A^{2}R: TRAINING-FREE SPARSE ATTENTION FOR AUTOREGRESSIVE VIDEO DIFFUSION

Anonymous authors
Paper under double-blind review



Figure 1: Compared with dense attention, A^{2}R preserves similar visual quality and temporal coherence across video frames while achieving nearly 80% attention sparsity on Self-Forcing Model.

ABSTRACT

Autoregressive video diffusion has recently emerged as a promising paradigm for real-time and streaming video generation, owing to its causal decoding formulation and natural support for key–value (KV) caching. However, as generation proceeds, the KV cache grows monotonically over rollout, making attention over long KV histories the dominant inference bottleneck. In this work, we propose **A^{2}R** (**A**nchored and **A**daptive **R**eusable Attention), a training-free sparse attention framework for long-horizon autoregressive video diffusion. Our design is motivated by three observations: early chunks act as persistent visual anchors for later generation, attention sparsity varies substantially across heads and routing patterns remain stable across intermediate denoising steps. Based on these observations, A^{2}R combines an anchored dense-to-sparse transition, per-head adaptive block routing and reusable sparse patterns that are refreshed only at the final denoising step. We further implement an exact Triton-based sparse attention kernel that computes attention only over retained KV blocks. Extensive experiments on autoregressive video diffusion backbones show that A^{2}R achieves around 77% average attention sparsity, reduces attention runtime by **2.1**×, and delivers a **1.33**× end-to-end speedup while preserving strong visual quality and temporal consistency, yielding a favorable quality–efficiency trade-off for video generation.

1 INTRODUCTION

Video generation has become a central problem in multimodal generative modeling, with diffusion-based architectures achieving remarkable progress in visual fidelity, semantic alignment and temporal coherence Ho et al. (2022b;a); Blattmann et al. (2023); Team et al. (2025); Polyak et al. (2024). In particular, diffusion transformers (DiTs) Bao et al. (2022); Peebles & Xie (2023) have become the backbone of many state-of-the-art systems due to their scalability and strong spatiotemporal modeling capacity. However, most existing video diffusion models rely on full bidirectional attention over the entire spatiotemporal sequence, making long-horizon generation increasingly expensive in both computation and memory.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

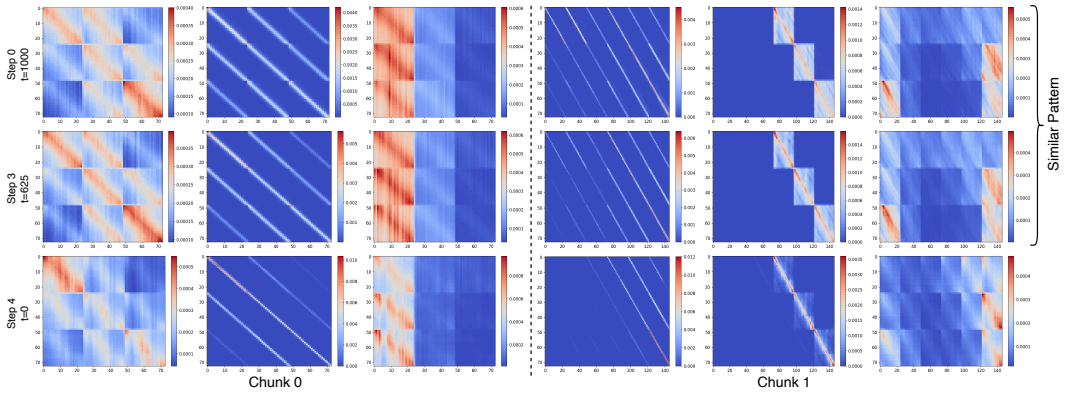

Figure 2: Visualization of head-wise attention patterns across denoising steps. Each column corresponds to the same layer and attention head, visualized at different denoising steps. Different heads exhibit different sparsity levels, motivating per-head adaptive sparsity rather than a fixed global sparsity ratio. In addition, the attention patterns from Step 0 to Step 3 are highly similar, indicating stable intermediate routing and supporting sparse-pattern reuse across denoising steps. The final step is treated separately in A^{2}R and uses refreshed routing.

Autoregressive video diffusion has recently emerged as a promising alternative for real-time and streaming generation Yin et al. (2025); Huang et al. (2025); Chen et al. (2024). By generating frames or chunks sequentially with key–value (KV) caching, autoregressive models naturally support causal decoding and lower viewing latency. However, as generation proceeds, the KV cache grows monotonically over rollout, making attention over long histories the dominant inference bottleneck.

To understand how sparsity should be introduced in this setting, we examine head-wise attention patterns in autoregressive video diffusion. As shown in Figure 2 (additional visualizations are provided in Appendix E), two observations are particularly important. First, different attention heads exhibit markedly different sparsity levels, suggesting that a uniform sparsity budget is inherently suboptimal and that sparse attention should adapt across heads. Second, attention patterns remain highly similar across intermediate denoising steps, indicating that sparse routing evolves slowly during most of the denoising trajectory and therefore need not be reconstructed at every step.

These observations expose several limitations of existing sparse-attention methods for autoregressive video diffusion. Fixed top-K selection or globally shared sparsity ratios fail to capture head-wise heterogeneity Zhang et al. (2025a); Lv et al. (2026); Zhang et al. (2025b), while static sparse masks such as predefined geometric or handcrafted patterns Zhang et al. (2025d); Li et al. (2025) cannot capture the highly non-uniform sparsity structure across heads and query blocks. Moreover, rebuilding sparse patterns independently at every denoising step introduces unnecessary routing overhead, whereas naively reusing the same pattern throughout the entire trajectory may degrade the final denoising step, which is more sensitive to fine-grained refinement Zhang et al. (2025a).

Sparse attention in autoregressive rollout must also preserve foundational context, since early chunks play a disproportionate role in establishing scene layout, object identity and visual style for later generation. Unlike long-context LLM settings motivated by attention sinks or prefix bias Xiao et al. (2023); Zhang et al. (2023); Jiang et al. (2024), autoregressive video diffusion exhibits persistent cross-chunk visual dependence: later frames continue to rely on earlier generated frames for scene structure and subject continuity. As a result, the initial chunk acts as a persistent visual anchor throughout rollout.

Motivated by these observations, we propose **A^{2}R** (**A**nchored and **A**daptive **R**eusable Attention), a training-free sparse attention framework for long-horizon autoregressive video diffusion. A^{2}R combines three components: (i) an *anchored dense-to-sparse transition* that preserves the initial chunk under dense attention, (ii) *per-head adaptive block routing* that allocates non-uniform sparsity across heads and query blocks and (iii) *reusable sparse patterns* that cache and reuse routing results across intermediate denoising steps while refreshing them at the final step. We further implement exact sparse attention over retained KV blocks with a custom Triton kernel. Together, these components substantially reduce the cost of KV-cache attention while preserving strong visual quality and temporal consistency in long-horizon autoregressive generation.

Our contributions are summarized as follows:

- We propose $\mathbf{A}^2\mathbf{R}$, a training-free sparse attention framework for autoregressive video diffusion that combines anchored dense attention, adaptive block routing and reusable sparse patterns.
- We identify two key properties of attention in autoregressive video diffusion: head-wise sparsity heterogeneity and stable intermediate routing, showing how they motivate per-head adaptive sparsity and sparse-pattern reuse across denoising steps.
- We develop an exact Triton-based sparse attention kernel that computes attention only over retained KV blocks, substantially reducing inference cost for long-horizon video generation.

2 Related Work

Diffusion and autoregressive generation. Diffusion models have become the dominant paradigm for video generation, evolving from pixel-space approaches Ho et al. (2022b;a); Singer et al. (2022) to latent-space formulations Rombach et al. (2022); Blattmann et al. (2023) and, more recently, transformer-based architectures Peebles & Xie (2023); Bao et al. (2022). This line of work has led to powerful large-scale systems such as Wan Team et al. (2025), HunyuanVideo Kong et al. (2024), MovieGen Polyak et al. (2024) and Open-Sora Zheng et al. (2024). To improve streaming capability and reduce viewing latency, recent works have explored autoregressive video diffusion Yin et al. (2025); Huang et al. (2025); Chen et al. (2024); Gu et al. (2025), where videos are generated sequentially with causal dependencies and KV caching. While this formulation is attractive for real-time deployment, it remains challenged by rollout instability and the growing cost of KV-cache attention over long histories.

Efficient inference for video generation. A broad line of work seeks to reduce the cost of diffusion-based video generation through optimized attention kernels Dao et al. (2022); Dao (2024), distillation and accelerated sampling Yin et al. (2024a;b); Kim et al. (2025), quantization Zhao et al. (2025), latent compression and flexible tokenization HaCohen et al. (2024); Bachmann et al. (2025), token merging and patch scaling Wu et al. (2025a); Lee et al. (2024) and linear-attention-based architectures Katharopoulos et al. (2020); Xie et al. (2024); Chen et al. (2025). These approaches reduce overall inference cost, but they typically do not address the rollout-specific question of how to preserve foundational context while selectively sparsifying long-history KV-cache attention in autoregressive video diffusion.

Sparse attention for long-context generation. Sparse attention has been widely studied for long-context modeling in both language and video domains Zhang et al. (2023); Liu et al. (2023); Jiang et al. (2024); Xiao et al. (2024; 2025); Xi et al. (2025); Zhang et al. (2025c); Yang et al. (2025); Wu et al. (2025b). Many prior methods employ fixed top-K selection, globally shared sparsity ratios, or step-wise sparse-pattern construction, which can be suboptimal for autoregressive video diffusion. In this setting, sparse attention must preserve a persistent visual anchor across chunks, adapt sparsity across heads and exploit stable intermediate routing across denoising steps. Recent methods such as Light Forcing Lv et al. (2026) and Sparse Forcing Xu et al. (2026) show that rollout-aware sparse designs can improve long-horizon generation. Our work is most closely related to this direction, but differs in focusing on a training-free sparse attention framework that unifies anchored dense-to-sparse transition, per-head adaptive sparsity and reusable sparse patterns for efficient KV-cache attention.

3 Method

3.1 A^2R: Anchored and Adaptive Reusable Attention

Autoregressive video diffusion generates videos chunk by chunk while maintaining a growing key–value (KV) cache. Although KV caching removes redundant computation across denoising steps, attention over the accumulated history still becomes increasingly expensive as generation proceeds. Let $\mathbf{Q} \in \mathbb{R}^{H\times L_Q\times d}$ and $\mathbf{K}, \mathbf{V} \in \mathbb{R}^{H\times L_K\times d}$ denote the query, key and value tensors at a given denoising step, where H is the number of attention heads, L_Q is the query length of the current chunk, L_K is the cached key length and d is the per-head hidden dimension. Standard dense attention is defined as

$$\mathrm{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V}, \tag{1}$$

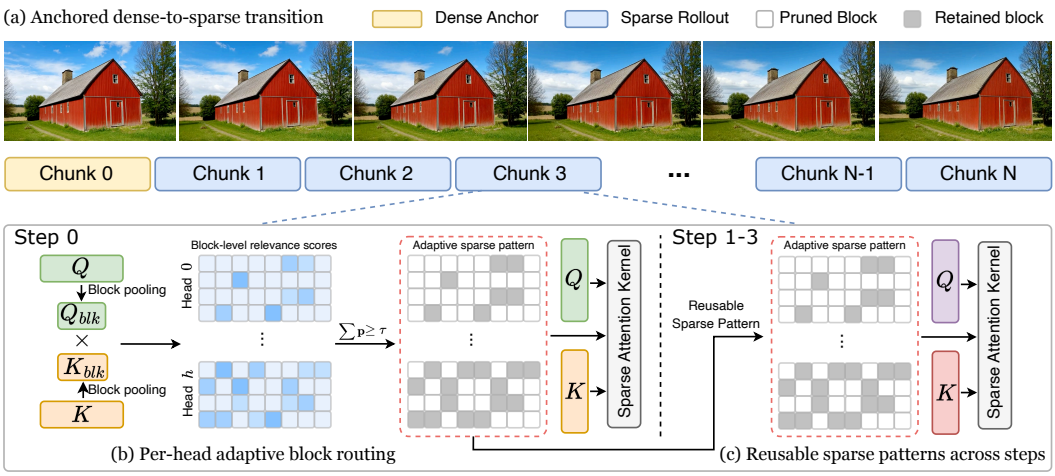

Figure 3: Overview of A^{2}R. (a) Autoregressive video diffusion proceeds chunk by chunk: the first chunk is preserved as a dense anchor, while later chunks use sparse attention. (b) For each later chunk, A^{2}R performs block pooling and computes block-level relevance scores to construct a per-head adaptive sparse pattern, which is then executed by an exact sparse attention kernel over retained KV blocks only. (c) The sparse routing pattern is reused across non-final denoising steps (Steps 0–3) and refreshed at the final step (Step 4), exploiting stable intermediate routing while preserving final-step fidelity.

with complexity $O(HL_QL_Kd)$. Since L_K grows approximately linearly with rollout length, dense KV-cache attention becomes the dominant inference bottleneck in long-horizon generation.

To address this issue, we propose **A^{2}R**, a training-free sparse attention framework built on three components: (i) an *anchored dense-to-sparse transition*, which preserves the initial chunk as a persistent visual anchor; (ii) *per-head adaptive block routing*, which allocates non-uniform sparsity across heads and query blocks; and (iii) *reusable sparse patterns*, which cache and reuse sparse routing results across intermediate denoising steps. A^{2}R preserves the first chunk under dense attention, constructs an adaptive sparse pattern over the growing KV cache using coarse block-level routing and executes exact sparse attention over the retained blocks with a custom Triton kernel. An overview is shown in Figure 3.

3.2 Anchored Dense-to-Sparse Transition

The initial chunk in autoregressive video diffusion plays a disproportionate role in establishing scene structure, object identity and coarse appearance for later generation. Sparsifying this chunk can degrade downstream generation quality. We therefore treat the first chunk as a *persistent visual anchor*: it is always processed with dense attention, while all later chunks are processed with sparse attention.

Formally, let $\mathbf{O}^{(c)}$ denote the attention output for chunk c. We define

$$\mathbf{O}^{(c)} = \begin{cases} \mathrm{Attn}(\mathbf{Q}^{(c)}, \mathbf{K}^{(c)}, \mathbf{V}^{(c)}), & c = 0, \\ \mathrm{SparseAttn}(\mathbf{Q}^{(c)}, \mathbf{K}^{(\le c)}, \mathbf{V}^{(\le c)}), & c \ge 1, \end{cases} \tag{2}$$

where $\mathbf{K}^{(\le c)}$ and $\mathbf{V}^{(\le c)}$ denote the accumulated KV cache up to chunk c.

3.3 Per-Head Adaptive Block Routing

To determine which historical context should be retained under sparse attention, A^{2}R performs coarse block-level routing. We partition the query and key sequences into contiguous blocks of size B_Q and B_K, respectively and define

$$Q_{\mathrm{blk}} = \left\lceil \frac{L_Q}{B_Q} \right\rceil, \qquad K_{\mathrm{blk}} = \left\lceil \frac{L_K}{B_K} \right\rceil. \tag{3}$$

For each head h, query block i and key block j, we compute mean-pooled block representations:

$$\bar{\mathbf{Q}}_{h,i} = \frac{1}{|\mathcal{B}_i^Q|} \sum_{u \in \mathcal{B}_i^Q} \mathbf{Q}_{h,u}, \qquad \bar{\mathbf{K}}_{h,j} = \frac{1}{|\mathcal{B}_j^K|} \sum_{v \in \mathcal{B}_j^K} \mathbf{K}_{h,v}, \tag{4}$$

where $\mathcal{B}_i^Q$ and $\mathcal{B}_j^K$ are the token index sets of the i-th query block and the j-th key block, respectively. We then compute a coarse block score tensor

$$\mathbf{S}_{h,i,j} = \frac{\bar{\mathbf{Q}}_{h,i}^\top \bar{\mathbf{K}}_{h,j}}{\sqrt{d}}, \tag{5}$$

which measures the coarse relevance between query block i and key block j for head h. Because $Q_{\text{blk}} \ll L_Q$ and $K_{\text{blk}} \ll L_K$, this routing stage is substantially cheaper than token-level dense attention. We further normalize the coarse scores over all key blocks:

$$\mathbf{a}_{h,i,:} = \text{Softmax}(\mathbf{S}_{h,i,:}), \tag{6}$$

yielding a block-level relevance distribution for each pair (h, i).

A fixed top-K sparsity ratio is often suboptimal because attention concentration varies substantially across heads and query blocks. Instead, A^{2}R adaptively selects the smallest set of key blocks that preserves a target fraction of coarse relevance mass. For each pair (h, i), we solve

$$\mathcal{I}_{h,i} = \arg \min_{\mathcal{I} \subseteq \{1,\dots,K_{\text{blk}}\}} |\mathcal{I}| \quad \text{s.t.} \quad \sum_{j \in \mathcal{I}} \mathbf{a}_{h,i,j} \geq \tau, \tag{7}$$

where $\tau \in (0, 1)$ is a relevance-preservation threshold. In practice, we solve equation 7 by sorting the relevance scores in descending order:

$$\mathbf{p}^{(h,i)}, \boldsymbol{\pi}^{(h,i)} = \text{SortDesc}\big(\mathbf{a}_{h,i,:}\big), \tag{8}$$

where $\mathbf{p}_1^{(h,i)} \geq \cdots \geq \mathbf{p}_{K_{\text{blk}}}^{(h,i)}$ and $\boldsymbol{\pi}^{(h,i)}$ stores the corresponding original key-block indices. The retained block count is

$$N^{(h,i)} = \min \left\{ k \in \{1, \dots, K_{\text{blk}}\} : \sum_{r=1}^{k} \mathbf{p}_r^{(h,i)} \geq \tau \right\}, \tag{9}$$

and the selected block set is

$$\mathcal{I}_{h,i} = \left\{ \boldsymbol{\pi}_r^{(h,i)} : 1 \leq r \leq N^{(h,i)} \right\}. \tag{10}$$

Here, r denotes the rank in the sorted list and $\boldsymbol{\pi}_r^{(h,i)}$ is the original key-block index of the r-th largest relevance score. Unless otherwise specified, we set $\tau = 0.85$. The visualization of this adaptive selection strategy is illustrated in Figure 4 and Appendix E. Compared with a fixed uniform sparsity rule, adaptive block routing better captures the non-uniform structure of block relevance and therefore uses the sparse budget more efficiently.

Given the selected block sets, we store the sparse pattern in a padded lookup table

$$\text{LUT} \in \mathbb{Z}^{H \times Q_{\text{blk}} \times K_{\text{blk}}}, \tag{11}$$

where unused entries are filled with a padding sentinel.

3.4 Reusable Sparse Patterns

A key observation behind A^{2}R is that routing patterns remain stable across non-final denoising steps for the same chunk. In our Self-Forcing setting, each chunk is denoised for five steps. The first four steps correspond to relatively similar time conditions, so their coarse attention routing structures remain highly similar. In contrast, the final denoising step corresponds to $t = 0$, where the model produces the clean frame that is written to the KV cache for subsequent chunks. As a result, this step is more sensitive to precise attention refinement: any routing error at $t = 0$ is directly written into the cache and can propagate through later autoregressive generation. We therefore construct the lookup table once, reuse it across Steps 0–3 and refresh the sparse pattern at Step 4. This behavior is also consistent with the attention visualization in Figure 2, where Steps 0–3 exhibit highly similar patterns while the final step shows more noticeable refinement. In practice, this partial reuse strategy exploits stable intermediate routing while preserving final-step fidelity.

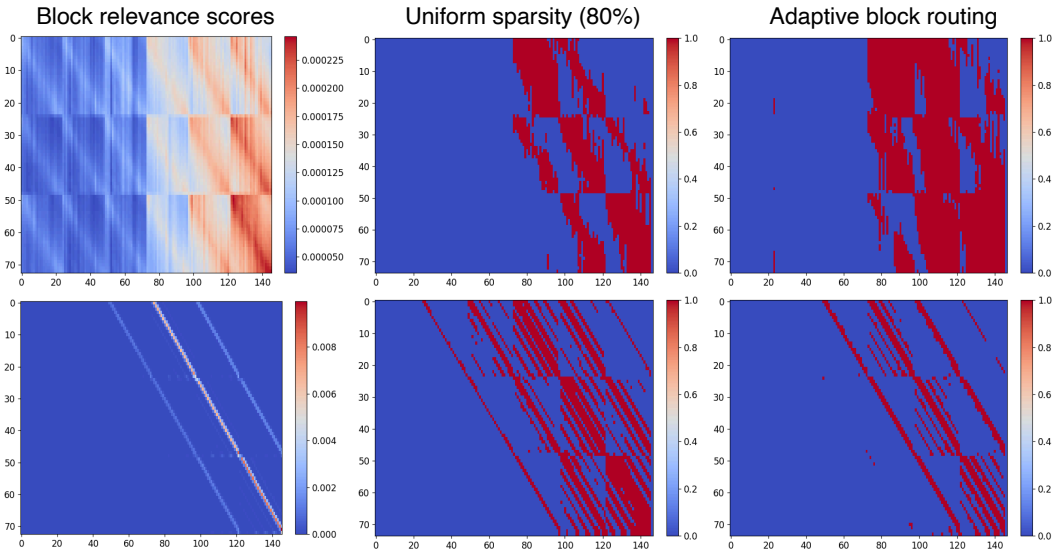

Figure 4: Visualization of adaptive block routing. Left: coarse block relevance scores. Middle: a uniform sparsity rule with a fixed sparsity ratio (80%). Right: the sparse pattern selected by adaptive block routing. Because block relevance is highly non-uniform, a fixed uniform sparsity rule cannot faithfully reflect the underlying attention structure. Adaptive block routing instead allocates the sparse budget according to relevance mass, retaining high-value blocks while pruning less informative ones.

3.5 Exact Sparse Attention Execution

Given the retained key-block indices, A^{2}R performs exact attention on the original unpooled tensors within the selected support while skipping all pruned blocks. Let $\mathcal{I}_{h,i}$ denote the retained key-block set for query block i in head h. Sparse attention is evaluated as

$$\mathrm{SparseAttn}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathrm{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}_{\mathcal{I}}^{\top}}{\sqrt{d}}\right)\mathbf{V}_{\mathcal{I}}, \tag{12}$$

where $\mathbf{K}_{\mathcal{I}}$ and $\mathbf{V}_{\mathcal{I}}$ denote the gathered key and value tokens associated with the retained block indices. The computation is exact over the retained tokens; the only approximation lies in the routing stage used to determine which blocks to keep.

We implement this stage using a custom Triton kernel in FP16/BF16. The kernel is launched over a 1D grid indexed by the query-block index $m \in \{0,\dots,Q_{\mathrm{blk}}-1\}$ for each head. Each thread block loads one query block and iterates only over the retained key blocks stored in $\mathrm{LUT}[h,m,:]$, while maintaining numerically stable online softmax accumulation. For simplicity, we omit the batch dimension here; in implementation, the batch and head dimensions can be flattened together. Additional implementation details and pseudo code are provided in Appendix G.

4 Experiments

4.1 Experimental Setup

Backbones and inference setting. We evaluate A^{2}R on autoregressive video diffusion models that generate videos chunk by chunk with KV caching. Unless otherwise specified, we preserve dense attention for the first chunk and apply A^{2}R to all subsequent chunks. For adaptive block routing, we use block sizes $B_Q = B_K = 64$ and a relevance-preservation threshold of $\tau = 0.85$. Sparse routing results are cached and reused across nearby denoising steps.

Metrics. We evaluate both generation quality and efficiency. For generation quality, we report standard video-generation metrics such as VBench Huang et al. (2024) and its sub-dimensions, including semantic alignment, perceptual quality and temporal consistency. For efficiency, we measure end-to-end latency and attention-only runtime. We additionally report long-horizon generation results to assess robustness under growing KV-cache context.

Table 1: Main results on autoregressive video diffusion. A^{2}R provides the best overall quality–efficiency trade-off among the compared methods, substantially reducing latency while preserving strong generation quality. Since our focus is inference-time attention acceleration, we evaluate SLA using only its sparse-attention component under the same autoregressive inference setting.

Method	Latency↓	Speedup↑	VBench Dimensions						Overall Scores		
			Aesthetic Quality↑	Imaging Quality↑	Motion Smoothness↑	Dynamic Degree↑	Subject Consistency↑	Background Consistency↑	Quality Score↑	Semantic Score↑	Total Score↑
Self-Forcing 1.3B (fps = 16)											
FlashAttention2 Dao (2024)	6.11	1.00×	67.3	70.1	98.2	63.4	95.2	96.4	84.7	81.1	84.0
STA Zhang et al. (2025d)	5.78	1.06×	64.7	71.5	98.4	49.3	96.1	96.8	83.5	81.0	83.0
Radial Li et al. (2025)	4.74	1.29×	46.2	66.4	96.1	68.1	90.5	93.8	79.0	54.1	74.0
SVG2 Yang et al. (2025)	13.58	0.45×	65.8	68.4	97.6	62.5	93.4	95.4	83.7	78.4	82.6
SLA Zhang et al. (2025a)	4.93	1.24×	66.5	69.9	98.1	44.7	95.4	96.6	82.9	80.0	82.3
Light Forcing Lv et al. (2026)	4.66	1.31×	66.9	71.0	98.2	62.9	95.3	96.6	84.2	80.7	83.5
A^{2}R	**4.59**	**1.33**×	67.4	71.1	98.4	63.1	95.3	96.6	84.6	80.9	**83.8**
LongLive 1.3B (fps = 16)											
FlashAttention2 Dao (2024)	6.65	1.00×	68.5	69.5	98.7	39.6	96.9	97.1	83.8	80.6	83.1
STA Zhang et al. (2025d)	6.05	1.10×	65.8	71.0	98.9	23.1	97.2	97.6	82.9	81.4	82.7
Radial Li et al. (2025)	5.68	1.17×	55.4	71.8	98.1	25.4	77.9	89.2	75.2	66.9	73.5
SVG2 Yang et al. (2025)	13.85	0.48×	66.5	67.2	98.4	34.8	95.1	95.9	82.6	78.5	81.8
SLA Zhang et al. (2025a)	5.32	1.25×	66.6	69.7	98.2	34.6	95.5	96.5	82.8	80.3	82.3
Light Forcing Lv et al. (2026)	5.37	1.24×	67.3	69.4	98.2	39.0	96.5	96.6	83.5	80.2	82.8
A^{2}R	**5.12**	**1.30**×	67.4	69.4	98.3	39.0	96.8	96.8	83.7	80.4	**83.0**

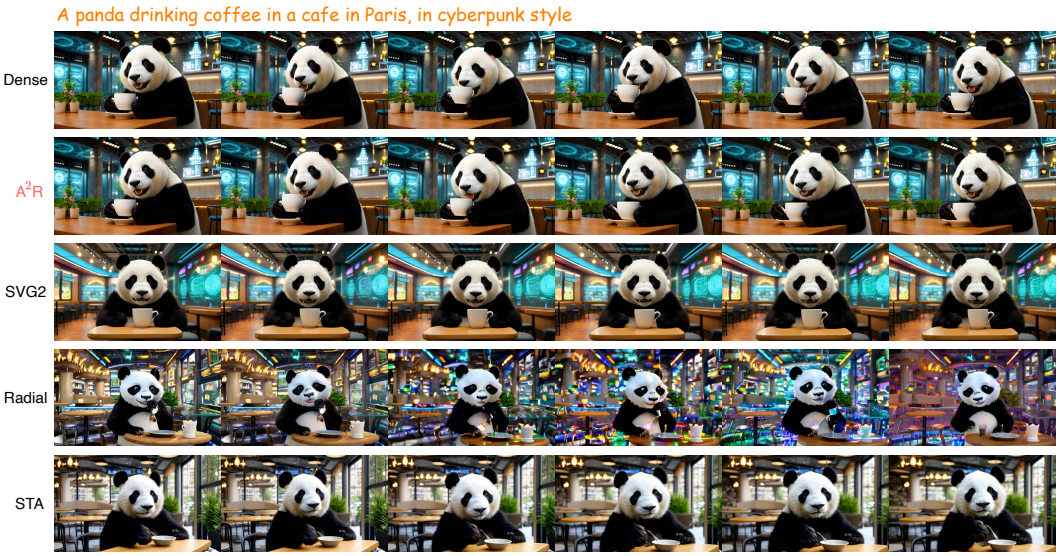

Figure 5: Qualitative comparison with other sparse-attention methods. Compared with static or heuristic sparse-attention baselines, our A^{2}R produces results that are visually closer to dense attention, better preserving subject identity, scene details and the intended cyberpunk style.

Implementation. A^{2}R is implemented as a training-free inference module with a custom Triton-based sparse attention kernel. All experiments are conducted on NVIDIA H800 GPUs with 80GB memory. Additional details are provided in Appendix B.

4.2 Main Results

Table 1 compares A^{2}R with dense autoregressive inference and representative efficient-attention baselines. A^{2}R achieves a strong quality–efficiency trade-off, substantially reducing the cost of attention while preserving strong visual quality and temporal consistency. Compared with fixed-ratio sparse baselines, A^{2}R benefits from adaptive block routing, which allocates non-uniform sparsity across heads and query blocks instead of enforcing a uniform sparsity budget. These results show that sparse context selection in autoregressive video diffusion can be effective when it is anchored, adaptive and reusable, rather than uniformly imposed across all chunks and denoising steps.

4.3 Qualitative Comparison and Ablation

A^{2}R remains visually closer to dense attention than the sparse baselines while using sparse attention itself (Figure 5). Under the prompt *"A panda drinking coffee in a cafe in Paris, in cyberpunk style,"* it better preserves subject identity, scene structure and the intended cyberpunk style than SVG2 Yang et al. (2025), Radial Li et al. (2025) and STA Zhang et al. (2025d). This suggests that the sparse

Table 2: Ablation study of A^{2}R under the Self-Forcing 1.3B setting. The full design provides the best quality–efficiency trade-off and its anchored, adaptive and reusable components are complementary.

Anchor	Adaptive	Reuse	Latency (s)↓	Speedup↑	Quality↑	Semantic↑	Total↑
-	-	-	6.11	1.00×	84.7	81.1	84.0
-	✓	✓	4.58	1.33×	83.4	80.1	82.7
✓	-	✓	4.55	1.34×	83.6	80.4	83.0
✓	✓	-	4.83	1.26×	84.6	**81.0**	**83.9**
✓	✓	✓	**4.59**	**1.33**×	**84.6**	80.9	83.8

Table 3: Sensitivity to the relevance-preservation threshold τ in A^{2}R under the Self-Forcing 1.3B setting.

τ	Sparsity	Latency (s)↓	Speedup↑	Total↑
0.70	87%	4.18	1.46×	82.9
0.80	82%	4.39	1.39×	83.5
0.85	77%	4.59	1.33×	83.8
0.90	72%	4.86	1.26×	83.8
0.95	69%	5.27	1.16×	**83.9**

Table 4: Ablation of sparse-pattern reuse in A^{2}R under the Self-Forcing 1.3B setting. "0–3" denotes Steps 0 to 3.

Reuse Strategy	Latency (s)↓	Speedup↑	Total↑
No reuse	4.83	1.26×	83.9
Attn. output (0–3)	4.35	1.40×	82.5
Sparse pattern (0–2)	4.67	1.31×	**83.8**
Sparse pattern (0–3)	4.59	1.33×	**83.8**
Sparse pattern (0–4)	**4.51**	**1.35**×	83.0

patterns selected by A^{2}R retain informative historical context more effectively than static or heuristic sparsity schemes.

The qualitative ablation further illustrates the role of each design component (Figure 6). Removing the anchor leads to severe corruption, confirming that early chunks should be preserved as a persistent visual anchor. Adding the anchor improves stability, but uniform sparsity still causes visible degradation, indicating that a fixed sparsity budget cannot adequately capture the underlying attention structure. Replacing uniform sparsity with per-head adaptive routing produces results much closer to dense attention, while adding sparse-pattern reuse yields almost no visible quality drop. Additional qualitative visualizations and failure cases are provided in Appendix F and Appendix H.

The quantitative ablation in Table 2 is consistent with these observations. Removing the anchor degrades quality and semantic consistency, confirming the importance of preserving foundational context. Replacing adaptive routing with a non-adaptive design further weakens quality, indicating that non-uniform sparsity across heads and query blocks is beneficial. Disabling reuse has little effect on quality but increases latency, showing that sparse-pattern reuse mainly contributes to efficiency. Overall, the full A^{2}R design provides the strongest quality–efficiency trade-off among the ablated variants. We provide additional ablations on block size and anchor variants in Appendix D.1 and Appendix D.2.

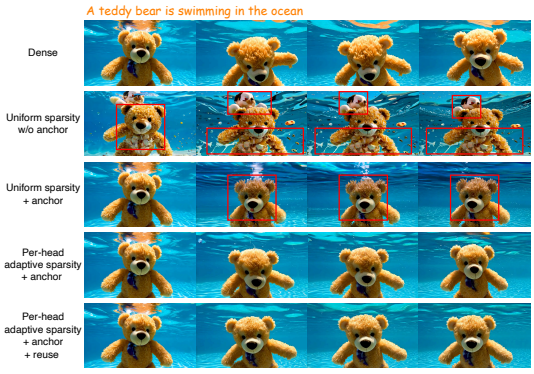

Figure 6: Qualitative ablation of A^{2}R on autoregressive video generation.

A^{2}R is robust to the choice of the relevance-preservation threshold τ, although the quality–efficiency trade-off still depends on how aggressively KV blocks are pruned (Table 3). As expected, larger τ retains more KV blocks on average, reducing sparsity and speedup while slightly improving quality. Very small thresholds such as $\tau = 0.70$ are more efficient but noticeably harm quality, whereas larger thresholds such as 0.90 and 0.95 bring little additional gain. Overall, $\tau = 0.85$ provides the best balance and is used by default.

Only the routing structure, rather than the attention output itself, can be safely reused across denoising steps (Table 4). Directly reusing the attention output reduces latency but causes a clear quality drop, indicating that hidden states continue to evolve even when routing remains stable. In contrast, reusing only the sparse routing pattern over Steps 0–3 preserves the quality of the no-reuse baseline while improving efficiency. Extending reuse to the final denoising step slightly lowers latency but degrades quality, confirming that the last step requires refreshed routing.

4.4 Efficiency Analysis

As rollout proceeds, dense autoregressive decoding becomes increasingly attention-dominated because the KV cache grows monotonically (Figure 7). A^{2}R suppresses this growth by pruning low-value KV blocks and executing attention only on retained blocks, consistently reducing overall latency across rollout lengths. Across the tested settings, A^{2}R reduces the attention component by about 2.1–2.3×, while the end-to-end speedup grows from about 1.3× to 1.5×. This shows that the gain mainly comes from accelerating KV-cache attention and becomes more pronounced as rollout proceeds.

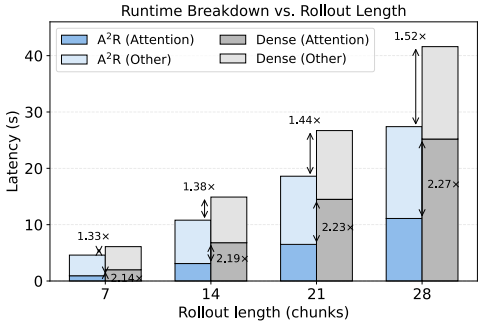

Figure 7: Runtime breakdown versus rollout length for dense attention and A^{2}R.

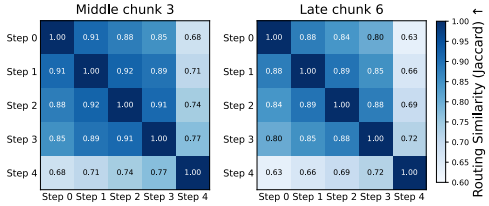

Figure 8: Pairwise Jaccard similarity of sparse routing patterns across denoising steps for representative middle and late autoregressive chunks.

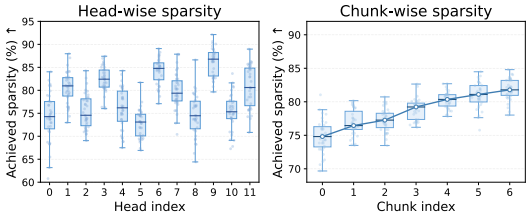

Figure 9: Head-wise and chunk-wise achieved sparsity in A^{2}R. Sparsity varies substantially across attention heads and also changes over autoregressive chunks.

4.5 Routing and Sparsity

Sparse routing remains highly stable across non-final denoising steps, which explains why routing patterns can be reused effectively. To quantify this effect, we measure the Jaccard similarity between routing patterns constructed at different steps for the same autoregressive chunk, averaging over heads, query blocks and layers. The reported routing statistics are further averaged over the evaluation set. The resulting similarity matrices (Figure 8) show consistently high agreement across Steps 0–3, whereas similarity involving Step 4 is noticeably lower. This supports reusing routing for intermediate steps while refreshing it at the final denoising step. We also analyze routing stability within each layer in Appendix D.3.

The achieved sparsity in A^{2}R is also clearly adaptive rather than fixed (Figure 9). The head-wise distributions reveal substantial heterogeneity, showing that different heads place attention mass on very different numbers of KV blocks. The chunk-wise distributions further indicate that the effective sparsity level evolves over rollout rather than remaining constant. These observations explain why fixed top-K or globally uniform sparsity budgets are limited: they cannot capture the head-dependent and chunk-dependent variability naturally modeled by A^{2}R.

5 Conclusion

We present **A^{2}R**, a training-free sparse attention framework for autoregressive video diffusion. Motivated by persistent early-chunk dependence, head-wise sparsity heterogeneity and stable intermediate routing, A^{2}R combines an anchored dense-to-sparse transition, per-head adaptive block routing and reusable sparse patterns with an exact Triton-based sparse attention kernel. Experiments show that A^{2}R achieves a strong quality–efficiency trade-off, substantially reducing KV-cache attention cost and inference latency while preserving visual quality and temporal consistency close to dense attention. Our ablations further support the complementary roles of anchoring, adaptive routing and partial sparse-pattern reuse. More broadly, our results suggest that sparse attention for autoregressive video diffusion should be rollout-aware rather than static or uniform. A limitation of the current study is that it focuses on inference-time sparsification on two autoregressive video-diffusion backbones; evaluating broader architectures, resolutions and longer rollout regimes is an important direction for future work. We hope A^{2}R provides a useful building block for future long-horizon autoregressive video generation.

REFERENCES

Roman Bachmann, Jesse Allardice, David Mizrahi, Enrico Fini, Oğuzhan Fatih Kar, Elmira Amirloo, Alaaeldin El-Nouby, Amir Zamir, and Afshin Dehghan. Flextok: Resampling images into 1d token sequences of flexible length. *Proceedings of the International Conference on Machine Learning*, 2025.

Fan Bao, Chongxuan Li, Yue Cao, and Jun Zhu. All are worth words: A vit backbone for score-based diffusion models. *NeurIPS Workshop on Score-Based Methods*, 2022.

Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. In *Advances in Neural Information Processing Systems*, volume 37, pp. 24081–24125, 2024.

Junsong Chen, Yuyang Zhao, Jincheng Yu, Ruihang Chu, Junyu Chen, Shuai Yang, Xianbang Wang, Yicheng Pan, Daquan Zhou, Huan Ling, et al. Sana-video: Efficient video generation with block linear diffusion transformer. *arXiv preprint arXiv:2509.24695*, 2025.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations*, 2024.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems*, volume 35, pp. 16344–16359, 2022.

Yuchao Gu, Weijia Mao, and Mike Zheng Shou. Long-context autoregressive video modeling with next-frame prediction. *arXiv preprint arXiv:2503.19325*, 2025.

Yoav HaCohen, Nisan Chiprut, Benny Brazowski, Daniel Shalem, Dudu Moshe, Eitan Richardson, Eran Levin, Guy Shiran, Nir Zabari, Ori Gordon, et al. Ltx-video: Realtime video latent diffusion. *arXiv preprint arXiv:2501.00103*, 2024.

Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022a.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022b.

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. *arXiv preprint arXiv:2506.08009*, 2025.

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

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H Abdi, Dongsheng Li, Chin-Yew Lin, et al. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. *Advances in Neural Information Processing Systems*, 37:52481–52515, 2024.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pp. 5156–5165, 2020.

Yeongmin Kim, Sotiris Anagnostidis, Yuming Du, Edgar Schönfeld, Jonas Kohler, Markos Georgopoulos, Albert Pumarola, Ali Thabet, and Artsiom Sanakoyeu. Autoregressive distillation of diffusion transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15745–15756, 2025.

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.

Seon-Ho Lee, Jue Wang, Zhikang Zhang, David Fan, and Xinyu Li. Video token merging for long-form video understanding. *Advances in Neural Information Processing Systems*, 2024.

Xingyang Li, Muyang Li, Tianle Cai, Haocheng Xi, Shuo Yang, Yujun Lin, Lvmin Zhang, Songlin Yang, Jinbo Hu, Kelly Peng, et al. Radial attention: O(n log n) sparse attention with energy decay for long video generation. *arXiv preprint arXiv:2506.19852*, 2025.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *Advances in Neural Information Processing Systems*, 36:52342–52364, 2023.

Chengtao Lv, Yumeng Shi, Yushi Huang, Ruihao Gong, Shen Ren, and Wenya Wang. Light forcing: Accelerating autoregressive video diffusion via sparse attention. *arXiv preprint arXiv:2602.04789*, 2026.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023.

Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, et al. Movie gen: A cast of media foundation models. *arXiv preprint arXiv:2410.13720*, 2024.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjorn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, et al. Make-a-video: Text-to-video generation without text-video data. *arXiv preprint arXiv:2209.14792*, 2022.

Wan Team, 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.

Haoyu Wu, Jingyi Xu, Hieu Le, and Dimitris Samaras. Importance-based token merging for efficient image and video generation. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4983–4995, 2025a.

Jie Wu, Lin Hou, Hao Yang, Xin Tao, Yuandong Tian, Pengfei Wan, Di Zhang, and Yuheng Tong. Vmoba: Mixture-of-block attention for video diffusion models. *arXiv preprint arXiv:2506.23858*, 2025b.

Haocheng Xi, Shuo Yang, Yilong Zhao, Chenfeng Xu, Muyang Li, Xiuyu Li, Yujun Lin, Han Cai, Jintao Zhang, Dacheng Li, et al. Sparse videogen: Accelerating video diffusion transformers with spatial-temporal sparsity. In *Proceedings of the International Conference on Machine Learning*, 2025.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. Infllm: Training-free long-context extrapolation for llms with an efficient context memory. *Advances in Neural Information Processing Systems*, 37:119638–119661, 2024.

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

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Shang Yang, Haotian Tang, Yao Fu, Song Han, et al. Duoattention: Efficient long-context llm inference with retrieval and streaming heads. In *International Conference on Learning Representations*, 2025.

Enze Xie, Junsong Chen, Junyu Chen, Han Cai, Haotian Tang, Yujun Lin, Zhekai Zhang, Muyang Li, Ligeng Zhu, Yao Lu, et al. Sana: Efficient high-resolution image synthesis with linear diffusion transformers. *arXiv preprint arXiv:2410.10629*, 2024.

Boxun Xu, Yuming Du, Zichang Liu, Siyu Yang, Ziyang Jiang, Siqi Yan, Rajasi Saha, Albert Pumarola, Wenchen Wang, and Peng Li. Sparse forcing: Native trainable sparse attention for real-time autoregressive diffusion video generation. *arXiv preprint arXiv:2604.21221*, 2026.

Shuo Yang, Haocheng Xi, Yilong Zhao, Muyang Li, Jintao Zhang, Han Cai, Yujun Lin, Xiuyu Li, Chenfeng Xu, Kelly Peng, et al. Sparse videogen2: Accelerate video generation with sparse attention via semantic-aware permutation. *arXiv preprint arXiv:2505.18875*, 2025.

Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Fredo Durand, and Bill Freeman. Improved distribution matching distillation for fast image synthesis. In *Advances in Neural Information Processing Systems*, volume 37, pp. 47455–47487, 2024a.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6613–6623, 2024b.

Tianwei Yin, Qiang Zhang, Richard Zhang, William T Freeman, Fredo Durand, Eli Shechtman, and Xun Huang. From slow bidirectional to fast autoregressive video diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22963–22974, 2025.

Jintao Zhang, Haoxu Wang, Kai Jiang, Shuo Yang, Kaiwen Zheng, Haocheng Xi, Ziteng Wang, Hongzhou Zhu, Min Zhao, Ion Stoica, et al. Sla: Beyond sparsity in diffusion transformers via fine-tunable sparse-linear attention. *arXiv preprint arXiv:2509.24006*, 2025a.

Jintao Zhang, Chendong Xiang, Haofeng Huang, Jia Wei, Haocheng Xi, Jun Zhu, and Jianfei Chen. Spargeattention: Accurate and training-free sparse attention accelerating any model inference. *arXiv preprint arXiv:2502.18137*, 2025b.

Peiyuan Zhang, Yongqi Chen, Haofeng Huang, Will Lin, Zhengzhong Liu, Ion Stoica, Eric Xing, and Hao Zhang. Vsa: Faster video diffusion with trainable sparse attention. *arXiv preprint arXiv:2505.13389*, 2025c.

Peiyuan Zhang, Yongqi Chen, Runlong Su, Hangliang Ding, Ion Stoica, Zhengzhong Liu, and Hao Zhang. Fast video generation with sliding tile attention. *arXiv preprint arXiv:2502.04507*, 2025d.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in neural information processing systems*, 36:34661–34710, 2023.

Tianchen Zhao, Tongcheng Fang, Haofeng Huang, Rui Wan, Widyadewi Soedarmadji, Enshu Liu, Shiyao Li, Zinan Lin, Guohao Dai, Shengen Yan, et al. Vidit-q: Efficient and accurate quantization of diffusion transformers for image and video generation. In *International Conference on Learning Representations*, 2025.

Zangwei Zheng, Xiangyu Peng, Tianji Yang, Chenhui Shen, Shenggui Li, Hongxin Liu, Yukun Zhou, Tianyi Li, and Yang You. Open-sora: Democratizing efficient video production for all. *arXiv preprint arXiv:2412.20404*, 2024.

A Additional Theoretical Clarifications

A.1 Optimality of cumulative-mass block selection

The adaptive routing rule in A^{2}R selects, for each head–query-block pair (h, i), a minimum-cardinality set of key blocks whose cumulative relevance mass reaches a threshold τ, where $0 < \tau \leq 1$.

Proposition 1. Let $\mathbf{a}_{h,i,:} \in \mathbb{R}^{K_{\text{blk}}}$ be a nonnegative relevance distribution with

$$\sum_{j=1}^{K_{\text{blk}}} \mathbf{a}_{h,i,j} = 1.$$

Consider

$$\mathcal{I}^{\star}_{h,i} = \arg \min_{\mathcal{I} \subseteq \{1,\ldots,K_{\text{blk}}\}} |\mathcal{I}| \quad \text{s.t.} \quad \sum_{j \in \mathcal{I}} \mathbf{a}_{h,i,j} \geq \tau. \tag{13}$$

If

$$\mathbf{p}^{(h,i)}_1 \geq \mathbf{p}^{(h,i)}_2 \geq \cdots \geq \mathbf{p}^{(h,i)}_{K_{\text{blk}}}$$

are the entries of $\mathbf{a}_{h,i,:}$ sorted in descending order, then an optimal solution is obtained by taking the shortest prefix whose cumulative mass is at least τ, i.e.,

$$N^{(h,i)} = \min \left\{ k : \sum_{r=1}^{k} \mathbf{p}^{(h,i)}_r \geq \tau \right\}.$$

Proof. Among all subsets of size k, the maximum achievable relevance mass is obtained by choosing the top-k entries. Indeed, if a subset contains an entry with smaller relevance than some excluded entry, swapping them preserves cardinality and does not decrease the total mass. Repeating this exchange yields the top-k set. Therefore, there exists a subset of size k satisfying equation 13 if and only if the top-k prefix does. The minimum feasible cardinality is the smallest prefix length whose cumulative mass is at least τ. Ties may admit multiple optimal sets. This statement concerns the pooled relevance distribution; it does not guarantee retention of the same fraction of token-level dense-attention mass.

A.2 Complexity analysis

Dense KV-cache attention has per-step complexity

$$O(HL_QL_Kd), \tag{14}$$

where H is the number of heads, L_Q is the query length, L_K is the cached key length and d is the per-head hidden dimension.

In A^{2}R, block-level routing operates on pooled query and key blocks. Its score computation costs

$$O(HQ_{\text{blk}}K_{\text{blk}}d), \tag{15}$$

where

$$Q_{\text{blk}} = \left\lceil \frac{L_Q}{B_Q} \right\rceil, \qquad K_{\text{blk}} = \left\lceil \frac{L_K}{B_K} \right\rceil.$$

Sorting the K_{blk} relevance scores for each (h, i) pair adds

$$O(HQ_{\text{blk}}K_{\text{blk}} \log K_{\text{blk}}). \tag{16}$$

Let $\bar{L}_K$ denote the average number of retained key tokens per query token after routing. The sparse execution stage then costs

$$O(HL_Q\bar{L}_Kd), \tag{17}$$

which is substantially smaller than equation 14 whenever $\bar{L}_K \ll L_K$.

Routing reuse further reduces routing overhead. If each sparse chunk has $T \geq 2$ attention steps, recomputing routing at every evaluation costs

$$T\, C_{\text{route}},$$

Table 5: Additional sparse-attention comparisons. Speedups are relative to dense inference on the same backbone. LF-attn denotes the attention-only Light Forcing adaptation.

Backbone	Method	Sparsity (%)	Total	Speedup
Self-forcing	Dense	0	84.0	1.00×
Self-forcing	LF-attn	78	83.5	1.31×
Self-forcing	A^2R	77	83.8	1.33×
LongLive	Dense	0	83.1	1.00×
LongLive	LF-attn	70	82.8	1.24×
LongLive	A^2R	72	83.0	1.30×

where C_{route} is the cost of constructing the sparse pattern once. With the partial reuse strategy in A^2R, routing is constructed once for non-final steps and refreshed once at the final step, reducing this to

$$2\,C_{\text{route}}.$$

In the reported Self-Forcing setting, $T = 5$, so reuse reduces routing overhead from $5C_{\text{route}}$ to $2C_{\text{route}}$ per sparse chunk under the equal-rebuild-cost approximation: a 60% reduction, or a $2.5\times$ speedup of routing alone. The dense initial chunk does not construct sparse routing. Appendix G.1 distinguishes sparse-evaluation and full-rollout accounting.

B Additional Experimental Details

B.1 Backbones and inference settings

We evaluate A^2R on Self-Forcing 1.3B and LongLive 1.3B. The default configuration uses a resolution of 832×480, 16 FPS, 3 frames per chunk, and 7 chunks. The first chunk uses dense attention, and sparse routing is applied to subsequent chunks. Unless otherwise specified, the query and key block sizes are $B_Q = B_K = 64$ and the cumulative-mass threshold is $\tau = 0.85$.

For Self-Forcing, the routing schedule indexes five attention steps as $0, \ldots, 4$. Routing is constructed at evaluation 0, reused at steps 1–3, and refreshed at evaluation 4. Reuse applies to the selected support. Current-chunk projections and attention outputs are recomputed, while historical keys and values are read from the backbone's KV cache.

B.2 Implementation and hardware details

A^2R is a training-free inference module with a custom Triton sparse attention kernel. The kernel operates in FP16/BF16 and evaluates attention over retained, unpooled key and value tokens. Batch and head dimensions can be flattened for kernel execution.

The hardware platform is NVIDIA H800 with 80GB memory. The software stack uses PyTorch 2.10.0, CUDA 13.0, and Triton 3.6.0. The low-precision composition experiments in Appendix G.2 additionally consider FP8 inference.

C Additional Experimental Results

C.1 Additional sparse-attention comparisons

Table 5 compares dense attention, an attention-only adaptation of Light Forcing (LF-attn), and A^2R on Self-Forcing and LongLive. LF-attn applies the sparse-attention component of Light Forcing to each backbone without applying the full method's training or distillation procedure.

C.2 Long-rollout quality and efficiency

Table 6 extends the LongLive evaluation to 30-second generation. Across these operating points, average sparsity increases from 72% to 85%, and speedup increases from $1.30\times$ to $1.56\times$. The

Table 6: LongLive results across rollout durations. Each speedup uses the dense baseline at the corresponding duration.

Duration (s)	Sparsity (%)	Speedup	Total
5	72	1.30×	83.0
10	74	1.34×	82.8
15	78	1.40×	82.7
20	81	1.48×	82.6
30	85	1.56×	82.6

Table 7: Block-size sensitivity on Self-Forcing 1.3B. Shading identifies the default configuration.

B_Q	B_K	Latency (s)↓	Avg. Sparsity	Total↑
32	32	4.78	74%	83.8
64	64	4.59	77%	83.8
64	128	4.41	82%	82.9

aggregate score changes from 83.0 to 82.6. These measurements characterize the quality and efficiency of A^2R as rollout duration increases.

D Additional Ablation Studies

D.1 Block size sensitivity

Table 7 compares three block sizes on Self-Forcing. Moving from 32×32 to 64×64 reduces latency from 4.78 to 4.59s at the same displayed aggregate score. Increasing the key block size to 128 further reduces latency to 4.41s, with a lower aggregate score of 82.9. We use 64×64 as the default operating point. Block size affects both the approximation and execution efficiency, so its effect is assessed jointly with the achieved sparsity.

D.2 Additional anchor variants

Table 8 compares dense initialization policies. Using the first chunk as an anchor improves both quality and semantic scores relative to the no-anchor configuration. Extending dense initialization to two chunks gives the same displayed component scores and increases latency from 4.59 to 4.82s. The default therefore uses one dense initial chunk. Dense initialization and mandatory retention of initial-chunk KV during later steps are distinct operations; Appendix D.6 discusses their implications for the attention budget.

Table 8: Dense initialization policies on Self-Forcing 1.3B. Shading identifies the default.

Anchor strategy	Latency (s)	Sparsity (%)	Quality	Semantic
No anchor	4.58	80	83.4	80.1
First chunk dense	4.59	77	84.6	80.9
First two chunks dense	4.82	71	84.6	80.9

D.3 Layer-wise routing stability

For two sets of retained blocks $\mathcal{I}$ and $\mathcal{J}$, routing similarity is measured by their Jaccard index,

$$J(\mathcal{I}, \mathcal{J}) = \frac{|\mathcal{I} \cap \mathcal{J}|}{|\mathcal{I} \cup \mathcal{J}|}.$$

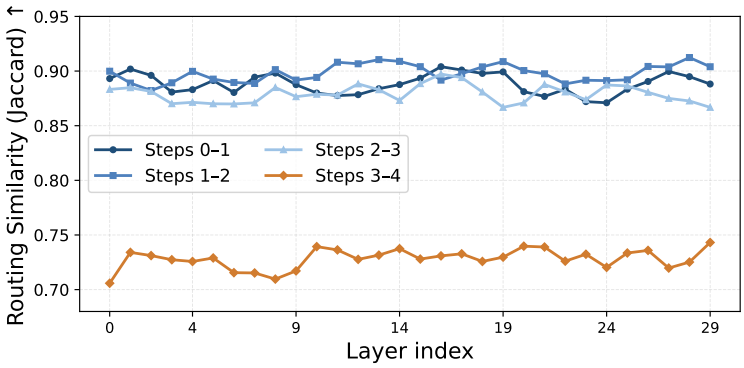

Figure 10: Layer-wise routing stability measured by Jaccard similarity between sparse routing patterns at adjacent denoising steps.

Table 9: Static allocation controls on Self-Forcing. These are internal controls, not implementations of an online adaptive-search baseline.

Allocation	Sparsity (%)	Latency (s)	Total
Static per-head	77	4.72	82.9
Static per-(head,layer)	77	4.77	83.3
A^2R	77	4.59	83.8

Figure 10 breaks down adjacent-step routing similarity by layer. Intermediate pairs, 0–1, 1–2, and 2–3, exhibit higher similarity than the final pair, 3–4. The pattern is present across layers and motivates reusing intermediate routing while refreshing it at the final evaluation.

D.4 Variation in retained-block counts

The cumulative-mass rule adapts the retained count to each head and query block. Concentrated relevance distributions reach the threshold with fewer blocks, whereas diffuse distributions require more blocks. The attention maps and selected patterns in Figure 12 illustrate this variation for selected heads and rollout chunks.

D.5 Static and adaptive allocation

Table 9 compares A^2R with static per-head and per-(head,layer) controls at the same displayed average sparsity. The static controls fix the allocation at their respective granularity, while A^2R uses query-dependent cumulative-mass selection. Average sparsity alone does not specify the allocation of retained blocks across queries.

D.6 Anchor support and the sparsity budget

Dense initialization determines how the first chunk is generated. Persistent KV retention instead constrains the support available to later queries. The latter imposes a lower bound on the attention budget, independent of the relevance ranking.

Consider seven equal-length chunks with a growing cache and full attention within each chunk. In units of one chunk-pair attention matrix, the total permitted work is $1 + \cdots + 7 = 28$. A dense first chunk together with mandatory retention of its KV for every later chunk requires at least $7/28 = 25\%$ of this work. The maximum global sparsity is therefore 75%. Also retaining the immediately preceding chunk raises the minimum to $12/28$, reducing the maximum global sparsity to approximately 57.14%. These bounds describe the stated support model; local windows and block padding change the accounting. They explain why anchor policies must be compared at a feasible common budget.

Table 10: Routing reuse schedules on Self-Forcing. Routing-call counts are per sparse chunk and attention layer, over five attention steps.

Routing steps	Calls	Latency (s)	Total
$\{0, 1, 2, 3, 4\}$	5	4.83	83.9
$\{0\}$	1	4.51	83.0
$\{0, 4\}$	2	4.59	83.8

D.7 Routing reuse schedules

Table 10 compares the routing schedules used in the Self-Forcing ablation. The default constructs routing at steps 0 and 4. Relative to recomputing at every evaluation, it reduces latency from 4.83 to 4.59s. Reusing the initial pattern through evaluation 4 further reduces latency to 4.51s, with an aggregate score of 83.0. Attention outputs are recomputed for all routing-reuse configurations.

D.8 Transfer across backbones

Table 11 summarizes results on the two autoregressive backbones and two additional video diffusion backbones. The aggregate-score differences between dense attention and A^2R are 0.1–0.2 points at these operating points. Attention-only acceleration is larger than pipeline acceleration because the remaining model computation is also included in pipeline latency.

Table 11: Cross-backbone comparison. Dense and sparse scores use the corresponding backbone; speedups are relative to its dense baseline.

Backbone	Dense Total	A^2R Total	Sparsity (%)	Attention speedup	Pipeline speedup
Self-Forcing 1.3B	84.0	83.8	77	$2.1\times$	$1.33\times$
LongLive 1.3B	83.1	83.0	72	$2.0\times$	$1.30\times$
CogVideoX-2B	80.9	80.8	73	$1.8\times$	$1.25\times$
LTX-Video-2B	80.0	79.9	70	$1.6\times$	$1.27\times$

E Additional Attention Visualizations

We further provide qualitative visualizations of attention maps across rollout chunks. As shown in Figure 11, several representative $(\mathrm{layer}, \mathrm{head})$ pairs exhibit broadly similar attention structures across chunks, although local variations remain. This suggests that some heads maintain relatively stable attention preferences throughout autoregressive rollout, complementing our quantitative analysis of routing stability across denoising steps.

We also visualize the correspondence between dense attention maps and the sparse patterns extracted by A^2R. As shown in Figure 12, the extracted sparse patterns align well with the dominant high-response regions in the original attention maps across rollout chunks. These examples illustrate the regions retained by block-level selection. Chunk 0 is omitted because it uses dense attention.

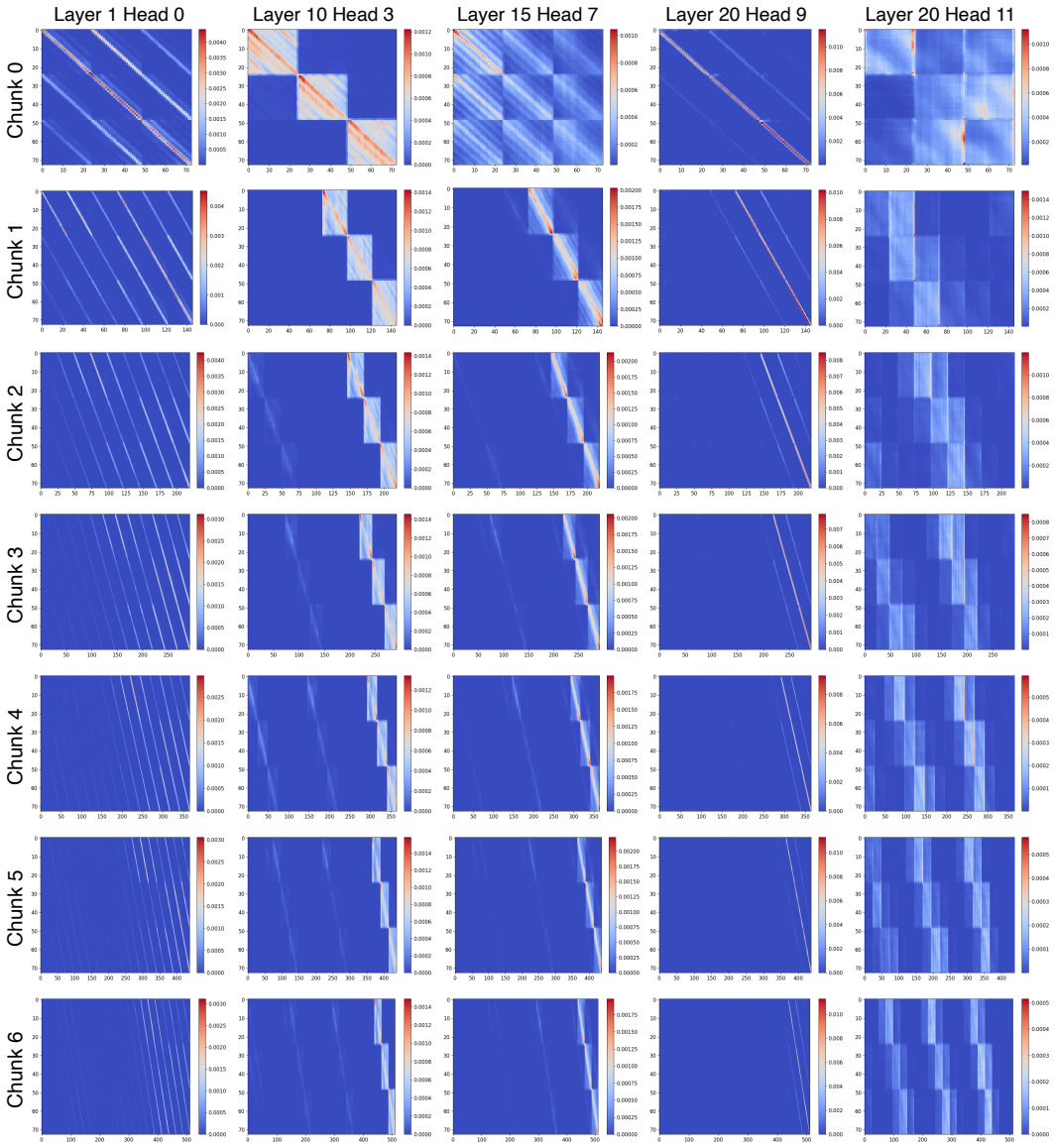

Figure 11: Chunk-wise attention consistency for selected $(\text{layer}, \text{head})$ pairs. Each column corresponds to a fixed $(\text{layer}, \text{head})$ pair visualized across different chunks. The similar attention structures observed across chunks indicate that certain heads exhibit stable, head-specific attention behavior over long-horizon autoregressive generation.

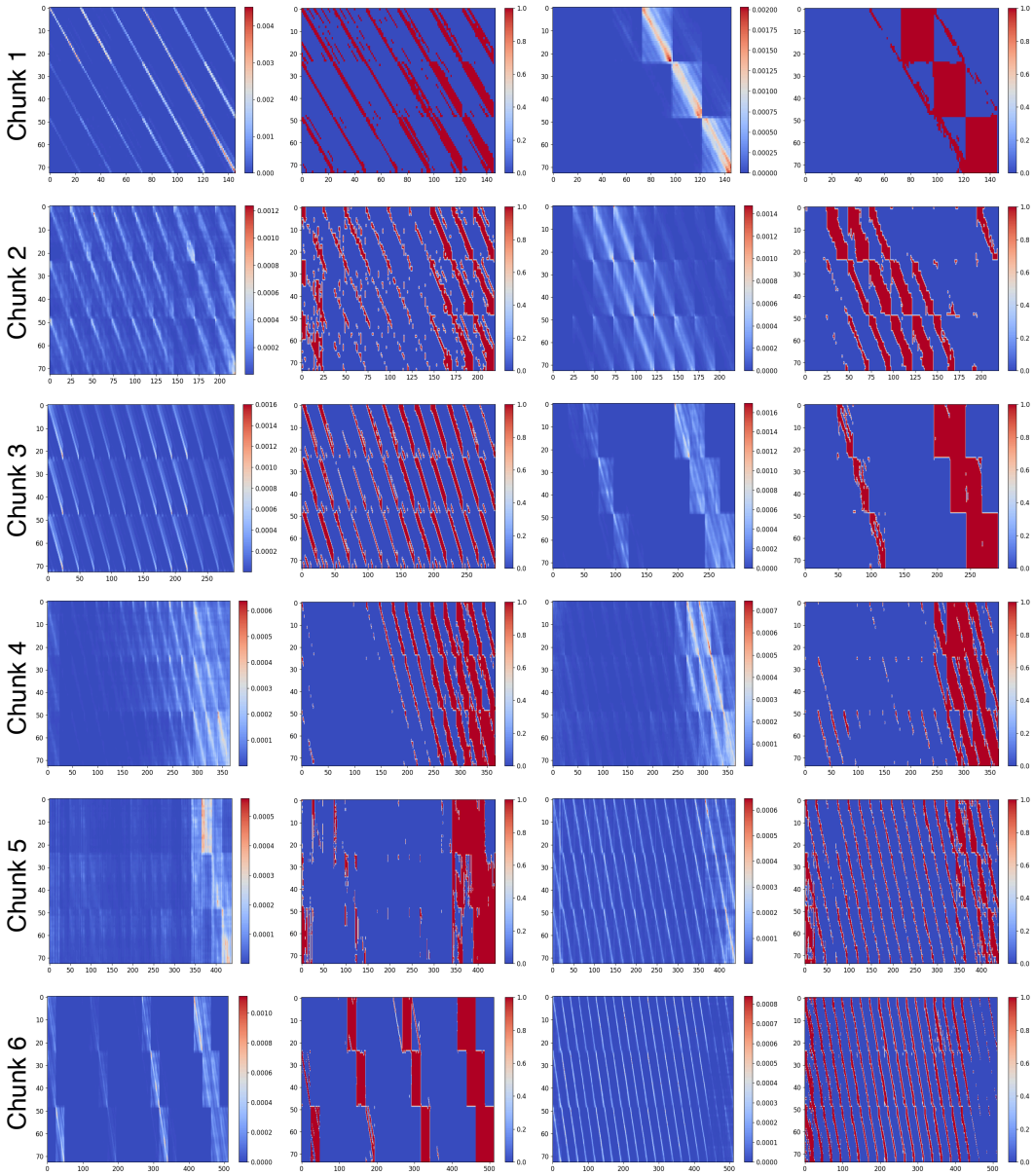

Figure 12: Attention maps and the corresponding sparse patterns across rollout chunks. The sparse patterns align well with the dominant attention regions in the original maps, indicating that A^2R preserves the main attention structure while discarding less relevant context.

F Additional Qualitative Results

Figures 13–15 compare generated videos for prompts involving complex backgrounds, stylized rendering, and artistic appearance. The examples illustrate subject fidelity, background coherence, and style consistency under dense and sparse inference.

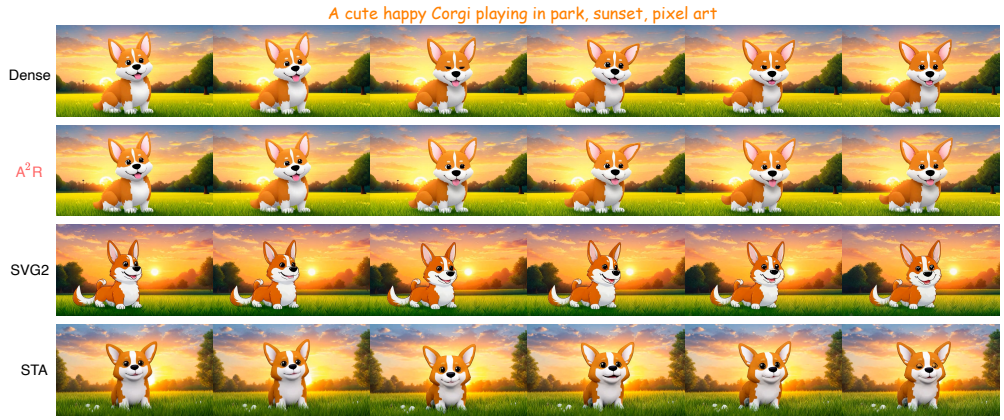

Figure 13: Qualitative comparison with different sparse methods on Self-Forcing.

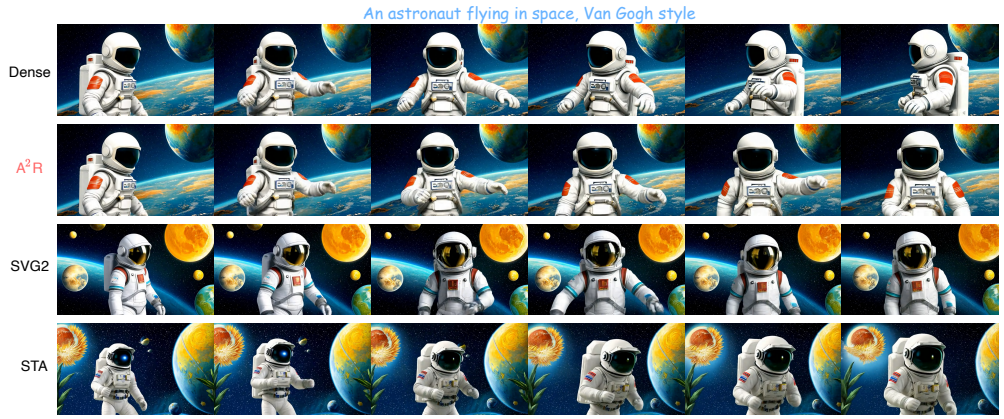

Figure 14: Qualitative comparison with different sparse methods on Self-Forcing.



Figure 15: Qualitative comparison with different sparse methods on Self-Forcing.

Table 12: Sparse kernel implementation variants. Ratios are computed from the displayed kernel times.

Implementation	Time (ms)	Speedup vs. dense
Dense	35.8	1.00×
PyTorch masked reference	29.8	1.20×
Sparse baseline	22.5	1.59×
With KV packing	19.2	1.86×
With scheduling	17.0	2.11×

G Sparse Attention Kernel Details

A^{2}R executes exact sparse attention over retained KV blocks only. The key idea is to separate routing from execution: block-level routing first identifies a compact set of key blocks to retain and the Triton kernel then performs exact attention over the gathered unpooled tokens from those retained blocks.

The sparse pattern is stored in a padded lookup table $\mathrm{LUT} \in \mathbb{Z}^{H \times Q_{\mathrm{blk}} \times K_{\mathrm{blk}}}$, where each entry stores the index of a retained key block or a padding sentinel for unused slots. During execution, each kernel instance processes one query block for one head and iterates only over the retained key blocks specified in the corresponding lookup-table row.

For numerical stability, the kernel maintains online softmax accumulation while streaming over retained key blocks. Algorithm 1 uses zero-based head, query-block, key-block and LUT-slot indices. Query-block index i is distinct from the running maximum $\mathbf{m}_{\max}$. Blocks are clipped at the true sequence lengths, and backbone-inadmissible attention edges are masked. Each valid query must retain at least one admissible key; duplicate key-block entries must be removed before execution. The computation is mathematically exact on the retained support, subject to floating-point rounding. Each kernel instance processes one (h, i) pair; flattening batch and head dimensions does not imply a separate launch per head.

G.1 Routing overhead and kernel execution

Let $\mathcal{C}$ be the attention steps in a given scope and $\mathcal{R} \subseteq \mathcal{C}$ the steps that construct routing. The average routing cost per attention evaluation is

$$\bar{t}_{\mathrm{route}} = \frac{\sum_{j \in \mathcal{R}} t_{\mathrm{route},j}}{|\mathcal{C}|}. \tag{18}$$

With two equally costly constructions in five steps, the average is $2t_{\mathrm{route}}/5$. For seven chunks with a dense first chunk, the default schedule uses $6 \times 2 = 12$ routing constructions per layer, compared with $6 \times 5 = 30$ without reuse. Averaging the same work over all 35 steps, including the dense first chunk, gives $12t_{\mathrm{route}}/35$.

Table 12 examines sparse execution. The final implementation takes 17.0 ms, compared with 35.8 ms for the dense kernel and 29.8 ms for a masked PyTorch reference. These correspond to approximately $2.11\times$ and $1.75\times$ kernel speedups, respectively. Kernel-level speedup is distinct from the speedup of the complete generation pipeline.

Under an additive runtime model, an attention speedup r and a dense attention fraction f yield

$$S_{\mathrm{pipeline}} = \frac{1}{(1-f) + f/r}. \tag{19}$$

This relation accounts for the smaller pipeline gain when non-attention computation remains unchanged. Routing and any preparation cost belong to the attention component when applying this model to a complete attention module.

G.2 Composition with low-precision inference

Composition with FP8 inference. Table 13 evaluates the combination of FP8 inference and A^{2}R on Self-Forcing 1.3B using five-step and three-step schedules. GEMMs use FP8, while attention

Algorithm 1 Exact sparse attention execution in $\mathrm{A}^2\mathrm{R}$

Require: Query $\mathbf{Q} \in \mathbb{R}^{H \times L_Q \times d}$, key $\mathbf{K} \in \mathbb{R}^{H \times L_K \times d}$, value $\mathbf{V} \in \mathbb{R}^{H \times L_K \times d}$
Require: Lookup table $\mathrm{LUT} \in \mathbb{Z}^{H \times Q_{\mathrm{blk}} \times K_{\mathrm{blk}}}$
Require: Query/key block sizes B_Q, B_K
Require: Backbone-permitted attention edges $\mathcal{E}$; deduplicated valid LUT indices or padding; nonempty permitted support per valid query
Ensure: Output $\mathbf{O} \in \mathbb{R}^{H \times L_Q \times d}$

for each head $h \in \{0, \dots, H-1\}$ and query block $i \in \{0, \dots, Q_{\mathrm{blk}} - 1\}$ **in parallel do**
  $q_0 \leftarrow iB_Q$, $q_1 \leftarrow \min((i+1)B_Q, L_Q)$
  Load query tokens $\mathbf{Q}^{\mathrm{blk}} \leftarrow \mathbf{Q}[h,\, q_0 : q_1, :]$
  Initialize running maximum $\mathbf{m}_{\max} \leftarrow -\infty$
  Initialize running normalizer $\mathbf{l} \leftarrow \mathbf{0}$
  Initialize accumulator $\mathbf{A} \leftarrow \mathbf{0}$
  for each slot $r \in \{0, \dots, K_{\mathrm{blk}} - 1\}$ **do**
    $j \leftarrow \mathrm{LUT}[h, i, r]$
    if j is padding **then**
      continue
    end if
    $k_0 \leftarrow jB_K$, $k_1 \leftarrow \min((j+1)B_K, L_K)$
    Load retained key block $\mathbf{K}^{\mathrm{blk}} \leftarrow \mathbf{K}[h,\, k_0 : k_1, :]$
    Load retained value block $\mathbf{V}^{\mathrm{blk}} \leftarrow \mathbf{V}[h,\, k_0 : k_1, :]$
    Compute block logits

$$\mathbf{S} \leftarrow \frac{\mathbf{Q}^{\mathrm{blk}}(\mathbf{K}^{\mathrm{blk}})^\top}{\sqrt{d}}$$

    Set logits for edges outside $\mathcal{E}$ to $-\infty$
    Apply the following updates only to rows with a permitted key in this tile; leave other rows unchanged
    Update online softmax statistics
    $\mathbf{m}_{\mathrm{new}} \leftarrow \max(\mathbf{m}_{\max},\, \max_{\mathrm{col}}(\mathbf{S}))$
    $\boldsymbol{\alpha} \leftarrow \exp(\mathbf{m}_{\max} - \mathbf{m}_{\mathrm{new}})$
    $\mathbf{P} \leftarrow \exp(\mathbf{S} - \mathbf{m}_{\mathrm{new}})$
    $\mathbf{l} \leftarrow \boldsymbol{\alpha} \odot \mathbf{l} + \sum_{\mathrm{col}} \mathbf{P}$
    $\mathbf{A} \leftarrow \boldsymbol{\alpha} \odot \mathbf{A} + \mathbf{P}\mathbf{V}^{\mathrm{blk}}$
    $\mathbf{m}_{\max} \leftarrow \mathbf{m}_{\mathrm{new}}$
  end for
  Normalize output block:

$$\mathbf{O}[h,\, q_0 : q_1, :] \leftarrow \mathbf{A} \oslash \mathbf{l}$$

end for
return O

computation and the KV cache remain in BF16. We report the overall speedup relative to the BF16 five-step dense baseline (6.11s), S_{base}, and the incremental speedup over FP8 dense inference with the same number of steps, S_{marginal}.

With five steps, $\mathrm{A}^2\mathrm{R}$ reduces latency from 4.54 to 3.02s, providing a $1.50\times$ incremental speedup with a 0.2-point decrease in VBench Total (83.9 to 83.7). With three steps, latency decreases from 2.95 to 2.31s, yielding a $1.28\times$ incremental speedup with a 0.1-point decrease in VBench Total (83.4 to 83.3). The combined FP8 and $\mathrm{A}^2\mathrm{R}$ configurations achieve overall speedups of $2.02\times$ and $2.65\times$ for five and three steps, respectively, relative to the common BF16 five-step dense baseline.

H Failure Cases

Figures 16 and 17 illustrate two cases where $\mathrm{A}^2\mathrm{R}$ degrades relative to dense attention. Their average intermediate-step routing similarities are 0.82 and 0.83, below the dataset mean of 0.89 (Table 14).

Table 13: FP8 inference results on Self-Forcing 1.3B. S_{base} is relative to the BF16 five-step dense baseline; S_{marginal} compares FP8 dense inference with FP8 A^2R at the same number of steps.

Attention	Steps	Latency (s)↓	Total↑	S_{base}	S_{marginal}
Dense	5	4.54	83.9	1.35×	–
A^2R	5	3.02	83.7	2.02×	1.50×
Dense	3	2.95	83.4	2.07×	–
A^2R	3	2.31	83.3	2.65×	1.28×

This association suggests examining routing stability when diagnosing difficult examples. The examples alone do not establish that routing reuse caused the degradation.



Figure 16: Qualitative failure case of A^2R relative to dense attention.



Figure 17: Qualitative failure case of A^2R relative to dense attention.

Table 14: Average intermediate-step routing similarity for two failure cases and the dataset mean.

Sample	Avg. Jaccard Similarity↑	Dataset Mean↑
Van Gogh's selfie, Van Gogh style	0.82	0.89
Gwen Stacy reading a book, pan right	0.83	0.89

I Broader Impacts

This paper studies efficient inference for autoregressive video diffusion models. By reducing attention cost in long-horizon generation, A^2R lowers inference latency and compute overhead, improving accessibility for research and creative applications while reducing per-sample resource usage. At the same time, more efficient video generation may lower the barrier to producing synthetic video content at scale, increasing risks such as misinformation, deceptive media, or unauthorized imitation. Although our method does not introduce new generation capabilities or new training data, it can amplify existing misuse risks by accelerating inference. We therefore recommend deployment together with standard safeguards, such as access control, provenance or watermarking when applicable and policy enforcement against harmful or deceptive use.