MonarchRT: Efficient Attention for Real-Time Video Generation

Krish Agarwal[1*] **Zhuoming Chen**[1] **Cheng Luo** **Yongqi Chen**[2]
Haizhong Zheng[1] **Xun Huang**[2] **Atri Rudra**[3] **Beidi Chen**[1]
[1]Carnegie Mellon University [2]Morpheus AI [3]University at Buffalo

Abstract

Video generation with Diffusion Transformers is bottlenecked by the quadratic cost of 3D self-attention, especially in real-time settings that are both few-step and autoregressive, where errors compound across time and each denoising step carries substantially more information. In this regime, prior sparse-attention approximations break down despite showing strong results for bidirectional, many-step diffusion. Specifically, we observe that video attention is not reliably sparse and instead combines periodic spatiotemporal structure with dynamic, sparse semantic interactions, exceeding the representational capacity of even oracle top-k attention. Building on this insight, we introduce **MONARCHRT**, a structured attention parameterization for video diffusion models that factorizes attention using Monarch matrices. Through appropriately aligned block structure and an extended *tiled Monarch parameterization*, we achieve high expressivity while preserving computational efficiency. We further overcome the overhead of parameterization through finetuning and optimized CUDA/Triton kernels. When applied to autoregressive Self-Forcing, MONARCHRT achieves approximately *95% effective attention sparsity* with no loss in quality, making MONARCHRT a pioneering work on high-quality sparse attention parameterization for real-time video generation. Our optimized implementation outperforms FlashAttention-2, FlashAttention-3, and FlashAttention-4 kernels on RTX 5090, H100, and B200 GPUs, respectively, providing kernel speedups in the range of 1.79–$5.28\times$ in the autoregressive setting and up to $13.6\times$ in the bidirectional setting. This enables us to achieve true real-time video generation with Self-Forcing at 16 FPS on a single RTX 5090.

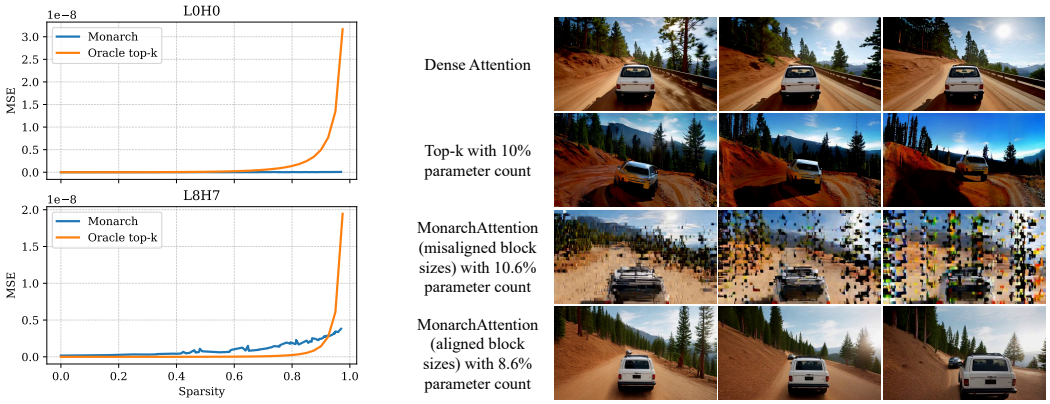
1 Introduction

With the common understanding that substantial redundancy exists in 3D attention, many approximation algorithms have been proposed to reduce its computational cost. However, we find that two key properties critical to real-time video generation [37], namely **auto-regressiveness** [13, 25, 32] and the use of **fewer diffusion steps** [31, 30, 16, 17], can significantly amplify approximation difficulties. Auto-regressive (AR) generation accumulates errors over time, while shortening the diffusion process compresses computation, causing each diffusion step to process substantially more information. Critically, these features are key in the design of real-time, *interactive* video generation models, including Genie 3 [1], WorldPlay [23], and LingBot-World [24].

Prior efficient approximations for 3D attention are fundamentally constrained by their **limited expressiveness**, which leads to substantial degradation in generated video quality. *Sparse attention* methods capture **positional patterns** by restricting attention to temporally and spatially local neighborhoods [15, 27, 39], or capture **semantic patterns** by attending to a small subset of tokens selected via

*All correspondence to krisha2@andrew.cmu.edu. Code is available at https://github.com/Infini-AI-Lab/MonarchRT.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) (b)

Figure 1: **Left:** MSE of oracle top-k and Monarch parameterizations of an attention map compared to the original dense attention map for varying levels of sparsity. Results shown for two different layers/heads on Self-Forcing. Monarch incurs much lower error for high levels of sparsity. **Right:** Example generations on the same prompt on Self-Forcing. Second row shows that exact top-k with 10% density still produces poor quality output. Fourth row shows that using inference-only MonarchAttention (with aligned block sizes) produces higher output quality with a lower parameter count of 8.6%. Third row shows that, although the parameter count is increased to 10.6%, inference-only MonarchAttention with misaligned block sizes incurs pixel-level permutation effects. Parameter counts are expressed as percentages of the entries in the full attention map.

clustering or retrieval-based strategies [38, 28, 34]. The former class weakens the inherent ability of DiTs to model long-range semantic dependencies, effectively regressing toward convolution-based architectures [4, 20], whose representational limitations are well known. The latter class, while more flexible, is highly parameter-inefficient when **dense mixing** is required, as evidenced by the failure of oracle top-k attention even with 10% of the FLOPs of full attention (Figure 1b). *Low-rank and linear attention* methods similarly struggle to represent long-range semantic patterns empirically [41, 12] and often require substantial retraining to adapt from DiTs [34]. As a result, existing approaches typically retain over 40% of full-attention FLOPs to preserve video quality [34, 28, 27]. While such compromises can be effective for bidirectional DiT models like Wan 2.1 [26], they are incompatible with state-of-the-art real-time video systems that operate autoregressively [13, 37].

Therefore, to preserve the strengths of DiTs while remaining viable for real-time video generation, an ideal attention approximation must capture three critical patterns simultaneously: **positional patterns**, **semantic patterns**, and **dense mixing**. This raises the question of whether an approach can be substantially more efficient than full attention while retaining expressive power.

Beyond sparse and low-rank methodologies, Monarch [7, 6, 21] introduces a unifying class of structured matrices that can efficiently represent a broad family of linear operators, including sparse and low-rank matrices as well as structured transforms such as FFT and Hadamard. More recently, MonarchAttention [29], which uses an iterative procedure to obtain Monarch factors, shows that attention matrices themselves can be effectively approximated using Monarch parameterizations. Our empirical analysis (Figure 1a) further reveals that Monarch recovers 3D attention matrices significantly better than oracle top-k or low-rank approximations, pointing to a promising direction for real-time video generation without sacrificing expressiveness.

However, leveraging Monarch parameterizations to approximate 3D attention in practical real-time video generation systems introduces several non-trivial technical challenges. First (**Shape Alignment**), unlike Monarch in MLPs [7, 10], where the semantics of input and output channels can be learned implicitly during training, the rows and columns of 3D attention matrices are explicitly tied to physical pixel patches in a video. Consequently, when the structural assumptions of Monarch parameterizations (e.g., block sizes) are misaligned with the spatial–temporal layout of pixel patches, the approximation quality can degrade dramatically, often leading to severe artifacts or even complete collapse of the generated videos. Second (**Limited Flexibility**), although Monarch allows adjusting computational cost by varying block sizes, we observe a systematic issue in which the approximation

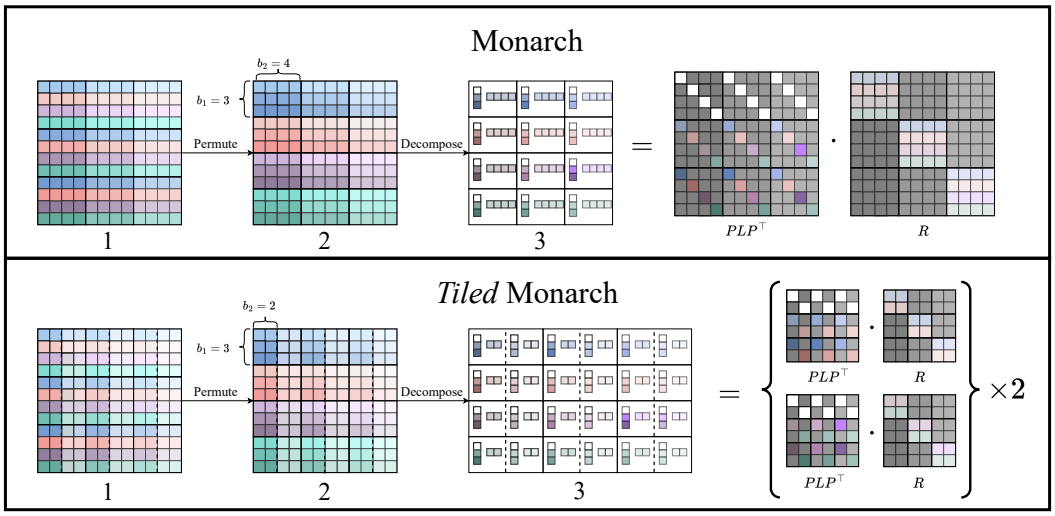


Figure 2: **Illustration of Regular and Tiled Monarch Parameterization.** Top: An example of Monarch parameterization applied to a 12×12 matrix with block size $(b_1, b_2) = (3, 4)$. Bottom: An example of *tiled* Monarch parameterization applied with block size $(b_1, b_2) = (3, 2)$. **(1)** The original matrix is first permuted to expose an implicit block-wise low-rank structure. **(2)** After permutation, the matrix is reorganized into blocks of size $b_1 \times b_2$, where each block corresponds to a group of rows and columns. **(3)** Each block is then independently decomposed into low-rank factors. Overall, Monarch represents the matrix as $PLP^\top R$, where P denotes a permutation matrix, $PLP^\top$ is a permuted block-diagonal matrix, and R is a block-diagonal matrix. In this example, the tiled Monarch parameterization has $\sim 1.43\times$ the parameter count and is strictly more expressive than the regular Monarch parameterization.

error does not reliably decrease as more FLOPs are allocated. This lack of monotonic refinement fundamentally limits Monarch's ability to progressively improve approximation quality, in sharp contrast to sparse-attention-based methods whose accuracy typically scales with increased computation. Third (**High Runtime Overhead**), the iterative refinement strategy adopted by MonarchAttention [29] requires multiple refinement steps to achieve high-quality approximations, rendering it computationally prohibitive for real-time video generation systems with strict latency constraints.

To address these challenges, we introduce MONARCHRT, a unified framework for applying Monarch parameterizations to video generation. MONARCHRT first provides a factorization structure that is explicitly compatible with 3D attention, and further enables *arbitrarily accurate* matrix approximations through our proposed *Tiled Monarch Parameterizations* (illustrated in Figure 2 Bottom). In addition, by finetuning MonarchAttention, we substantially reduce the number of iterative refinement steps required during inference. These components make Monarch parameterization both accurate and efficient for real-time video generation.

In summary, our contributions are as follows. We (1) analyze the structural patterns of 3D video attention, explaining why existing sparse- and low-rank-based methods fail to capture them and why Monarch provides a principled and expressive parameterization (Sections 3.1 to 3.3); (2) identify three practical challenges in applying Monarch to video models—*shape misalignment*, *limited flexibility*, and *parameterization overhead* (Section 3.4); (3) introduce *Tiled Monarch Parameterization* together with a simple yet crucial block-alignment strategy, which generalizes the original Monarch parameterization and enables arbitrarily accurate approximations of 3D attention (Sections 4.1 and 4.2); and (4) reduce inference-time cost via finetuning and an optimized CUDA/Triton kernel implementation supporting both forward and backward passes (Section 4.3).

In Section 5, we conduct comprehensive evaluations of MONARCHRT across multiple video generation configurations. We center our results on autoregressive Self-Forcing and demonstrate that MONARCHRT jointly achieves the highest visual quality and lowest latency among existing sparse-attention baselines. Namely, while prior sparse attention methods fail to maintain visual fidelity beyond 85% sparsity, MONARCHRT preserves high generation quality even at 95% effective sparsity. Through our optimized kernel implementation, this enables attention speedups of 1.79–$5.28\times$ and

end-to-end speedups of 1.17–$1.42\times$ over dense FlashAttention backends [8, 5, 22] on RTX 5090, H100, and B200 GPUs. Notably, this translates to true real-time 16 FPS generation with Self-Forcing on a single RTX 5090. We extend our evaluations in Section D.1 to 50-step and 4-step bidirectional Wan2.1-1.3B [26], where MONARCHRT continues to achieve strong quality and efficiency, demonstrating the generality of our approach.

2 Background

2.1 Monarch

Monarch Parameterization. Monarch parameterization [7] represents an $N \times N$ matrix as $\boldsymbol{M} = \boldsymbol{P}\boldsymbol{L}\boldsymbol{P}^\top\boldsymbol{R}$, with $N = b_1 b_2$, where $\boldsymbol{P}$ is a fixed permutation that reshapes a length-N vector into a $b_1 \times b_2$ matrix, transposes, and flattens it back. This permutation exposes a block-wise structure (illustrated in Figure 2): $\boldsymbol{L}$ is block-diagonal with b_2 blocks of size $b_1 \times b_1$, while $\boldsymbol{R}$ is block-diagonal with b_1 blocks of size $b_2 \times b_2$. Equivalently, Monarch can be viewed as a block-wise rank-1 structure under permutation: as a 4D tensor of shape $b_1 \times b_2 \times b_1 \times b_2$, the entries are $\boldsymbol{M}_{\ell jki} = \boldsymbol{L}_{j\ell k}\,\boldsymbol{R}_{kji}$. The class of Monarch matrices strictly generalizes butterfly matrices, and since sparse matrices can be represented as products of butterfly matrices [6], the same representational guarantees extend to Monarch.

MonarchAttention [29] approximates the attention matrix $\boldsymbol{A} = \mathrm{softmax}(\boldsymbol{Q}\boldsymbol{K}^\top) \in \mathbb{R}^{N\times N}$ using Monarch factors $\boldsymbol{L} \in \mathbb{R}^{b_2\times b_1\times b_1}$ and $\boldsymbol{R} \in \mathbb{R}^{b_1\times b_2\times b_2}$ with $N = b_1 b_2$. A direct projection via per-block SVD would require materializing $\boldsymbol{A}$, which is prohibitive. Instead, MonarchAttention exploits a variational formulation of softmax [2] to directly optimize the Monarch factors via alternating maximization, then applies $\boldsymbol{L}$ and $\boldsymbol{R}$ to $\boldsymbol{V}$ to obtain the attention output without ever constructing $\boldsymbol{A}$. Algorithmic details are provided in Section A.

2.2 Problem Formulation

Given query, key, and value matrices $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{n\times d}$, the attention output is $\mathrm{Attn}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \mathrm{softmax}(\boldsymbol{Q}\boldsymbol{K}^\top)\boldsymbol{V}$, where the softmax is applied row-wise. We denote the attention matrix by $\boldsymbol{A} = \mathrm{softmax}(\boldsymbol{Q}\boldsymbol{K}^\top) \in \mathbb{R}^{n\times n}$. The goal of attention parameterization is to reduce the quadratic cost of computing $\boldsymbol{A}$ while preserving approximation quality. Formally, given an approximation procedure $f(\boldsymbol{Q}, \boldsymbol{K})$, we seek to balance the approximation error $\mathbb{E}[\|f(\boldsymbol{Q}, \boldsymbol{K}) - \boldsymbol{A}\|_F^2]$ and the computational cost $\mathcal{C}(f(\cdot))$.

We consider three representative parameterization families, each imposing a distinct structural constraint on $\boldsymbol{A}$:

1. **Sparse.** $\boldsymbol{A}_\mathrm{s} = \mathrm{argmin}_{\boldsymbol{A}'\in\mathbb{R}^{n\times n}\,:\,\mathrm{NNZ}(\boldsymbol{A}')\le k} \|\boldsymbol{A}' - \boldsymbol{A}\|_F^2$, for a budget of k nonzero entries.
2. **Low-rank.** $\boldsymbol{A}_\ell = \mathrm{argmin}_{\bar{\boldsymbol{Q}},\,\bar{\boldsymbol{K}}\in\mathbb{R}^{n\times\bar{n}}} \|\bar{\boldsymbol{Q}}\bar{\boldsymbol{K}}^\top - \boldsymbol{A}\|_F^2$, with $\bar{n} < n$.
3. **Monarch.** $\boldsymbol{A}_\mathrm{m} = \mathrm{argmin}_{\boldsymbol{A}'\in\mathcal{M}} \|\boldsymbol{A}' - \boldsymbol{A}\|_F^2$, where $\mathcal{M}$ denotes the family of Monarch matrices.

An ideal parameterization achieves low approximation error with minimal computational cost, which is ultimately governed by the structural properties of $\boldsymbol{A}$.

3 Observations and Analysis

In this section, we present our core argument that, in principle, Monarch parameterization provides a fundamentally stronger approximation than sparse parameterization for modeling 3D attention. We further identify three key challenges that arise in practice, each of which limits its flexibility and achievable accuracy.

3.1 Approximation Error Analysis

Failure of oracle top-k attention. In Figures 1a and 1b, we show that oracle top-k attention incurs large errors when approximating the attention map for high sparsity levels, leading to end-to-end



(a) (b)

Figure 3: **Left:** Percent of key-value tokens that fall under the top-p threshold per head in Self-Forcing, averaged across all query tokens for several decoding iterations, on the first denoising iteration. Results shown for 5 randomly sampled heads/layers. **Right:** Example generations on the same prompt on Self-Forcing using MONARCHRT with 10 and 1 iterations of iterative refinement. Using 10 iterations yields much higher quality but is practically inefficient, so we improve the one-iteration result through training.

quality degradation (the example car front exhibits severe geometric distortion), even with a 10% computation budget (which is reasonably high for an oracle approximation). We attribute this to the 3D attention map being less sparse than expected. As illustrated in Figure 3a, for certain attention heads, 48–84% of tokens are required to recover 95% of the attention mass. For non-oracle sparse parameterizations, such as position- or block-top-k, the situation worsens, as such methods are not guaranteed to retrieve the highest-scoring keys.

3.2 Rethinking the Structure of 3D Attention Maps

The failure of sparse parameterization stems from the incorrect assumption that the attention map is inherently sparse. 3D attention does not simply exhibit sparsity; instead, it reveals pronounced *periodic structure* driven by spatiotemporal position, indicating that dense and repeating interactions are fundamental rather than exceptional. Inspired by Li et al. [15], we informally model the attention map as

$$\boldsymbol{A}_{(f_0,h_0,w_0),(f_1,h_1,w_1)} = \boldsymbol{D}_{(f_0,h_0,w_0),(f_1,h_1,w_1)} + \boldsymbol{S}_{(f_0,h_0,w_0),(f_1,h_1,w_1)} + \epsilon, \tag{1}$$
$$\boldsymbol{D}_{(f_0,h_0,w_0),(f_1,h_1,w_1)} = d_w(w_0,w_1)\; d_h(h_0,h_1)\; d_t(f_0,f_1).$$

Here, $\boldsymbol{D}$ captures the *positional* component of attention, modeled as the separable product of three distance functions along the spatial width (d_w), spatial height (d_h), and temporal (d_t) dimensions. Empirical observations suggest that attention scores induced by positional structure decay smoothly as tokens become farther apart. However, our analysis does not require a specific functional family for the distance functions $d(\cdot,\cdot) \le 1$. $\boldsymbol{S}$ represents the *semantic* component, which models long-range relationships independent of positional proximity; most entries of $\boldsymbol{S}$ are 0 due to the sparsity of meaningful semantic correspondence, the others are 1. ϵ denotes residual noise or modeling error. For simplicity, we omit the normalization here. This modeling immediately leads to Theorem 3.1.

Theorem 3.1. *(informal) The 3D attention matrix $\boldsymbol{A} \in \mathbb{R}^{fhw \times fhw}$ defined above admits a structural decomposition $\boldsymbol{A} = \boldsymbol{P}\boldsymbol{D}' + \boldsymbol{S} + \epsilon$, where $\boldsymbol{P}$ is a permutation matrix, $\boldsymbol{D}'$ is* blockwise rank-1 *with block sizes (b_1, b_2) satisfying $b_1 b_2 = fhw$, and $\boldsymbol{S}$ is a sparse matrix.*

Naively combining low-rank and sparse parameterization [34, 3, 9] is not sufficient, because a matrix that is low-rank within each local block can still become full-rank when viewed globally. We formally present Theorem 3.1 in Section B.1.

3.3 Monarch Well-Represents 3D Attention Maps

The decomposition $\boldsymbol{A} = \boldsymbol{P}\boldsymbol{D}' + \boldsymbol{S} + \epsilon$ motivates approximating the permuted attention map with rank-1 blocks. When strong semantic interactions are limited, an appropriate permutation and block sizes can reduce their overlap within a block and their mixing with positional interactions. Section B.3 analyzes representative cases in which blocks are rank-1, and Section B.4 shows how exact blockwise rank-1 structure admits a Monarch representation. Real attention need only approximately follow this structure.

Assessing blockwise rank-1 structure. For a nonzero attention block B, let $B_1 = \sigma_1 u_1 v_1^\top$ denote its best rank-1 approximation under Frobenius norm, given by the leading singular component of its

SVD. SVD orthogonality gives

$$e(B) = \frac{\sigma_1(B)^2}{\|B\|_F^2}, \qquad \frac{\|B - B_1\|_F^2}{\|B\|_F^2} = 1 - e(B).$$

Thus, a high leading-component energy fraction directly implies a small optimal relative squared reconstruction error. However, whether attention blocks exhibit this structure depends on how video tokens are partitioned. This raises the practical questions discussed next, namely how to choose suitable blocks, increase their expressive capacity, and estimate their factors efficiently. We empirically evaluate the resulting block structure in Section D.2.

3.4 Practical Challenges of MonarchAttention in Video Generation

Aside from the advantages, we identify three challenges that prevent direct usage of MonarchAttention in video generation models.

3.4.1 Challenge 1: Block misalignment with spatiotemporal structure.

The effectiveness of Monarch parameterization critically depends on whether its block structure aligns with the underlying spatiotemporal organization of video tokens. We illustrate this using a 3D attention example with $(f, h, w) = (2, 3, 3)$, yielding an 18×18 attention matrix, and compare two block configurations. With $(b_1, b_2) = (6, 3)$, tokens spatially adjacent within the same frame are grouped together, respecting the natural locality of the video, and as a result nearly all blocks exhibit an approximately rank-1 structure. In contrast, choosing $(9, 2)$ produces blocks of the same total size but groups tokens by flattened indices, mixing tokens that are distant in the original layout, so most blocks no longer admit a low-rank structure. This is visualized in Figure 11 (Section C). Instead, as shown in Figure 1b, aligned grouping preserves visual fidelity while misaligned grouping causes severe degradation.

3.4.2 Challenge 2: Lack of monotonic refinement with increased computation.

Even when block sizes are perfectly aligned, semantic interactions may still cause certain blocks to deviate from being rank-1, and such correspondences are sparse, irregular, and inherently unpredictable. A natural strategy is to increase the computational budget via smaller blocks that are less likely to mix multiple semantic interactions. However, the constraint $b_1 b_2 = N$ means changing (b_1, b_2) redistributes parameters between $\boldsymbol{L}$ and $\boldsymbol{R}$ rather than guaranteeing more total computation. Specifically, it trades finer partitioning in one dimension for coarser partitioning in the other, which does not reliably maintain the representation of sparse semantic interactions, so additional compute does not reliably reduce approximation error. In contrast, traditional sparse attention methods achieve sharper monotonic accuracy–efficiency tradeoffs for approximating a fixed attention matrix since computing more entries indeed reduces the error of its zero-filled approximation.

3.4.3 Challenge 3: High overhead of iterative refinement.

MonarchAttention estimates the Monarch factors $\boldsymbol{L}$ and $\boldsymbol{R}$ through an iterative refinement procedure. While increasing the number of refinement steps generally improves approximation accuracy, the runtime scales linearly with the number of iterations, directly reducing throughput.

4 MonarchRT

In this section, we propose MONARCHRT by addressing the three challenges identified above, i.e., how to align block sizes effectively (Section 4.1), how to enable finer-grained Monarch parameterization beyond fixed block areas (Section 4.2), and how to reduce the parameterization cost for real-time usage (Section 4.3). We then describe our custom implementation that supports efficient training of MonarchAttention on long sequences.

4.1 Aligning Monarch Blocks

Key insight: A Monarch parameterization is *aligned* with video attention if each spatiotemporal dimension (f, h, w) is entirely contained within exactly one block dimension. Only under this

condition can Monarch exactly represent fully separable positional attention patterns; if a dimension partially spans both b_1 and b_2, the attention cannot factor into fully separable terms $\boldsymbol{L}$ and $\boldsymbol{R}$, and the approximation cannot exactly represent the assumed attention structure even with increased parameter count. This explains the behavior in Figure 1b, where aligned block sizes (e.g., (fh, w)) preserve visual fidelity while misaligned choices (e.g., $(f \cdot \frac{h}{4}, 4w)$) lead to severe degradation despite higher nominal capacity. Excluding the degenerate dense cases $(fhw, 1)$ and $(1, fhw)$, this rule yields exactly six aligned block configurations: (fh, w), (w, fh), (f, hw), (hw, f), (fw, h), (h, fw). We provide a formal construction showing that any aligned configuration exactly reproduces the separable positional model in Section B.6.

4.2 Tiled Monarch Parameterization

We begin by recalling the standard Monarch parameterization. Given block sizes (b_1, b_2) with $N = b_1 b_2$, a Monarch matrix $\boldsymbol{M} \in \mathbb{R}^{N \times N}$ can be written as $\boldsymbol{M}_{mn} = \boldsymbol{M}_{(\ell b_2 + j)(k b_2 + i)} = \boldsymbol{L}_{j\ell k} \boldsymbol{R}_{kji}$, where $\boldsymbol{L} \in \mathbb{R}^{b_2 \times b_1 \times b_1}$ and $\boldsymbol{R} \in \mathbb{R}^{b_1 \times b_2 \times b_2}$.

Algorithm. We propose *Tiled Monarch Parameterization*, which generalizes Monarch by splitting each permuted $b_1 \times b_2$ block into $c_1 c_2$ sub-blocks of size $\frac{b_1}{c_1} \times \frac{b_2}{c_2}$, where $c_1 \mid b_1$ and $c_2 \mid b_2$. Collecting the same sub-block position from every original block, while preserving the original block grid, gives $c_1 c_2$ matrices of size $\frac{N}{c_1} \times \frac{N}{c_2}$. Dividing each collected matrix into a $c_2 \times c_1$ grid yields $(c_1 c_2)^2$ square pieces in total, each corresponding to a smaller Monarch matrix after row permutation. We call these *Monarch tiles*. Each Monarch tile is an $\frac{N}{c_1 c_2} \times \frac{N}{c_1 c_2}$ matrix with block sizes $(\frac{b_1}{c_1}, \frac{b_2}{c_2})$ and contains $\frac{N}{c_1 c_2}$ rank-1 sub-blocks after permutation.

Formally, we parameterize $\boldsymbol{M}$ using tiled factors $\boldsymbol{L}'$ and $\boldsymbol{R}'$, where $\boldsymbol{L}'$ has shape $(c_1, c_2, c_1, c_2, \frac{b_2}{c_2}, \frac{b_1}{c_1}, \frac{b_1}{c_1})$ and $\boldsymbol{R}'$ has shape $(c_1, c_2, c_1, c_2, \frac{b_1}{c_1}, \frac{b_2}{c_2}, \frac{b_2}{c_2})$.

To estimate tiled Monarch factors for the attention matrix $\boldsymbol{A}$, we extend MonarchAttention by applying the same alternating refinement procedure independently to each tile. To enforce row-stochasticity, we impose slightly stronger constraints on the tiled factors, detailed in Section A.2.

Tiled Monarch is strictly more expressive than regular Monarch for the same base block sizes (Theorem B.1, formally stated and proved in Section B.2): tiled factors increase parameter counts by factors of c_2 and c_1 respectively for $\boldsymbol{L}$ and $\boldsymbol{R}$, and by tying parameters across tiles one can exactly recover the original Monarch parameterization, so tiled Monarch is at least as expressive as standard Monarch and strictly more so when $c_1 > 1$ or $c_2 > 1$.

Crucially, tiled Monarch enables *controllable refinement*. By increasing (c_1, c_2), each original block is subdivided into smaller sub-blocks, allowing the approximation to better capture sparse and irregular semantic interactions that cannot be modeled by a single rank-1 block.

Example. Recall that for video attention with resolution (f, h, w), we set the aligned Monarch block sizes to $(b_1, b_2) = (fh, w)$ as shown in Section 4.1. However, when attention exhibits neighborhood-level sparsity with neighborhood size (n_f, n_h, n_w), a single Monarch block may span multiple neighborhoods and therefore violate the rank-1 assumption.

Tiled Monarch resolves this issue by enforcing locality within each rank-1 sub-block. Specifically, we choose $c_1 = \frac{f}{n_f} \cdot \frac{h}{n_h}$ and $c_2 = \frac{w}{n_w}$, splitting each original block into $c_1 c_2$ sub-blocks of size $(n_f n_h) \times n_w$. Each sub-block therefore contains tokens from only a single spatiotemporal neighborhood along every dimension, making the rank-1 assumption locally valid.

The degenerate case $(n_f, n_h, n_w) = (1, 1, 1)$ reduces to dense attention. Empirically, we find that choosing n_f as a small constant while letting $n_h = \mathcal{O}(h)$ and $n_w = \mathcal{O}(w)$ achieves high visual quality with a sparse parameterization.

Effective sparsity and computational complexity. A permuted sub-block of size $p \times q$, where $p = \frac{b_1}{c_1}$ and $q = \frac{b_2}{c_2}$, is represented by a rank-1 outer product using $p + q$ scalars instead of pq dense entries. We therefore define effective sparsity as

$$s_{\text{eff}} = 1 - \frac{p + q}{pq} = 1 - \frac{c_1}{b_1} - \frac{c_2}{b_2}. \tag{2}$$

All permuted sub-blocks have the same size and use rank-1 factors, so their count cancels for the full matrix's effective sparsity. This measures parameter savings relative to the dense matrix, and the represented attention matrix can itself remain dense. Tiled Monarch preserves the efficient computation pattern of MonarchAttention. Each tile is processed independently using the same alternating refinement procedure, and the overall complexity scales linearly with the number of tiles.

4.3 Finetuning and Efficient Implementation

To further reduce the computational overhead of iterative refinement, we introduce *Monarch finetuning*, a lightweight training procedure that dramatically decreases the number of refinement steps required to obtain high-quality Monarch parameterizations. In practice, we find that with finetuning, even a single refinement iteration is often sufficient to match the visual fidelity of much more expensive multi-step optimization.

To enable end-to-end training of Monarch factors, we implement custom forward and backward kernels tailored for long-sequence 3D attention. We visualize the forward process in Figure 6 (Section A.3) and provide additional information on the MonarchAttention algorithm itself in Sections A.1 and A.2. As in the original MonarchAttention, the $\boldsymbol{\beta}$ terms—and hence $\boldsymbol{L}$ and $\boldsymbol{R}$—can be materialized directly in SRAM using FlashAttention-style online softmax.

5 Empirical Validation

In this section, we demonstrate that MONARCHRT can accelerate state-of-the-art real-time video generation models while preserving video fidelity. We center our evaluation on autoregressive Self-Forcing [13]:

- In Section 5.1, we demonstrate that MONARCHRT achieves quality on par with the original dense model while enabling substantial speedups, and it achieves strict quality–latency dominance over existing sparse attention baselines. Furthermore, our latency evaluations demonstrate that MONARCHRT enables true real-time 16 FPS generation with Self-Forcing on a single RTX 5090.
- We further conduct an ablation study to isolate the effects of each component of our method, i.e., alignment, tiling, and finetuning. We demonstrate that misaligned configurations lead to quality collapse, while tiling and finetuning can improve quality when applied to aligned configurations.

Additional 50-step and 4-step bidirectional Wan evaluations are provided in Section D.1. The relationship between sparsity, attention approximation error, and video quality is also analyzed in Section D.3.

5.1 Generation Quality and Efficiency

Setup. We evaluate the 1.3B Self-Forcing model, generating 81 frames at 480p with batch size one. Table 1 shows total VBench [14] scores alongside end-to-end (E2E) latency on RTX 5090, H100, and B200. We use one Monarch refinement iteration ($T = 1$) for all timed configurations.

Evaluation settings and baselines. All E2E latencies are measured using `torch.compile` with `max-autotune-no-cudagraphs`, and all measurements use BF16 inputs. For the original dense model, we use FlashAttention (FA) kernels, namely FA2 on RTX 5090, FA3 on H100, and FA4 on B200. The trained result uses MONARCHRT during Self-Forcing's DMD training stage [31, 30]. The same inference kernels and shapes are used regardless of whether MONARCHRT is applied with or without additional training, so the relevant rows share latency measurements.

For sparse baselines, SpargeAttention [36] natively supports rectangular attention map shapes for autoregressive generation, and we evaluate its 85%, 90%, and 95% sparsity settings. We additionally compare against oracle top-k, SVG [27], and RadialAttention [15]. SVG and RadialAttention are quality-only AR adaptations, as they only natively support square attention maps for bidirectional models; to adapt them, their masks are intersected with Self-Forcing's block-causal mask, so their nominal sparsity does not measure sparsity beyond causality. Similarly for oracle top-k, no latency numbers are reported since this serves as a theoretically optimal sparse attention baseline, and it is not efficiently implementable in practice. VSA [38] and SVG2 [28] are evaluated in their native bidirectional setting in Section D.1.

Table 1: Self-Forcing, 4 denoising steps, 81 frames at 480p. Quality-only baselines have no native AR latency since there is no existing efficient implementation to compare against.

Method	Sparsity	VBench (↑)	E2E latency (s) ↓		
			RTX 5090	H100	B200
Dense	0%	0.836	7.278	2.935	1.742
Training-free					
SpargeAttention	85%	0.765	5.279	2.935	2.160
SpargeAttention	90%	0.723	5.193	2.893	2.073
SpargeAttention	95%	0.648	5.126	2.849	2.002
Oracle top-k	85%	0.799	—	—	—
SVG	85%	0.615	—	—	—
RadialAttention	85%	0.816	—	—	—
MONARCHRT	85%	0.832	8.350	4.357	2.609
MONARCHRT	90%	**0.839**	5.483	2.670	1.705
MONARCHRT	95%	0.825	**5.110**	**2.389**	**1.489**
Trained					
MONARCHRT	95%	0.838	**5.110**	**2.389**	**1.489**

Quality–latency comparison. Table 1 shows that training-free MONARCHRT at 90% effective sparsity and trained MONARCHRT at 95% achieve quality comparable to the dense model. Both 95% variants exceed all sparse baselines in VBench and have lower latency than each native sparse baseline on all three GPUs, thus establishing MONARCHRT's quality-and-speed dominance over the sparse baselines. At 95% effective sparsity, MONARCHRT delivers 1.17–1.42× E2E speedups over dense, enabling **true real-time 16 FPS playback** on a single RTX 5090.

Table 2: Self-Forcing ablation study, reporting total VBench. The first row uses the original misaligned MonarchAttention. Subsequent rows isolate tiling, alignment, and integration into DMD training.

Aligned?	Tiled?	Training	Sparsity	VBench (↑)
✗	✗	—	97%	0.542
✗	✓	—	95%	0.425
✓	✗	—	97%	0.750
✓	✓	—	95%	0.825
✓	✓	—	90%	0.839
✓	✓	DMD	95%	0.838

Attention latency. Figure 4 tracks attention latency as the autoregressive KV cache grows. At the final decoding position, the 95% MONARCHRT configuration achieves 1.79–5.28× kernel speedups over dense attention across the three GPUs. The smaller E2E gains reflect the remaining costs of text encoding, other transformer operations, VAE decoding, and output transfer.

5.2 Ablation Study

Table 2 identifies alignment as the prerequisite for effective tiling. Namely, tiling alone fails to recover quality under misalignment, whereas aligning untiled blocks substantially improves it. With alignment in place, tiling provides further gains, and DMD training improves quality while preserving the sparsity and inference configuration.

6 Related Works

Efficient Attention. As transformers have widely grown in adoption across a variety of applications, significant work has gone to reducing quadratic softmax attention cost, both algorithmically and via implementation. FlashAttention [8, 22] is a fused attention kernel implementation that avoids materializing large attention score matrices in GPU memory and instead computes them on the fly in a tiled manner within on-chip SRAM. For video generation, both static [39, 27, 15] and dynamic [38, 28, 36] sparse attention are explored.

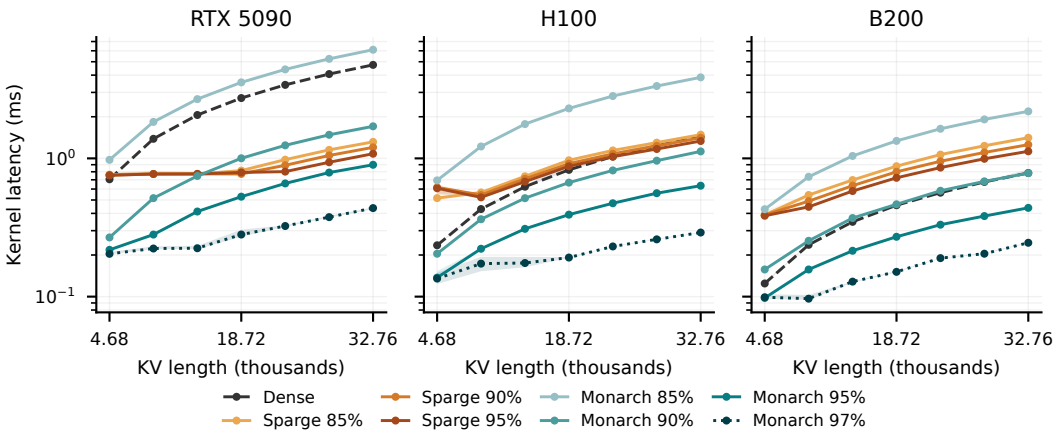


Figure 4: Self-Forcing attention latency at all seven generation positions. Query length is fixed at 4680 tokens, and KV length grows from 4680 to 32760 tokens, matching Self-Forcing's 480p autoregressive chunk sizes. Dense uses FA2 on RTX 5090, FA3 on H100, and FA4 on B200. The 97% Monarch curve is a theoretical setting evaluated more extensively in the bidirectional setting.

Efficient video generation. Light-Forcing [18] develops autoregressive sparse attention through chunk-aware sparsity allocation and hierarchical context retrieval, combining it with FP8 quantization and an efficient LightVAE decoder [11]. Other efficient video diffusion systems like TurboDiffusion [37] also employ low-bit weight and attention quantization [35, 33]. These system optimizations are complementary to structured attention parameterization, which is the focus of our work.

7 Limitations and Broader Impacts

MONARCHRT is explicitly designed for video generation DiTs, and generalization to non-video modalities and models with substantially different attention patterns remains untested. Furthermore, videos that involve fast motion or scene transitions may weaken positional interactions and make sparse semantic correlations more prevalent, potentially reducing the suitability of blockwise rank-1 approximations. Evaluating MONARCHRT in these settings remains future work.

By reducing video generation costs, MONARCHRT can broaden access to video generation capabilities on consumer hardware and support positive applications such as world-model simulation and robotics. However, we acknowledge that the same efficiency gains may also facilitate easier creation of deceptive synthetic videos for misinformation.

8 Conclusion

We presented MONARCHRT, a principled and efficient attention parameterization for real-time video generation. By analyzing the structure of 3D attention, we showed that its spatiotemporal periodicity and sparse semantic interactions align naturally with the expressive power of Monarch matrices. Our design combines aligned block structures, a proposed *tiled Monarch parameterization*, and finetuning with an optimized CUDA/Triton kernel implementation to deliver substantial speedups while preserving fidelity.

Acknowledgements

We are grateful for access to NVIDIA computing resources. This work was supported in part by Google Research Award, Google ML & System Junior Faculty Award, Amazon Research Award, Fireworks AI, Intel, Li Auto, Moffett AI, and CMU CyLab Seed funding. This material is also based upon work supported by the National Science Foundation (NSF) under Grant Nos. CCF-2504353 and CCF-2247014, and by IARPA. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

[1] Philip J. Ball, Jakob Bauer, Frank Belletti, Bethanie Brownfield, Ariel Ephrat, Shlomi Fruchter, Agrim Gupta, Kristian Holsheimer, Aleksander Holynski, Jiri Hron, Christos Kaplanis, Marjorie Limont, Matt McGill, Yanko Oliveira, Jack Parker-Holder, Frank Perbet, Guy Scully, Jeremy Shar, Stephen Spencer, Omer Tov, Ruben Villegas, Emma Wang, Jessica Yung, Cip Baetu, Jordi Berbel, David Bridson, Jake Bruce, Gavin Buttimore, Sarah Chakera, Bilva Chandra, Paul Collins, Alex Cullum, Bogdan Damoc, Vibha Dasagi, Maxime Gazeau, Charles Gbadamosi, Woohyun Han, Ed Hirst, Ashyana Kachra, Lucie Kerley, Kristian Kjems, Eva Knoepfel, Vika Koriakin, Jessica Lo, Cong Lu, Zeb Mehring, Alex Moufarek, Henna Nandwani, Valeria Oliveira, Fabio Pardo, Jane Park, Andrew Pierson, Ben Poole, Helen Ran, Tim Salimans, Manuel Sanchez, Igor Saprykin, Amy Shen, Sailesh Sidhwani, Duncan Smith, Joe Stanton, Hamish Tomlinson, Dimple Vijaykumar, Luyu Wang, Piers Wingfield, Nat Wong, Keyang Xu, Christopher Yew, Nick Young, Vadim Zubov, Douglas Eck, Dumitru Erhan, Koray Kavukcuoglu, Demis Hassabis, Zoubin Gharamani, Raia Hadsell, Aäron van den Oord, Inbar Mosseri, Adrian Bolton, Satinder Singh, and Tim Rocktäschel. Genie 3: A new frontier for world models. 2025.

[2] Mathieu Blondel, André F. T. Martins, and Vlad Niculae. Learning classifiers with fenchel-young losses: Generalized entropies, margins, and algorithms, 2019. URL `https://arxiv.org/abs/1805.09717`.

[3] Beidi Chen, Tri Dao, Eric Winsor, Zhao Song, Atri Rudra, and Christopher Ré. Scatterbrain: Unifying sparse and low-rank attention. *Advances in Neural Information Processing Systems*, 34:17413–17426, 2021.

[4] Özgün Çiçek, Ahmed Abdulkadir, Soeren S. Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: Learning dense volumetric segmentation from sparse annotation. In Sebastien Ourselin, Leo Joskowicz, Mert R. Sabuncu, Gozde Unal, and William Wells, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016*, pages 424–432, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46723-8.

[5] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

[6] Tri Dao, Nimit S. Sohoni, Albert Gu, Matthew Eichhorn, Amit Blonder, Megan Leszczynski, Atri Rudra, and Christopher Ré. Kaleidoscope: An efficient, learnable representation for all structured linear maps, 2021. URL `https://arxiv.org/abs/2012.14966`.

[7] Tri Dao, Beidi Chen, Nimit Sohoni, Arjun Desai, Michael Poli, Jessica Grogan, Alexander Liu, Aniruddh Rao, Atri Rudra, and Christopher Ré. Monarch: Expressive structured matrices for efficient and accurate training, 2022. URL `https://arxiv.org/abs/2204.00595`.

[8] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness, 2022. URL `https://arxiv.org/abs/2205.14135`.

[9] Harry Dong, Xinyu Yang, Zhenyu Zhang, Zhangyang Wang, Yuejie Chi, and Beidi Chen. Get more with less: Synthesizing recurrence with kv cache compression for efficient llm inference. *arXiv preprint arXiv:2402.09398*, 2024.

[10] Dan Fu, Simran Arora, Jessica Grogan, Isys Johnson, Evan Sabri Eyuboglu, Armin Thomas, Benjamin Spector, Michael Poli, Atri Rudra, and Christopher Ré. Monarch mixer: A simple sub-quadratic gemm-based architecture. *Advances in Neural Information Processing Systems*, 36:77546–77603, 2023.

[11] Shiqiao Gu, Yang Yong, Fengjun Zhong, Ruihao Gong, Chengtao Lv, Xiangyu Fan, Lei Yang, and Xianglong Liu. Lightvae: Towards compact and efficient video autoencoders, 2026. URL `https://arxiv.org/abs/2610.14726`.

[12] Mutian He and Philip N. Garner. Alleviating forgetfulness of linear attention by hybrid sparse attention and contextualized learnable token eviction, 2025. URL `https://arxiv.org/abs/2510.20787`.

[13] Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion, 2025. URL https://arxiv.org/abs/2506.08009.

[14] Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. Vbench: Comprehensive benchmark suite for video generative models, 2023. URL https://arxiv.org/abs/2311.17982.

[15] Xingyang Li, Muyang Li, Tianle Cai, Haocheng Xi, Shuo Yang, Yujun Lin, Lvmin Zhang, Songlin Yang, Jinbo Hu, Kelly Peng, Maneesh Agrawala, Ion Stoica, Kurt Keutzer, and Song Han. Radial attention: $o(n \log n)$ sparse attention with energy decay for long video generation, 2025. URL https://arxiv.org/abs/2506.19852.

[16] Enshu Liu, Xuefei Ning, Yu Wang, and Zinan Lin. Distilled decoding 1: One-step sampling of image auto-regressive models with flow matching. *arXiv preprint arXiv:2412.17153*, 2024.

[17] Enshu Liu, Qian Chen, Xuefei Ning, Shengen Yan, Guohao Dai, Zinan Lin, and Yu Wang. Distilled decoding 2: One-step sampling of image auto-regressive models with conditional score distillation. *arXiv preprint arXiv:2510.21003*, 2025.

[18] Chengtao Lv, Yumeng Shi, Yushi Huang, Ruihao Gong, Shen Ren, and Wenya Wang. Light forcing: Accelerating autoregressive video diffusion via sparse attention, 2026. URL https://arxiv.org/abs/2602.04789.

[19] 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.

[20] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. URL https://arxiv.org/abs/1505.04597.

[21] Christopher De Sa, Albert Gu, Rohan Puttagunta, Christopher Ré, and Atri Rudra. A two pronged progress in structured dense matrix multiplication, 2017. URL https://arxiv.org/abs/1611.01569.

[22] Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. Flashattention-3: Fast and accurate attention with asynchrony and low-precision, 2024. URL https://arxiv.org/abs/2407.08608.

[23] Wenqiang Sun, Haiyu Zhang, Haoyuan Wang, Junta Wu, Zehan Wang, Zhenwei Wang, Yunhong Wang, Jun Zhang, Tengfei Wang, and Chunchao Guo. Worldplay: Towards long-term geometric consistency for real-time interactive world modeling, 2025. URL https://arxiv.org/abs/2512.14614.

[24] Robbyant Team, Zelin Gao, Qiuyu Wang, Yanhong Zeng, Jiapeng Zhu, Ka Leong Cheng, Yixuan Li, Hanlin Wang, Yinghao Xu, Shuailei Ma, Yihang Chen, Jie Liu, Yansong Cheng, Yao Yao, Jiayi Zhu, Yihao Meng, Kecheng Zheng, Qingyan Bai, Jingye Chen, Zehong Shen, Yue Yu, Xing Zhu, Yujun Shen, and Hao Ouyang. Advancing open-source world models, 2026. URL https://arxiv.org/abs/2601.20540.

[25] Hansi Teng, Hongyu Jia, Lei Sun, Lingzhi Li, Maolin Li, Mingqiu Tang, Shuai Han, Tianning Zhang, WQ Zhang, Weifeng Luo, et al. Magi-1: Autoregressive video generation at scale. *arXiv preprint arXiv:2505.13211*, 2025.

[26] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi,

Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models, 2025. URL https://arxiv.org/abs/2503.20314.

[27] Haocheng Xi, Shuo Yang, Yilong Zhao, Chenfeng Xu, Muyang Li, Xiuyu Li, Yujun Lin, Han Cai, Jintao Zhang, Dacheng Li, Jianfei Chen, Ion Stoica, Kurt Keutzer, and Song Han. Sparse videogen: Accelerating video diffusion transformers with spatial-temporal sparsity, 2025. URL https://arxiv.org/abs/2502.01776.

[28] 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.

[29] Can Yaras, Alec S. Xu, Pierre Abillama, Changwoo Lee, and Laura Balzano. Monarchattention: Zero-shot conversion to fast, hardware-aware structured attention, 2025. URL https://arxiv.org/abs/2505.18698.

[30] Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Fredo Durand, and William T. Freeman. Improved distribution matching distillation for fast image synthesis, 2024. URL https://arxiv.org/abs/2405.14867.

[31] Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T. Freeman, and Taesung Park. One-step diffusion with distribution matching distillation, 2024. URL https://arxiv.org/abs/2311.18828.

[32] 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 Computer Vision and Pattern Recognition Conference*, pages 22963–22974, 2025.

[33] Jintao Zhang, Haofeng Huang, Pengle Zhang, Jia Wei, Jun Zhu, and Jianfei Chen. Sageattention2: Efficient attention with thorough outlier smoothing and per-thread int4 quantization, 2025. URL https://arxiv.org/abs/2411.10958.

[34] Jintao Zhang, Haoxu Wang, Kai Jiang, Shuo Yang, Kaiwen Zheng, Haocheng Xi, Ziteng Wang, Hongzhou Zhu, Min Zhao, Ion Stoica, Joseph E. Gonzalez, Jun Zhu, and Jianfei Chen. Sla: Beyond sparsity in diffusion transformers via fine-tunable sparse-linear attention, 2025. URL https://arxiv.org/abs/2509.24006.

[35] Jintao Zhang, Jia Wei, Haofeng Huang, Pengle Zhang, Jun Zhu, and Jianfei Chen. Sageattention: Accurate 8-bit attention for plug-and-play inference acceleration, 2025. URL https://arxiv.org/abs/2410.02367.

[36] Jintao Zhang, Chendong Xiang, Haofeng Huang, Jia Wei, Haocheng Xi, Jun Zhu, and Jianfei Chen. Spargeattn: Accurate sparse attention accelerating any model inference. In *International Conference on Machine Learning (ICML)*, 2025.

[37] Jintao Zhang, Kaiwen Zheng, Kai Jiang, Haoxu Wang, Ion Stoica, Joseph E Gonzalez, Jianfei Chen, and Jun Zhu. Turbodiffusion: Accelerating video diffusion models by 100-200 times. *arXiv preprint arXiv:2512.16093*, 2025.

[38] 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, 2025. URL https://arxiv.org/abs/2505.13389.

[39] Peiyuan Zhang, Yongqi Chen, Runlong Su, Hangliang Ding, Ion Stoica, Zhengzhong Liu, and Hao Zhang. Fast video generation with sliding tile attention, 2025. URL https://arxiv.org/abs/2502.04507.

[40] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric, 2018. URL https://arxiv.org/abs/1801.03924.

[41] Yang Zhou, Hongyi Liu, Zhuoming Chen, Yuandong Tian, and Beidi Chen. Gsm-infinite: How do your llms behave over infinitely increasing context length and reasoning complexity?, 2025. URL https://arxiv.org/abs/2502.05252.

A MonarchAttention Algorithm

A.1 Original MonarchAttention

Algorithm 1. Below are the updates that MonarchAttention uses in a single iteration, derived from optimizing the objective function with respect to $\boldsymbol{L}$ and $\boldsymbol{R}$ individually:

$$\begin{aligned}
&\boldsymbol{\alpha}^{(t)}_{R,kjv} = \sum_{\ell} \boldsymbol{L}^{(t-1)}_{j\ell k} \boldsymbol{Q}_{\ell jv},\ \boldsymbol{c}^{(t)}_{R,kj} = \sum_{\ell} \boldsymbol{L}^{(t-1)}_{j\ell k} \\
&\boldsymbol{\beta}^{(t)}_{R,kji} = \sum_{v} \boldsymbol{\alpha}^{(t)}_{R,kjv} \boldsymbol{K}_{kiv} \\
&\boldsymbol{R}^{(t)} = \mathrm{softmax}_i\left(\boldsymbol{Z}^{(t)}_R\right), \boldsymbol{Z}^{(t)}_{R,kji} = \boldsymbol{\beta}^{(t)}_{R,kji} / \boldsymbol{c}^{(t)}_{R,kj} \\
&\boldsymbol{\alpha}^{(t)}_{L,jkv} = \sum_{i} \boldsymbol{R}^{(t)}_{kji} \boldsymbol{K}_{kiv},\ \boldsymbol{c}^{(t)}_{L,jk} = \sum_{i} \boldsymbol{R}^{(t)}_{kji} \log \boldsymbol{R}^{(t)}_{kji} \\
&\boldsymbol{\beta}^{(t)}_{L,j\ell k} = \sum_{v} \boldsymbol{\alpha}^{(t)}_{L,jkv} \boldsymbol{Q}_{\ell jv} \\
&\boldsymbol{L}^{(t)} = \mathrm{softmax}_k\left(\boldsymbol{Z}^{(t)}_L\right), \boldsymbol{Z}^{(t)}_{L,j\ell k} = \boldsymbol{\beta}^{(t)}_{L,j\ell k} - \boldsymbol{c}^{(t)}_{L,jk}
\end{aligned}$$

where we interpret $\boldsymbol{Q}$ and $\boldsymbol{K}$ as $b_1 \times b_2 \times d$ tensors. Then, to obtain the output via $\boldsymbol{V}$,

$$\boldsymbol{Y}_{kjv} = \sum_{i} \boldsymbol{R}_{kji} \boldsymbol{V}_{kiv},\ \boldsymbol{O}_{\ell jv} = \sum_{k} \boldsymbol{L}_{j\ell k} \boldsymbol{Y}_{kjv}$$

where we also interpret $\boldsymbol{V}$ as a $b_1 \times b_2 \times d$ tensor. This algorithm is visualized in Figure 5, and its pseudocode is provided in Figure 7a.

A.2 Modified MonarchAttention using Tiled Monarch Parameterization

Under our tiled Monarch parameterization, we adopt the following set of updates for iteratively refining $\boldsymbol{L}$ and $\boldsymbol{R}$, which are also derived from optimizing the objective function with respect to $\boldsymbol{L}$ and $\boldsymbol{R}$ individually:

$$\begin{aligned}
&\boldsymbol{\alpha}^{(t)}_{R,\ell_1 j_1 k_1 i_1 k_2 j_2 v} = \sum_{\ell_2} \boldsymbol{L}^{(t-1)}_{\ell_1 j_1 k_1 i_1 j_2 \ell_2 k_2} \boldsymbol{Q}_{\ell_1 \ell_2 j_1 j_2 v} \\
&\boldsymbol{c}^{(t)}_{R,\ell_1 j_1 k_1 i_1 k_2 j_2} = \sum_{\ell_2} \boldsymbol{L}^{(t-1)}_{\ell_1 j_1 k_1 i_1 j_2 \ell_2 k_2} \\
&\boldsymbol{\beta}^{(t)}_{R,\ell_1 j_1 k_1 i_1 k_2 j_2 i_2} = \sum_{v} \boldsymbol{\alpha}^{(t)}_{R,\ell_1 j_1 k_1 i_1 k_2 j_2 v} \boldsymbol{K}_{k_1 k_2 i_1 i_2 v} \\
&\boldsymbol{R}^{(t)} = \mathrm{softmax}_{i_2}\left(\boldsymbol{Z}^{(t)}_R\right), \\
&\quad \boldsymbol{Z}^{(t)}_{R,\ell_1 j_1 k_1 i_1 k_2 j_2 i_2} = \boldsymbol{\beta}^{(t)}_{R,\ell_1 j_1 k_1 i_1 k_2 j_2 i_2} / \boldsymbol{c}^{(t)}_{R,\ell_1 j_1 k_1 i_1 k_2 j_2} \\
&\boldsymbol{\alpha}^{(t)}_{L,\ell_1 j_1 k_1 i_1 j_2 k_2 v} = \sum_{i_2} \boldsymbol{R}^{(t)}_{\ell_1 j_1 k_1 i_1 k_2 j_2 i_2} \boldsymbol{K}_{k_1 k_2 i_1 i_2 v} \\
&\boldsymbol{c}^{(t)}_{L,\ell_1 j_1 k_1 i_1 j_2 k_2} = \sum_{i_2} \boldsymbol{R}^{(t)}_{\ell_1 j_1 k_1 i_1 k_2 j_2 i_2} \log \boldsymbol{R}^{(t)}_{\ell_1 j_1 k_1 i_1 k_2 j_2 i_2} \\
&\boldsymbol{\beta}^{(t)}_{L,\ell_1 j_1 k_1 i_1 j_2 \ell_2 k_2} = \sum_{v} \boldsymbol{\alpha}^{(t)}_{L,\ell_1 j_1 k_1 i_1 j_2 k_2 v} \boldsymbol{Q}_{\ell_1 \ell_2 j_1 j_2 v} \\
&\boldsymbol{L}^{(t)} = \mathrm{softmax}_{k_1,k_2,i_1}\left(\boldsymbol{Z}^{(t)}_L\right), \\
&\quad \boldsymbol{Z}^{(t)}_{L,\ell_1 j_1 k_1 i_1 j_2 \ell_2 k_2} = \boldsymbol{\beta}^{(t)}_{L,\ell_1 j_1 k_1 i_1 j_2 \ell_2 k_2} - \boldsymbol{c}^{(t)}_{L,\ell_1 j_1 k_1 i_1 j_2 k_2}
\end{aligned}$$

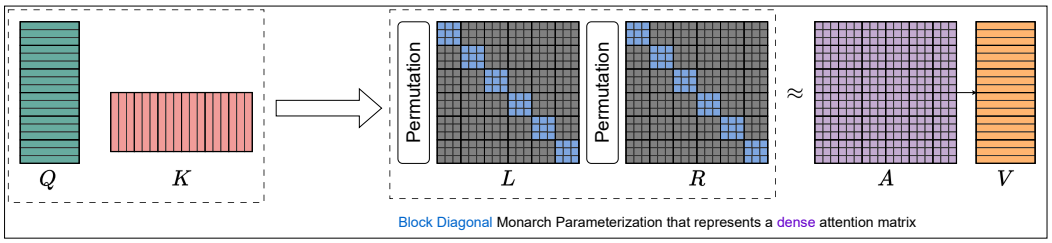


Figure 5: **Overview of the MonarchAttention pipeline.** Given query and key matrices $(\boldsymbol{Q}, \boldsymbol{K})$, MonarchAttention iteratively refines the Monarch factors $\boldsymbol{L}$ and $\boldsymbol{R}$, each composed of sparse block-diagonal matrices. At each iteration, one factor is updated while the other is held fixed, without explicitly materializing the full attention matrix. Despite the highly structured and sparse parameterization of $\boldsymbol{L}$ and $\boldsymbol{R}$, the resulting attention matrix $\boldsymbol{A} \approx \boldsymbol{P}\boldsymbol{L}\boldsymbol{P}^\top\boldsymbol{R}$ is dense, highlighting the strong expressiveness of Monarch parameterization.

where we interpret $\boldsymbol{Q}$ and $\boldsymbol{K}$ as $c_1 \times \frac{b_1}{c_1} \times c_2 \times \frac{b_2}{c_2} \times d$ tensors. Then to obtain the final attention output via $\boldsymbol{V}$:

$$\boldsymbol{Y}_{\ell_1 j_1 k_1 i_1 j_2 k_2 v} = \sum_{i_2} \boldsymbol{R}_{\ell_1 j_1 k_1 i_1 k_2 j_2 i_2} \boldsymbol{V}_{k_1 k_2 i_1 i_2 v}$$

$$\boldsymbol{O}_{\ell_1 \ell_2 j_1 j_2 v} = \sum_{k_1, k_2, i_1} \boldsymbol{L}_{\ell_1 j_1 k_1 i_1 j_2 \ell_2 k_2} \boldsymbol{Y}_{\ell_1 j_1 k_1 i_1 j_2 k_2 v}$$

As these update rules may be difficult to interpret, we have provided pseudocode for this algorithm in Figure 7b. In our efficient Triton kernel implementation, we adopt the same approach as MonarchAttention to avoid materializing $\boldsymbol{L}$ and $\boldsymbol{R}$ in HBM and instead use a FlashAttention-like implementation that computes them on-the-fly in SRAM.

A.3 Kernel Implementation Visualization

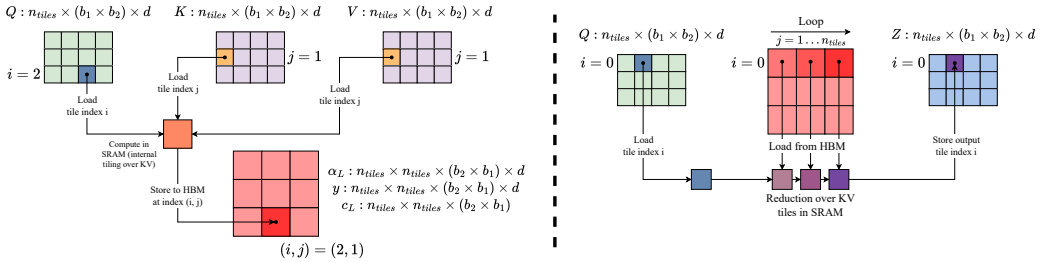


Figure 6: Visualization of efficient kernel implementation for MonarchAttention algorithm with a tiled Monarch parameterization. **Left**: first stage to compute α_L, y, c_L tiles. **Right**: second stage (separate kernel) to compute output by reducing over KV tiles.

B Proofs and Analysis

B.1 Proof for Theorem 3.1

Let $f,\ h,\ w$ denote the number of frames, height, and width respectively, so the total number of tokens is $N = fhw$ and the attention matrix $A \in \mathbb{R}^{N \times N}$ is indexed by pairs of spatiotemporal positions

$$p = (f_0,\ h_0,\ w_0),\ q = (f_1,\ h_1,\ w_1).$$

Assume a row-major ordering of tokens along the time, height, and width dimensions. Under the attention map model in Equation (1), each entry of A decomposes as

$$\boldsymbol{A}_{pq} = \boldsymbol{D}_{pq} + \boldsymbol{S}_{pq} + \epsilon,$$

```
# Q, K, V: (N, d); T: refinement steps; b1, b2: block sizes
def monarch_attention(Q, K, V, T, b1, b2):
    N, d = Q.shape
    L = stack(b2 * [eye(b1)])
    Q = Q.view(b1, b2, d)
    K = K.view(b1, b2, d)

    for t in range(T):
        aR = einsum("jkl,ljv->kjv", L, Q)
        bR = einsum("kjv,kiv->kji", aR, K)
        cR = einsum("jkl->kj", L)
        R = softmax(bR / cR[:, :, None], axis=2)

        aL = einsum("kji,kiv->jkv", R, K)
        bL = einsum("jkv,ljv->jkl", aL, Q)
        cL = einsum("kji->jk", R * log(R))
        L = softmax(bL - cL[:, :, None], axis=1)

    V = V.view(b1, b2, d)
    Y = einsum("kji,kiv->jkv", R, V)
    Z = einsum("jkl,jkv->ljv", L, Y)
    O = Z.reshape(N, d)
    return O
```

(a) MonarchAttention

```
# bb1, bb2: base block sizes; c1, c2: tiling factors
def tiled_monarch_attention(Q, K, V, T, bb1, bb2, c1, c2):
    ntiles = c1 * c2
    b1 = bb1 // c1
    b2 = bb2 // c2
    L = stack(ntiles * [stack(ntiles * [stack(b2 * [eye(b1)])])])
    Q = rearrange(Q, "(a l b j) v -> (a b) l j v", a=c1, b=c2, l=b1, j=b2)
    K = rearrange(K, "(a k b i) v -> (a b) k i v", a=c1, b=c2, k=b1, i=b2)

    for t in range(T):
        aR = einsum("mnjkl,mljv->mnkjv", L, Q)
        bR = einsum("mnkjv,nkiv->mnkji", aR, K)
        cR = einsum("mnjkl->mnkj", L)
        R = softmax(bR / cR[:, :, :, :, None], axis=4)

        aL = einsum("mnkji,nkiv->mnjkv", R, K)
        bL = einsum("mnjkv,mljv->mnjkl", aL, Q)
        cL = einsum("mnkji->mnjk", R * log(R))
        L = softmax(bL - cL[:, :, :, :, None], axis=(1, 3))

    V = rearrange(V, "(a k b i) v -> (a b) k i v", a=c1, b=c2, k=b1, i=b2)
    Y = einsum("mnkji,nkiv->mnjkv", R, V)
    Z = einsum("mnjkl,mnjkv->mljv", L, Y)
    O = rearrange(Z, "(a b) l j v -> (a l b j) v", a=c1, b=c2, l=b1, j=b2)
    return O
```

(b) Tiled MonarchAttention

Figure 7: Pseudocode for MonarchAttention variants. **(a)** Original MonarchAttention [29]. **(b)** Our extension to tiled Monarch parameterization.

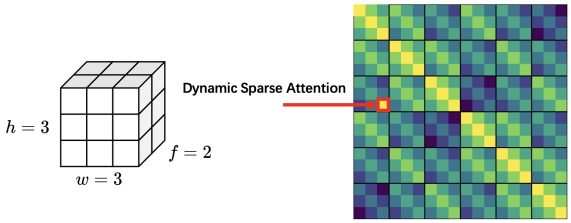


Figure 8: An illustration of our modeling of the 3D attention map in Equation (1). **Left:** the shape of the video. **Right:** an example attention map. The periodic diagonal bands arise from spatiotemporal positional structure, while the large activation at position $(8, 2)$ reflects a semantic relationship that is independent of position, requiring dynamic (retrieval-based) sparse attention to capture.

where the positional term factorizes along the three axes as

$$D_{pq} = d_w(w_0, w_1)d_h(h_0, h_1)d_t(f_0, f_1).$$

We show that $\boldsymbol{D}$ can be written as a row-wise permutation of a blockwise rank-1 matrix $\boldsymbol{D}'$ (using permutation $\boldsymbol{P}$).

Assume a standard row-major token ordering on the frame, height, and width indices, the width dimension being contiguous in the flattened token sequence. Concretely, the absolute index $\phi(f_0, h_0, w_0)$ of a token with spatiotemporal indices (f_0, h_0, w_0) is

$$\phi(f_0, h_0, w_0) = ((f_0 h) + h_0)w + w_0.$$

We can also consider a permuted token ordering that makes width noncontiguous:

$$\rho(f_0, h_0, w_0) = w_0 fh + (f_0 h + h_0).$$

Let $\boldsymbol{P} \in \mathbb{R}^{N\times N}$ be the permutation matrix that maps the token ordering $\rho \to \phi$, i.e.

$$\boldsymbol{P}_{\phi(f_0,h_0,w_0),\rho(f_0,h_0,w_0)} = 1$$

and zero elsewhere. Then define

$$\boldsymbol{D}' = \boldsymbol{P}^\top \boldsymbol{D},$$

so that D' is obtained from D by a row-wise permutation that maps the original row-major indices from ϕ to column-major indices given by ρ. Importantly,

$$\boldsymbol{D} = \boldsymbol{P}\boldsymbol{D}'$$

We now prove that $\boldsymbol{D}'$ is blockwise rank-1. Using block sizes $(b_1, b_2) = (fh, w)$, we take a 4D blocked view of $\boldsymbol{D}'$ with shape (b_2, b_1, b_1, b_2), so $\boldsymbol{D}'_{j,:,k,:}$ is a block of size $b_1 \times b_2$ for all $j \in [b_2], k \in [b_1]$. Due to the row-wise permutation, the rows of $\boldsymbol{D}'$ (corresponding to query tokens) follow the column-major ordering while the columns (corresponding to key tokens) follow the row-major ordering. For convenience, let us define

$$\sigma(f_0, h_0) = f_0 h + h_0$$

to map the frame and height indices for a given token to a combined (row-major) time/height index so that $\sigma(f_0, h_0) \in [b_1]$. Then we can also define $d_{t,h}$:

$$d_{t,h}(\sigma(f_0, h_0), \sigma(f_1, h_1)) = d_t(f_0, f_1)d_h(h_0, h_1)$$

which takes combined time/height indices for two tokens and computes the combined time-wise and height-wise components of the positional attention score for the two tokens.

Let $\boldsymbol{B}^{(j,k)} = \boldsymbol{D}'_{j,:,k,:} \in \mathbb{R}^{b_1\times b_2}$ denote the (j, k)-th block of $\boldsymbol{D}'$. Note that columns of $\boldsymbol{B}^{(j,k)}$ correspond to the key/value tokens with the same time/height index while rows of $\boldsymbol{B}^{(j,k)}$, due to the row-wise permutation for $\boldsymbol{D}'$, correspond to query tokens with the same width index. This means

$$\boldsymbol{B}^{(j,k)}_{i,l} = d_w(j, l)d_{t,h}(i, k)$$

Alternatively,

$$\boldsymbol{B}^{(j,k)} = \begin{bmatrix} d_{t,h}(0,k) \\ d_{t,h}(1,k) \\ \vdots \\ d_{t,h}(b_1-1,k) \end{bmatrix} \begin{bmatrix} d_w(j,0) \\ d_w(j,1) \\ \vdots \\ d_w(j,b_2-1) \end{bmatrix}^\top$$

So $\boldsymbol{B}^{(j,k)}$ is rank-1, meaning $\boldsymbol{D}'$ is blockwise rank-1.

Therefore, since Equation (1) defines

$$\boldsymbol{A} = \boldsymbol{D} + \boldsymbol{S} + \epsilon$$

and since

$$\boldsymbol{D} = \boldsymbol{P}\boldsymbol{D}'$$

then

$$\boldsymbol{A} = \boldsymbol{P}\boldsymbol{D}' + \boldsymbol{S} + \epsilon$$

where $\boldsymbol{P}$ is a permutation matrix, $\boldsymbol{D}'$ is blockwise rank-1 with block sizes (b_1, b_2) that satisfy $b_1 b_2 = fhw$, and $\boldsymbol{S}$ is a sparse matrix.

Note that the proof can be generalized for any of the "proper" block sizes specified in Section 4.1, so long as the appropriate sequence flattening order and permutation (defined by ϕ and ρ) are used.

B.2 Proof for Theorem B.1

Theorem B.1 (Strict expressiveness of tiled Monarch)**.** *Fix base block sizes $b_1 \geq 2$ and $b_2 \geq 2$ with $N = b_1 b_2$, and let $c_1 \mid b_1$, $c_2 \mid b_2$ be tiling factors. Let $\mathcal{M}(b_1, b_2)$ denote the set of Monarch matrices with block sizes (b_1, b_2), i.e., matrices $\boldsymbol{M} \in \mathbb{R}^{N \times N}$ that admit factors $\boldsymbol{L} \in \mathbb{R}^{b_2 \times b_1 \times b_1}$, $\boldsymbol{R} \in \mathbb{R}^{b_1 \times b_2 \times b_2}$ satisfying $\boldsymbol{M}_{(\ell b_2 + j)(k b_2 + i)} = \boldsymbol{L}_{j\ell k}\, \boldsymbol{R}_{kji}$. Let $\mathcal{M}_{\text{tile}}(b_1, b_2; c_1, c_2)$ denote the set of matrices representable by the tiled Monarch parameterization with the same base block sizes (b_1, b_2) and tiling factors (c_1, c_2). Then $\mathcal{M}(b_1, b_2) \subseteq \mathcal{M}_{\text{tile}}(b_1, b_2; c_1, c_2)$, and if $c_1 > 1$ or $c_2 > 1$, the inclusion is strict: $\mathcal{M}(b_1, b_2) \subset \mathcal{M}_{\text{tile}}(b_1, b_2; c_1, c_2)$.*

(Containment). Let $\boldsymbol{M} \in \mathcal{M}(b_1, b_2)$ with factors $(\boldsymbol{L}, \boldsymbol{R})$. Let $c_1 \mid b_1$, $c_2 \mid b_2$ be tiling factors, and write $\tilde{b}_1 := b_1/c_1$, $\tilde{b}_2 := b_2/c_2$. Now define tiled factors $(\boldsymbol{L}', \boldsymbol{R}')$ by parameter tying: for all valid indices,

$$\boldsymbol{L}'_{\ell_1, j_1, k_1, i_1, j_2, \ell_2, k_2} := \boldsymbol{L}_{j,\ell,k} \quad \text{with} \quad j = j_1 \tilde{b}_2 + j_2,\ \ell = \ell_1 \tilde{b}_1 + \ell_2,\ k = k_1 \tilde{b}_1 + k_2,$$

i.e., $\boldsymbol{L}'$ ignores i_1, and

$$\boldsymbol{R}'_{\ell_1, j_1, k_1, i_1,\ k_2, j_2, i_2} := \boldsymbol{R}_{k,j,i} \quad \text{with} \quad i = i_1 \tilde{b}_2 + i_2,\ j = j_1 \tilde{b}_2 + j_2,\ k = k_1 \tilde{b}_1 + k_2,$$

i.e., $\boldsymbol{R}'$ ignores ℓ_1. Substituting these definitions into the tiled formula yields

$$\boldsymbol{L}'_{\ell_1, j_1, k_1, i_1,\ j_2, \ell_2, k_2}\ \boldsymbol{R}'_{\ell_1, j_1, k_1, i_1,\ k_2, j_2, i_2} = \boldsymbol{L}_{j\ell k}\, \boldsymbol{R}_{kji} = \boldsymbol{M}_{(\ell b_2 + j)(k b_2 + i)},$$

so $\boldsymbol{M} \in \mathcal{M}_{\text{tile}}(b_1, b_2; c_1, c_2)$. Hence $\mathcal{M}(b_1, b_2) \subseteq \mathcal{M}_{\text{tile}}(b_1, b_2; c_1, c_2)$.

(Strictness). Assume $c_1 > 1$ or $c_2 > 1$, so at least one of $\tilde{b}_1 < b_1$ or $\tilde{b}_2 < b_2$ holds, meaning that a slice $\boldsymbol{B}^{(j,k)} \in \mathbb{R}^{b_1 \times b_2}$ can contain *multiple* rank-1 tiles. We construct a tiled matrix whose $(j, k) = (0, 0)$ slice has rank 2, which is impossible to represent exactly for untiled Monarch.

Consider the slice $\boldsymbol{B}^{(0,0)}$ (fix $j = 0, k = 0$). Pick two distinct rows $r_1 \neq r_2$ and two distinct columns $s_1 \neq s_2$ such that (r_1, s_1) and (r_2, s_2) lie in *different tiles* of the $c_1 \times c_2$ tiling of $\{0, \ldots, b_1 - 1\} \times \{0, \ldots, b_2 - 1\}$. (This is always possible when $c_1 > 1$ or $c_2 > 1$: if $c_1 > 1$, choose r_1 and r_2 from different row-tiles; if $c_2 > 1$, choose s_1 and s_2 from different column-tiles.)

Define a tiled Monarch matrix $\boldsymbol{M}$ by setting all tiles to zero except the two tiles containing (r_1, s_1) and (r_2, s_2), and within each of these two tiles choose local factors so that the tile equals a single-entry rank-1 matrix with value 1 at that coordinate (and zeros elsewhere). This is feasible because each tile independently parameterizes an arbitrary rank-1 matrix on its $\tilde{b}_1 \times \tilde{b}_2$ support.

Then $\boldsymbol{B}^{(0,0)}$ has exactly two nonzero entries: $\boldsymbol{B}^{(0,0)}_{r_1,s_1} = 1$ and $\boldsymbol{B}^{(0,0)}_{r_2,s_2} = 1$. The 2×2 submatrix of $\boldsymbol{B}^{(0,0)}$ restricted to rows $\{r_1, r_2\}$ and columns $\{s_1, s_2\}$ is the identity matrix, hence $\mathrm{rank}(\boldsymbol{B}^{(0,0)}) \geq 2$. Therefore $\boldsymbol{M} \notin \mathcal{M}(b_1, b_2)$, since every (j, k) slice of an untiled Monarch matrix must have rank at most 1. But by construction $\boldsymbol{M} \in \mathcal{M}_{\mathrm{tile}}(b_1, b_2; c_1, c_2)$. Thus the containment is strict whenever $c_1 > 1$ or $c_2 > 1$.

B.3 Per-Block Rank-1 Case Analysis

In this section, we provide a case-by-case analysis supporting the observation in Section 3.3 that blocks of the permuted attention matrix are often well approximated by rank-1 matrices. We consider each block

$$\widetilde{\boldsymbol{A}}_{[\boldsymbol{i},\boldsymbol{j}]} = \boldsymbol{D'}_{[\boldsymbol{i},\boldsymbol{j}]} + \widetilde{\boldsymbol{S}}_{[\boldsymbol{i},\boldsymbol{j}]} + \epsilon \in \mathbb{R}^{b_1 \times b_2},$$

where $\widetilde{\boldsymbol{A}} = \boldsymbol{P}^\top \boldsymbol{A}$, $\widetilde{\boldsymbol{S}} = \boldsymbol{P}^\top \boldsymbol{S}$, $[i, j]$ indexes the block positions in the partitioned attention matrix (i.e. $\widetilde{\boldsymbol{A}}_{[\boldsymbol{i},\boldsymbol{j}]} = \widetilde{\boldsymbol{A}}_{[ib_1:(i+1)b_1, jb_2:(j+1)b_2]}$), and b_1, b_2 denote the block sizes. We now intuitively analyze three representative cases, also illustrated in Figure 9. In these idealized cases, we omit residual noise and terms explicitly described as negligible.

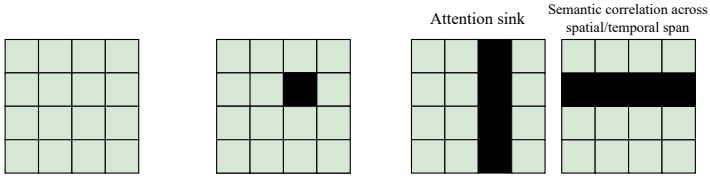


Figure 9: Illustration of several representative cases regarding occurrences of sparse, semantic correlations from a per-block ($\widetilde{\boldsymbol{A}}_{[\boldsymbol{i},\boldsymbol{j}]}$) view of the full Monarch-parameterized attention map ($\boldsymbol{A}$).

Case 1. If $\mathrm{NNZ}(\widetilde{\boldsymbol{S}}_{[\boldsymbol{i},\boldsymbol{j}]}) = 0$, the block is governed by the positional term. As the positional structure factorizes along the three dimensions, $\widetilde{\boldsymbol{A}}_{[\boldsymbol{i},\boldsymbol{j}]}$ is rank-1.

Case 2. When $\mathrm{NNZ}(\widetilde{\boldsymbol{S}}_{[\boldsymbol{i},\boldsymbol{j}]}) = 1$ and the positional term $\boldsymbol{D'}_{[\boldsymbol{i},\boldsymbol{j}]}$ is negligible, the block is dominated by a single semantic interaction between two distant tokens. The single nonzero entry in $\widetilde{\boldsymbol{S}}_{[\boldsymbol{i},\boldsymbol{j}]}$ makes $\widetilde{\boldsymbol{A}}_{[\boldsymbol{i},\boldsymbol{j}]}$ rank-1.

Case 3. When $\mathrm{NNZ}(\widetilde{\boldsymbol{S}}_{[\boldsymbol{i},\boldsymbol{j}]}) > 1$ but all nonzero entries lie within a single row or a single column and the positional term is negligible, the block contains multiple semantic connections originating from (or pointing to) the same token. This pattern includes the common attention-sink phenomenon. Since these semantic interactions are confined to one row or column, $\widetilde{\boldsymbol{A}}_{[\boldsymbol{i},\boldsymbol{j}]}$ is rank-1.

B.4 Attention Maps as Monarch Matrices

In this section, we formalize the notion from Section 3.3 that an attention map with exact rank-1 blocks, as in the idealized cases of Section B.3, can be represented as a Monarch matrix via permutation. We use the same blocked indexing and notation as in Section 3.3.

By assumption, the permuted attention matrix $\widetilde{\boldsymbol{A}} \in \mathbb{R}^{N \times N}$ is partitioned into blocks $\widetilde{\boldsymbol{A}}_{[j,k]} \in \mathbb{R}^{b_1 \times b_2}$, with $j \in [b_2], k \in [b_1]$ and $b_1 b_2 = N$, and each block is rank-1. Thus, for every (j, k) there exist vectors $\boldsymbol{u}^{(j,k)} \in \mathbb{R}^{b_1}$ and $\boldsymbol{v}^{(j,k)} \in \mathbb{R}^{b_2}$ such that

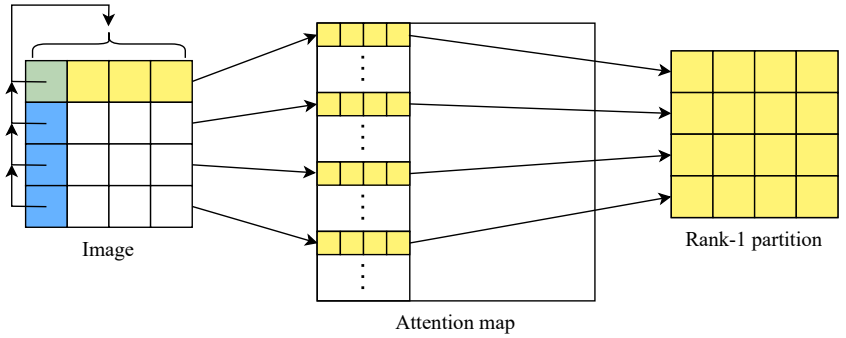
$$\widetilde{\boldsymbol{A}}_{[j,k]} = \boldsymbol{u}^{(j,k)} (\boldsymbol{v}^{(j,k)})^\top, \quad \text{i.e.} \quad \widetilde{\boldsymbol{A}}_{[j,k]}(\ell, i) = \boldsymbol{u}^{(j,k)}_\ell \, \boldsymbol{v}^{(j,k)}_i$$

We now construct the Monarch factors directly from these blockwise rank-1 decompositions. Define

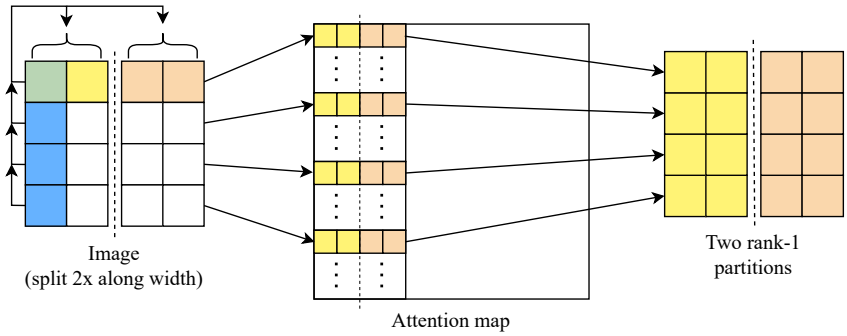
$$\boldsymbol{L}_{j\ell k} = \boldsymbol{u}^{(j,k)}_\ell, \qquad \boldsymbol{R}_{kji} = \boldsymbol{v}^{(j,k)}_i$$

Then, under the same blocked indexing used for Monarch matrices in Section 2, we have

$$\widetilde{\boldsymbol{A}}_{[j,k]}(\ell, i) = \boldsymbol{L}_{j\ell k} \, \boldsymbol{R}_{kji}$$



(a) Untiled (h, w) block size partitioning



(b) Tiled $(h, \frac{w}{2})$

Figure 10: Illustration of untiled/tiled block size partitioning schemes.

Since $\boldsymbol{A} = \boldsymbol{P}\widetilde{\boldsymbol{A}}$, then we also have

$$\boldsymbol{A}_{\ell jki} = \boldsymbol{L}_{j\ell k}\,\boldsymbol{R}_{kji}$$

when we take the 4D view of $\boldsymbol{A}$. This matches the Monarch parameterization exactly. Therefore $\boldsymbol{A} = \boldsymbol{P}\widetilde{\boldsymbol{A}}$ is a Monarch matrix with block sizes (b_1, b_2) when $\widetilde{\boldsymbol{A}}$ is blockwise rank-1.

B.5 Block Size Analysis

For simplicity, let us consider the single-image case ($f = 1$). Based on the block size analysis conducted in Section 4.1, block sizes (h, w) would enable a proper Monarch parameterization of the positional component of the $hw \times hw$ attention map. With these block sizes, the parameterization effectively forms partitions of the attention map that contain pairwise attention scores between all tokens in a given column and all tokens in a given row of the image. This allows each partition to be rank-1 (under the assumptions from Equation (1)), enabling a proper parameterization. We illustrate this blocking scheme in Figure 10a. Using different block sizes (e.g. $(2h, \frac{w}{2})$) would cause each partition to span all tokens along multiple columns or multiple rows of the image, which would remove the rank-1 structure as previously noted. Importantly, this restricts block sizes to be equivalent to the video dimensions.

In Section 4.2, we introduce a tiled Monarch parameterization to reduce this constraint. Effectively, our modified parameterization to an extent enables decoupling the block size selection from the actual video dimensions. We illustrate how this parameterization functions in Figure 10b in the case where we select one of the block sizes as $\frac{w}{2}$. In the tiled parameterization, by applying the appropriate permutation/blocking, we gain much more flexibility in how we select our block sizes, which decide how rank-1 partitions are formed.

B.6 Aligned Construction for Positional Attention

We provide a formal construction supporting the alignment claim in Section 4.1: that an aligned Monarch parameterization exactly reproduces the separable positional attention model of Equation (1).

Method	Sparsity	PSNR (↑)	SSIM (↑)	LPIPS (↓)	VBench (↑)	RTX 5090		H100		B200	
						E2E (s)	Kernel (ms)	E2E (s)	Kernel (ms)	E2E (s)	Kernel (ms)
Dense (TF ref.)	0%	—	—	—	0.826	137.115	29.75	47.041	9.40	25.250	4.79
Training-free											
SpargeAttn	85%	12.344	0.300	0.671	0.795	59.309	3.13	26.339	3.06	21.498	4.07
SVG	85%	9.361	0.176	0.776	0.750	76.496	8.93	33.453	5.20	21.532	4.00
SVG2	85%	11.108	0.303	0.673	0.797	95.608	18.77	47.230	10.40	37.689	9.88
RadialAttn	85%	12.350	0.267	0.720	0.681	79.150	9.78	36.104	5.91	26.682	5.79
MONARCHRT	90%	**12.624**	**0.360**	**0.650**	0.801	87.130	12.82	38.984	7.46	26.121	5.10
Trained											
Dense (ref.)	0%	—	—	—	0.839	137.115	29.75	47.041	9.40	25.250	4.79
VSA	85%	—	—	—	0.819	81.810	12.34	34.985	6.11	22.588	4.72
MONARCHRT	97%	—	—	—	**0.835**	**55.725**	**2.19**	**21.971**	**1.47**	**14.342**	**1.28**

Table 3: Bidirectional Wan2.1-1.3B, 50 steps, 81 frames at 480p.

To formalize this notion, we consider a simplified setting where the attention follows the purely positional model in Equation (1) by temporarily assuming $\boldsymbol{S} = 0$. Recall that a Monarch matrix is parameterized by two block-diagonal factors $\boldsymbol{L}$ and $\boldsymbol{R}$, which can be interpreted as 3D tensors of shape $b_2 \times b_1 \times b_1$ and $b_1 \times b_2 \times b_2$, respectively, with block sizes b_1 and b_2 satisfying $b_1 b_2 = fhw$.

Consider choosing block sizes $b_1 = fh$ and $b_2 = w$. In this case, $\boldsymbol{L}$ has shape $w \times fh \times fh$ and $\boldsymbol{R}$ has shape $fh \times w \times w$. Under this parameterization, the post-softmax attention scores can be written as

$$\boldsymbol{A}_{(f_0,h_0,w_0),(f_1,h_1,w_1)} = \boldsymbol{L}_{w_0,(f_0,h_0),(f_1,h_1)} \boldsymbol{R}_{(f_1,h_1),w_0,w_1}.$$

By setting

$$\boldsymbol{L}_{w_0,(f_0,h_0),(f_1,h_1)} = d_t(f_0, f_1)\, d_h(h_0, h_1),$$
$$\boldsymbol{R}_{(f_1,h_1),w_0,w_1} = d_w(w_0, w_1),$$

the Monarch parameterization exactly reproduces the fully separable positional attention model in Equation (1). Importantly, any permutation of the dimensions across the block sizes, such as $(b_1, b_2) = (f, hw)$ or $(b_1, b_2) = (fw, h)$, admits an equivalent exact decomposition.

C Block Alignment Illustration

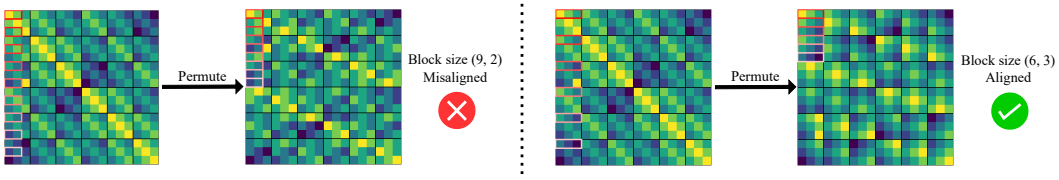


Figure 11: The comparison of two Monarch factorizations. **Left:** By selecting misaligned block sizes, most of the blocks are obviously not rank-1, losing the ability to represent positional relationships. **Right:** With aligned block sizes, the factorization is able to recover positional relationships.

D Extended Empirical Results

D.1 Bidirectional Quality and Latency

Table 3 shows total VBench scores alongside holistic metrics, namely PSNR, SSIM, and LPIPS [40] for bidirectional 50-step Wan2.1-1.3B [26]. The same table also shows corresponding attention kernel and E2E generation latency. Baselines include native SpargeAttention, SVG, SVG2 [28], RadialAttention, and VSA [38]. Figure 12 shows speedup over dense across the MONARCHRT full sparsity sweep. These results are extended to a distilled 4-step version of Wan2.1-1.3B in Table 4.

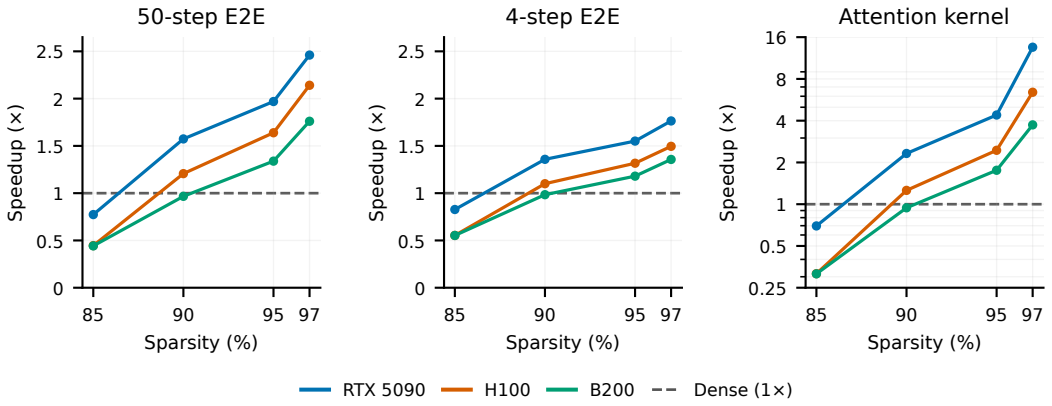


Figure 12: Monarch speedup over dense across the full bidirectional sparsity sweep. Left to right: 50-step E2E, 4-step E2E, and attention kernel speedup.

The reported SVG2 kernel latencies exclude centroid initialization, which occurs only during the first denoising step; E2E timings include this initialization.

Results. Trained MONARCHRT exceeds trained VSA on total VBench for both 50-step and 4-step models, using 97% and 95% sparsity respectively for MONARCHRT and 85% for VSA in both settings, with lower measured latency on each GPU in all cases. Among 50-step settings, MONARCHRT at 97% sparsity is fastest on all three GPUs. MONARCHRT at 90% sparsity achieves the best PSNR, SSIM, and LPIPS over all other sparse methods on the 4-step distilled Wan model, while remaining competitive on VBench. Analogous holistic-metric improvements hold on the 50-step model.

D.2 Empirical Rank Structure of Attention Blocks

Setup. We use ten randomly selected VBench prompts, generating one Self-Forcing video per prompt. We collect attention inputs across all 30 transformer layers, 12 attention heads, 4 denoising steps, and 7 autoregressive decoding positions. We examine the blocks induced by the aligned untiled and tiled parameterizations of the dense post-softmax attention matrix.

For a nonzero block B, define the fraction of squared Frobenius norm in its leading singular component and its stable rank as

$$e(B) = \frac{\sigma_1(B)^2}{\|B\|_F^2}, \qquad \mathrm{sr}(B) = \frac{\|B\|_F^2}{\|B\|_2^2} = \frac{1}{e(B)}. \tag{3}$$

Both equal 1 for a rank-1 block. Every block is weighted equally to compute the means in Table 5. The mean stable rank is therefore not generally the reciprocal of the mean leading-component fraction.

Departures from rank-1. Equal-weight block means are nonmonotonic in sparsity, because changing the partition affects both block geometry and the contribution of near-empty softmax-tail blocks to these averages. A block can have a diffuse relative spectrum yet contribute little absolute approximation error when its norm is small. To quantify this distinction, Table 6 reports the fraction of total squared Frobenius norm in blocks below each threshold:

$$m(\tau) = \frac{\sum_{B:\, e(B)<\tau} \|B\|_F^2}{\sum_B \|B\|_F^2}. \tag{4}$$

At 95% sparsity, blocks with $e(B) < 0.7$ make up 9.13% of blocks but hold only 0.73% of total squared Frobenius norm. Across the three configurations, less than 6% of squared Frobenius norm lies in blocks with $e(B) < 0.9$. At each threshold this mass decreases with less sparsity in the measured configurations. These results support approximate rank-1 structure in the blocks carrying most attention mass. We do not require every block to satisfy an exact rank constraint to obtain a high-quality approximation.

Method	Sparsity	PSNR (↑)	SSIM (↑)	LPIPS (↓)	VBench (↑)	RTX 5090		H100		B200	
						E2E (s)	Kernel (ms)	E2E (s)	Kernel (ms)	E2E (s)	Kernel (ms)
Dense	0%	—	—	—	0.837	7.576	29.78	2.930	9.38	1.674	4.77
Training-free											
SpargeAttn	85%	12.883	0.328	0.616	**0.838**	**4.393**	**3.11**	**2.100**	**3.03**	1.530	4.07
SVG	85%	10.406	0.192	0.748	0.757	5.308	8.93	2.387	5.16	1.528	4.02
SVG2	85%	12.095	0.304	0.635	0.824	8.539	17.11	4.797	9.62	3.559	9.32
RadialAttn	85%	11.890	0.263	0.721	0.729	5.203	9.65	2.488	5.90	1.730	5.79
MonarchRT	90%	**13.193**	**0.349**	**0.609**	0.826	5.579	12.83	2.663	7.46	1.701	5.07
Trained											
VSA	85%	—	—	—	0.821	5.402	12.34	2.457	6.11	1.551	4.72
MonarchRT	95%	—	—	—	0.832	4.885	6.78	2.225	3.83	**1.419**	**2.72**

Table 4: Bidirectional Wan2.1-1.3B, 4 steps, 81 frames at 480p.

Table 5: Blockwise rank statistics on ten Self-Forcing videos. The leading-component fraction e and stable rank sr are defined in Equation (3). Note that values of 1.00 are within numerical precision and do not establish exact rank-1.

			Quantiles of $e(B)$					
Configuration	Mean e	Mean sr	p1	p5	p10	p25	p50	p75
Aligned (97%)	0.887	1.170	0.44	0.58	0.67	0.83	0.95	1.00
Aligned+tiled (95%)	0.910	1.131	0.48	0.62	0.72	0.88	0.98	1.00
Aligned+tiled (90%)	0.897	1.150	0.47	0.60	0.69	0.85	0.96	1.00

D.3 Approximation Error, Refinement, and Video Quality

Using the same ten-prompt attention map data, we compare the dense post-softmax attention matrix A with the matrix $\widehat{A}$ induced by Monarch factors. We measure relative Frobenius error $\|\widehat{A}-A\|_F/\|A\|_F$ while varying sparsity and the number of refinement iterations $T \in \{1, 2, 3, 5, 10\}$. Figure 13 shows the reported mean errors: every measured configuration improves with additional iterations, and less sparse configurations have lower error at each tested T.

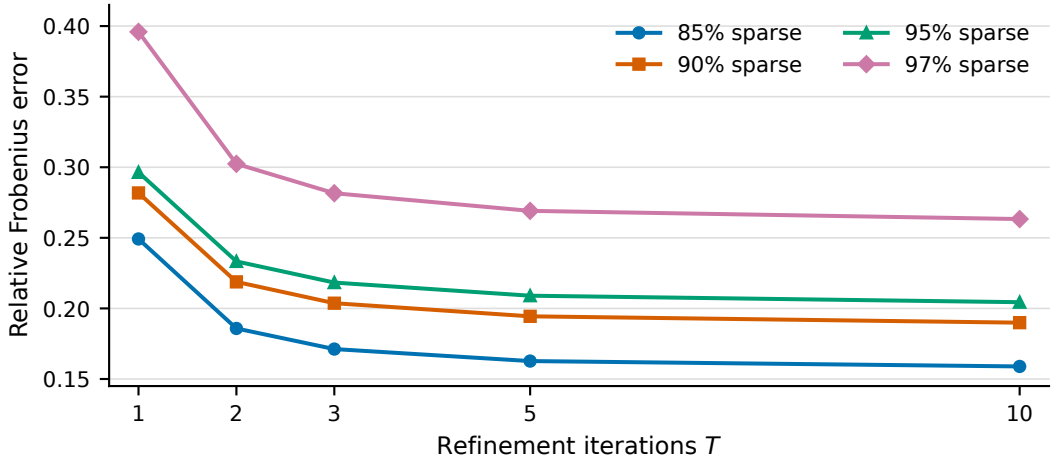


Figure 13: Relative attention-matrix approximation error on ten Self-Forcing prompt trajectories, covering all layers, heads, denoising steps, and autoregressive positions. More refinement and less sparsity both reduce the measured error. Curves show aggregate errors; prompt-level uncertainty is not estimated.

Table 6: Attention mass in blocks below each leading-component threshold, measured by $m(\tau)$ in Equation (4). Blocks that depart most from rank-1 account for a small fraction of total squared Frobenius norm.

	Fraction of total squared Frobenius norm				
Configuration	$e < 0.5$	$e < 0.6$	$e < 0.7$	$e < 0.8$	$e < 0.9$
Aligned (97%)	0.22%	0.76%	1.63%	2.96%	5.53%
Aligned+tiled (95%)	0.08%	0.32%	0.73%	1.36%	2.57%
Aligned+tiled (90%)	0.06%	0.26%	0.62%	1.19%	2.33%

Attention error vs. downstream quality. Expressiveness, estimation error, and generated-video quality represent different forms of quality. Firstly, nested MONARCHRT tiling configurations expose a larger feasible matrix class and can only improve the best attainable matrix approximation error. However, fixed-iteration factor estimation does not necessarily attain that optimum: while Figure 13 suggests that even $T = 1$ produces monotonically improving mean attention error as effective sparsity decreases, this is not strictly guaranteed. The nonmonotonic VBench result in Table 1 therefore cannot be attributed to an observed reversal of this mean error. An aggregate attention error does not fully characterize how approximation changes propagate through denoising and autoregressive generation, which can unpredictably affect downstream evaluation scores.

D.4 Example Generations

We provide additional example generations for Self-Forcing and Wan in Figures 15 and 16, respectively. We use 5 prompts from Movie Gen Video Bench [19] to generate videos for the dense model and MONARCHRT. The sample videos demonstrate that MONARCHRT can achieve comparable visual quality to the dense model.



Figure 14: Example generations on Self-Forcing for exact top-k and MONARCHRT with 1 and 20 iterative refinement steps (training-free).



Figure 15: Example generations for dense baseline and MONARCHRT (with 95% sparsity) on the same prompts for Self-Forcing.

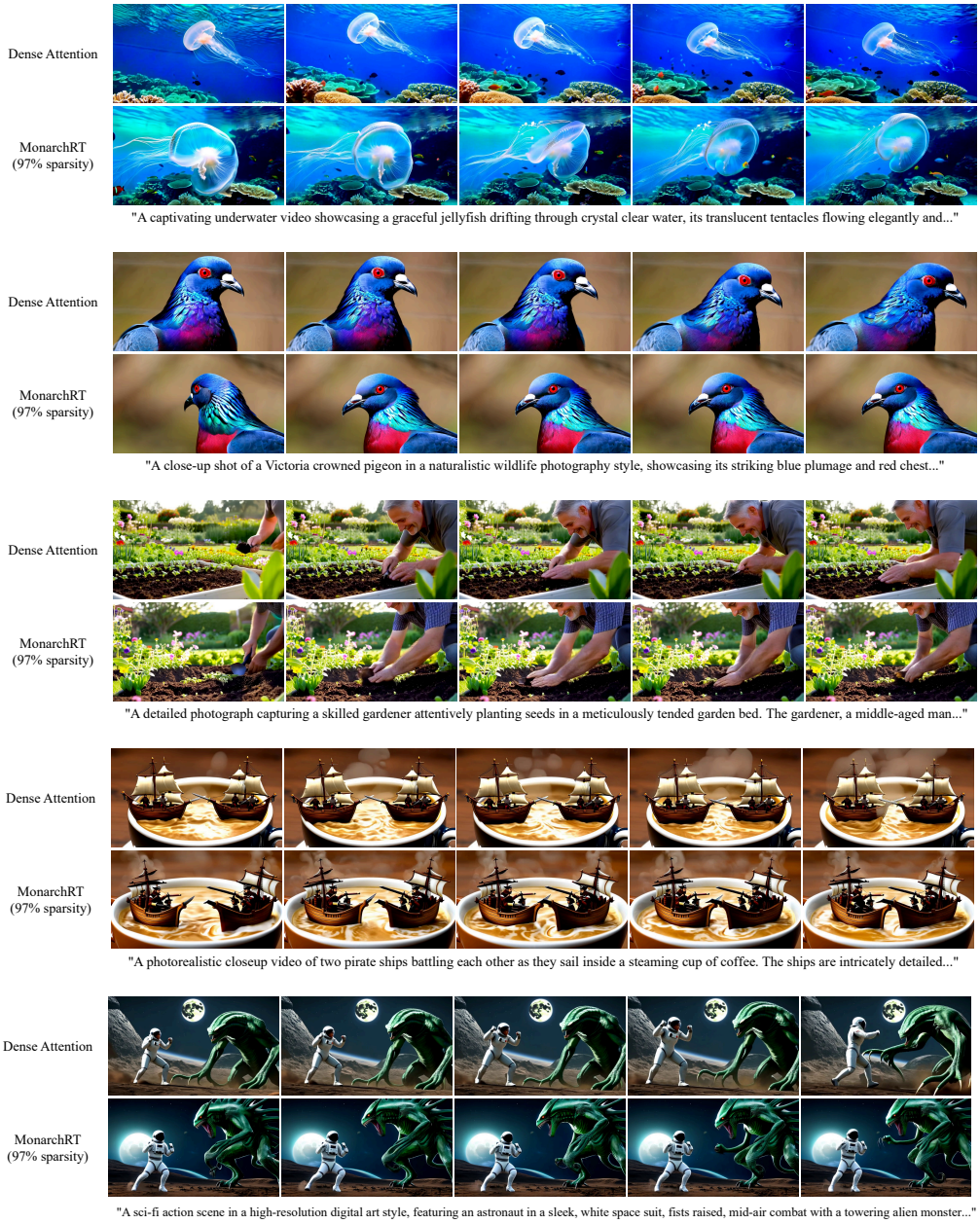


Figure 16: Example generations for dense baseline and trained MONARCHRT (with 97% effective sparsity) on the same prompts for Wan 2.1-1.3B (50-step bidirectional model).

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in the abstract and introduction correspond to results we demonstrate in our experiments section or additional ablations.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 7 discusses the structural assumptions and potential limitations when extending to non-video modalities, high-motion videos, and scene transitions.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: All proofs are provided in appendix sections, which include detail on the required assumptions and steps to complete each proof.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All experiments are performed on known datasets with clearly specified baselines and configurations.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

Justification: The code repository is provided, and it provides instructions on how to run the code.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The code provides the appropriate settings and instructions for datasets and hyperparameters for training and testing.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Kernel latency plots indicate 10th–90th percentile ranges. The protocol distinguishes observed timing spread from confidence intervals. VBench scores are not typically accompanied by uncertainty estimates as they are already robust measures based on multiple video generations per prompt (we generate 5 videos per prompt).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The required compute configurations are provided in the code.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: All authors respect the ethics guidelines.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Section 7 discusses wider access to video generation and potential simulation and robotics applications, alongside risks from deceptive synthetic videos and misinformation.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes

for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: No data or models are released as part of this work.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All relevant works are properly cited, and their licenses and terms of use are respected.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

Justification: Our released code repository includes installation, training, and inference documentation. No new datasets or trained model checkpoints are released.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: There are no human subjects in this project.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: There are no human subjects in this project.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: Ideating and implementing this work did not require the usage of LLMs.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.