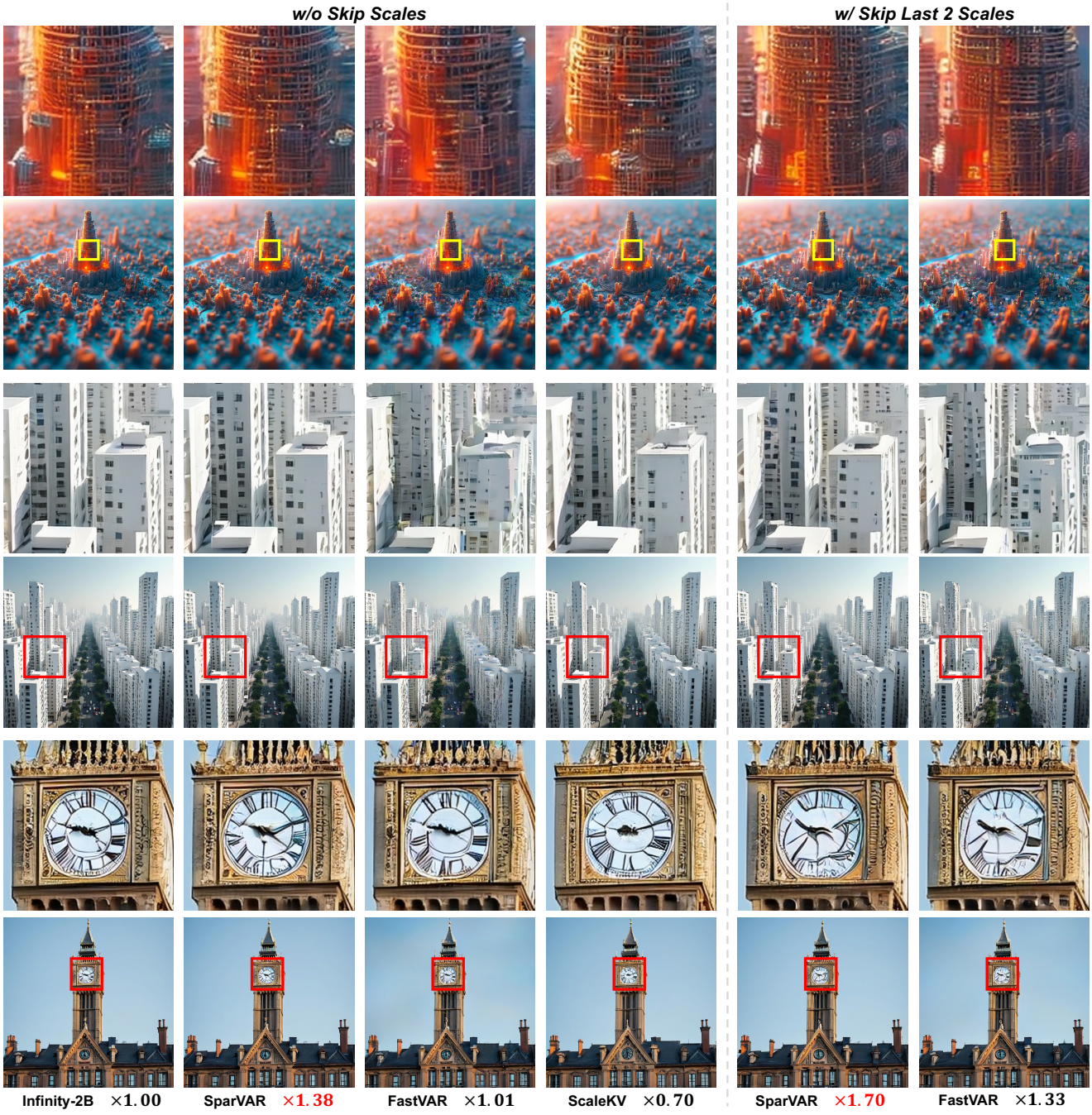

Figure 5. Qualitative comparison of complex scene generation on HPSv2.1 [11] benchmark. Zoom in for fine-detail visualization.

still successfully generates coherent structural details (e.g., the square windows), benefiting from the reinforced spatial locality in our sparse attention mechanism. In contrast, FastVAR exhibits degradation, introducing severe artifacts and failing to maintain the geometric integrity of the scene. These results qualitatively confirm that SparVAR's cross-scale sparse attention is highly effective at compressing redundant computation while selectively preserving the critical high-frequency information necessary for photorealistic generation. **Human evaluation of visualization** results is extremely important. We conducted a small-scale blind user study (constrained by budget) with 7 participants. The results indicate that $> 70\%$ of evaluators preferred SparVAR over other acceleration methods for its superior preservation of fine-grained details.

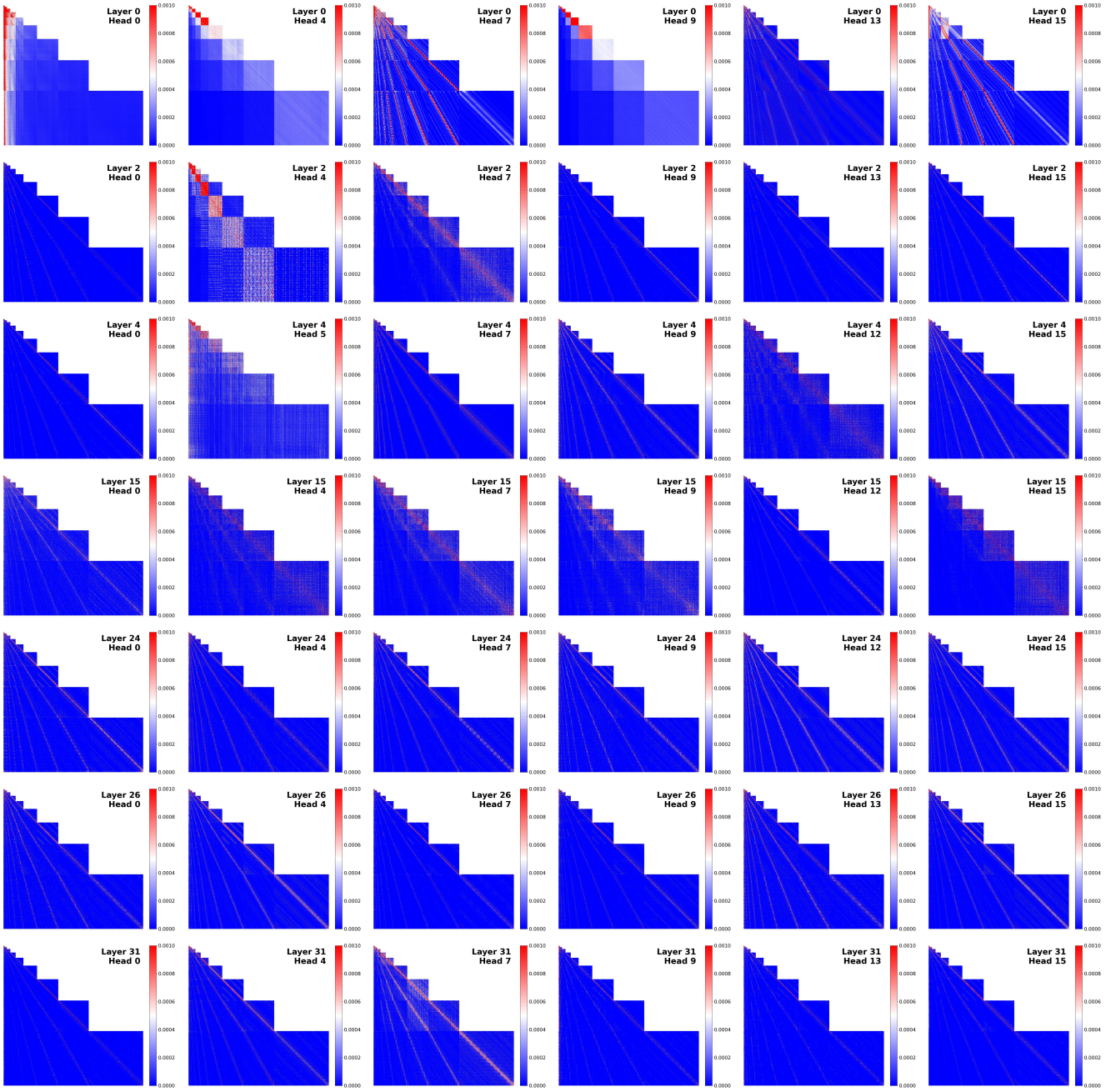

Figure 6. Visualization of the evolving attention activation patterns in VAR across layer depths.

Optimal Configuration of Sparse Attention Modules. As illustrated in Fig. 6, we further observe that in pretrained VAR models, the attention activation pattern evolves with layer depth, shifting from a relatively diverse distribution to a pronounced local pattern concentrated along the cross-scale diagonal. Consequently, the dispersed activation patterns in shallower layers are better suited for CS^4A, which dynamically predicts sparse structures, whereas the fixed, strong locality in deeper layers is more effectively handled by the specialized $CSLA$. To determine the optimal configuration, we conduct a systematic ablation study on the layer-wise allocation of CS^4A and $CSLA$. Starting from a baseline where all attention layers utilize $CSLA$, we progressively substitute them with CS^4A from the shallowest to the deepest layers.

The quantitative results, illustrated in Fig. 7, reveal two key insights: **1) Synergistic effect on generation quality.** As the proportion of CS^4A gradually increases, we ob-

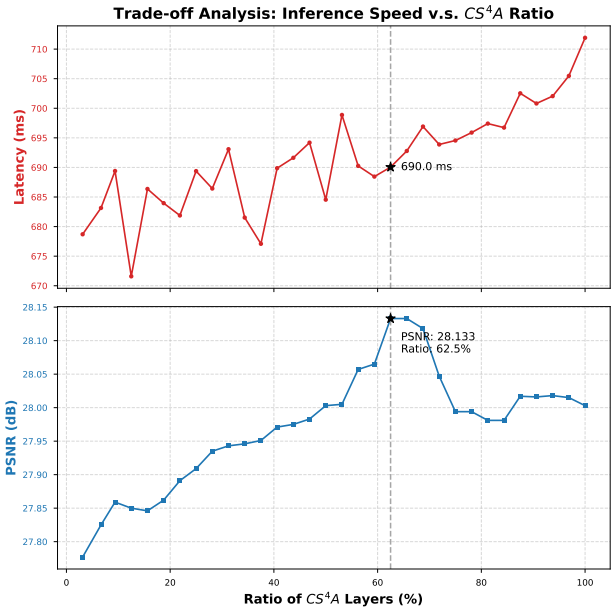

Figure 7. Configuration trade-off between CS^4A and $CSLA$.

serve a corresponding steady improvement in PSNR. This trend aligns with our analysis, indicating that for the relatively dispersed sparse activation patterns in shallow layers, the cross-scale mapping of CS^4A achieves superior pattern prediction compared to the fixed local window sparse of $CSLA$. However, extending CS^4A to the deepest layers (beyond 70%) leads to a slight decline in PSNR. This suggests that deep layers are indeed specialized for high-frequency texture refinement, where the strong inductive bias of spatial locality enforced by $CSLA$ is more effective than the mapped sparse patterns. **2) Latency-Quality trade-off.** In terms of efficiency, replacing $CSLA$ with CS^4A introduces a marginal increase in latency, primarily due to the overhead of index mapping and memory gathering compared to the highly optimized block-wise kernel of $CSLA$. Nevertheless, the gain in visual fidelity (+0.35 dB PSNR) justifies this minimal cost. Consequently, we adopt a hybrid configuration with about 60% CS^4A and 40% $CSLA$ as the default setting for 2B and 8B models, ensuring optimal performance.

References

[1] Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, Sumith Kulal, et al. Flux. 1 kontext: Flow matching for in-context image generation and editing in latent space. *arXiv e-prints*, pages arXiv–2506, 2025. 3

[2] Hang Guo, Yawei Li, Taolin Zhang, Jiangshan Wang, Tao Dai, Shu-Tao Xia, and Luca Benini. Fastvar: Linear visual autoregressive modeling via cached token pruning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19011–19021, 2025. 4, 5, 6

[3] Jian Han, Jinlai Liu, Yi Jiang, Bin Yan, Yuqi Zhang, Zehuan Yuan, Bingyue Peng, and Xiaobing Liu. Infinity: Scaling bitwise autoregressive modeling for high-resolution image synthesis. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 15733–15744, 2025. 4, 5, 6

[4] Xiwei Hu, Rui Wang, Yixiao Fang, Bin Fu, Pei Cheng, and Gang Yu. Ella: Equip diffusion models with llm for enhanced semantic alignment. *arXiv preprint arXiv:2403.05135*, 2024. 5

[5] Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, et al. Hunyuanvideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024. 3

[6] Jiajun Li, Yue Ma, Xinyu Zhang, Qingyan Wei, Songhua Liu, and Linfeng Zhang. Skipvar: Accelerating visual autoregressive modeling via adaptive frequency-aware skipping. *arXiv preprint arXiv:2506.08908*, 2025. 4, 5

[7] Kunjun Li, Zigeng Chen, Cheng-Yen Yang, and Jenq-Neng Hwang. Memory-efficient visual autoregressive modeling with scale-aware KV cache compression. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. 4

[8] Austin Silveria, Soham V Govande, and Daniel Y Fu. Chipmunk: Training-free acceleration of diffusion transformers with dynamic column-sparse deltas. In *ES-FoMo III: 3rd Workshop on Efficient Systems for Foundation Models*, 2025. 1

[9] Haotian Tang, Yecheng Wu, Shang Yang, Enze Xie, Junsong Chen, Junyu Chen, Zhuoyang Zhang, Han Cai, Yao Lu, and Song Han. HART: Efficient visual generation with hybrid autoregressive transformer. In *The Thirteenth International Conference on Learning Representations*, 2025. 5, 6

[10] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025. 3

[11] Xiaoshi Wu, Yiming Hao, Keqiang Sun, Yixiong Chen, Feng Zhu, Rui Zhao, and Hongsheng Li. Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis. *arXiv preprint arXiv:2306.09341*, 2023. 6, 7

[12] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Representation Learning*, pages 21875–21895, 2024. 5