STILL: Selecting Tokens for Intra-Layer Hybrid Attention to Linearize LLMs

Weikang Meng[1,2,*] **Liangyu Huo**[1,*] **Yadan Luo**[3] **Jiawen Guan**[1]
Jingyi Zhang[4] **Yingjian Li**[2,†] **Zheng Zhang**[1,2,†]

[1]Harbin Institute of Technology, Shenzhen [2]Pengcheng Laboratory
[3]UQMM Lab, University of Queensland [4]Relty (Shanghai) Technology Co., Ltd.

zacharymengwk@gmail.com yingjian1122@gmail.com darrenzz219@gmail.com
*Equal contribution
†Corresponding authors

Abstract

While alleviating the quadratic complexity of Softmax attention is crucial, training large linear models from scratch is computationally prohibitive. Thus, linearizing pretrained LLMs via intra-layer hybrid architectures has emerged as the indispensable paradigm. Existing methods perform token routing based on sliding-window partitions, resulting in position-based selection and fails to capture token-specific global importance. Meanwhile, linear attention further suffers from distribution shift caused by learnable feature maps that distort pretrained feature magnitudes. Motivated by these limitations, we propose STILL, an intra-layer hybrid linearization framework for efficiently linearizing LLMs. STILL introduces a Self-Saliency Score with strong local–global consistency, enabling accurate token selection using sliding-window computation, and retains salient tokens for sparse softmax attention while summarizing the remaining context via linear attention. To preserve pretrained representations, we design a Norm-Preserved Feature Map (NP-Map) that decouples feature direction from magnitude and reinjects pretrained norms. We further adopt a unified training–inference architecture with chunk-wise parallelization and delayed selection to improve hardware efficiency. Experiments show that STILL matches or surpasses the original pretrained model on commonsense and general reasoning tasks, and achieves up to a 86.2% relative improvement over prior linearized attention methods on long-context benchmarks. Source Code is available at this https URL.

1 Introduction

Transformer-based Large Language Models (LLMs) [1, 2] have achieved remarkable performance across a wide range of language tasks, largely owing to self-attention's [3] ability to model long-range dependencies. However, vanilla self-attention requires computing pairwise interactions between all tokens, resulting in *quadratic* complexity in both time and memory with respect to sequence length. This cost becomes a primary bottleneck as contemporary models push toward substantially longer contexts. In response, linear attention [4, 5, 6] has emerged as a promising alternative, enabling a better efficiency-performance balance and allowing LLMs to be scaled to much longer sequences [7, 8, 9] with higher speed and lower memory.

However, building these models from scratch remains computationally exorbitant. Recent efforts, such as MiniMax-Text-01 [7] and Kimi-linear [8], attempt to offset the performance drop of pure linear attention by training ***inter-layer*** *hybrid* architectures from scratch, interleaving linear and softmax layers. Additionally, models like DeepSeek-V4 [10] adopt an intra-layer hybrid of compressed sparse and sliding window attention during pretraining to achieve acceleration. While effective, training

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

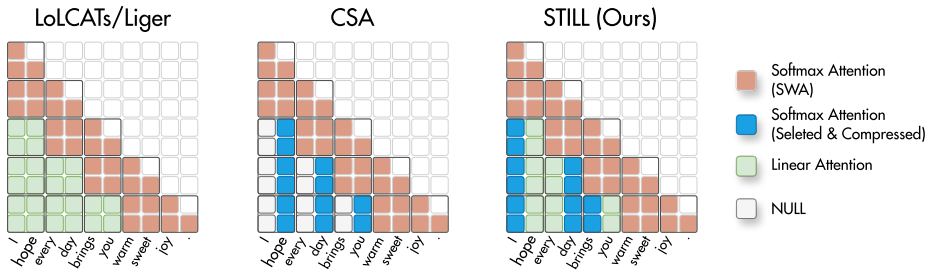


Figure 1: **Illustration of Intra-Layer Hybrid Attention.** While both act as intra-layer hybrids using local Softmax Attention (SA), they differ in handling the remaining sequence. LoLCATs/Liger applies uniform Linear Attention (LA), which limits long-context capacity, while compressed sparse attention (CSA) utilizes sparse compressed SA but completely discards unselected tokens (NULL). STILL selects tokens for high-fidelity SA and preserves the context via LA without information loss.

these foundation models from scratch requires thousands of GPU hours and massive compute clusters, making such architectures inaccessible to most researchers and practitioners. Furthermore, this inter-layer strategy still retains quadratic layers and fails to leverage the rich, full-attention features already established in existing open-source models.

To bypass the prohibitive costs of pretraining, recent work has shifted toward ***intra-layer*** *hybrid* attention designs that inherit checkpoints from *already-pretrained* LLMs. Approaches such as LoLCATs [11], Liger [12] and KL-LS [13] distill and fine-tune pretrained LLMs while replacing full softmax attention with intra-layer operator: softmax attention is computed over a fixed-length local window and linear attention summarizes the remaining context, yielding near-linear cost with relatively modest additional training. Most of these pipelines follow a similar recipe: start from a pretrained LLM, route the context tokens into SA branch and LA branch to encode the current token, train the learnable LA branch on a restricted subset of tokens, and finally finetune the model to align the hybrid operator with the original model behavior. This makes linearization feasible *without* rebuilding the model from scratch.

Despite the progress, this paradigm faces two key challenges. First, current token routing relies heavily on absolute position (e.g., a sliding window), implicitly assuming key dependencies are local (Fig. 1). For the remaining global context, approaches such as LoLCATs [11] and Liger [12] apply uniform linear attention, while Compressed Sparse Attention (CSA) [10] condenses tokens into block entries and restricts its attention to a top-k subset, leaving the unselected blocks unattended. Since crucial evidence often appears far outside this fixed window, pure position-based routing can inadvertently exclude vital tokens from high-fidelity softmax attention. Second, linearizing a pretrained Transformer must respect its distributional geometry, particularly the norm structure induced by softmax attention. Prior work [14, 15, 16] highlights that pretrained attention is "norm-aware": vector magnitudes modulate representational intensity and influence attention weights. However, many linearization schemes introduce learnable feature maps (often implemented with MLPs) that inadvertently *distort* these norms [17], so that even a nominally norm-aware kernel may operate on *miscalibrated* magnitudes. Together, these issues suggest that effective linearization requires (i) routing that reflects token importance, not just position, and (ii) kernels that preserve pretrained norm statistics rather than relearning them.

In this work, we propose STILL, a novel linearization framework that converts pretrained Transformers into efficient long-context models via in-layer hybrid attention, while explicitly addressing both challenges. STILL introduces Self-Saliency Score, which computes token importance within a local window to determine its relevance as a historical context. Notably, Self-Saliency Score exhibits high consistency between local and global contexts, allowing for reliable token selection using only a sliding-window mechanism. STILL introduces a Self-Saliency Score that estimates token importance using only local computation, yet is empirically consistent with global relevance. This enables reliable selection of a small set of salient tokens for softmax attention, while the remaining context is summarized with linear attention. In addition, to mitigate the norm distortion during linearization, we propose the Norm-Preserved Feature Map (NP-Map) that decouples direction from magnitude and reinjects pretrained norms, aligning the linear kernel with the model's representational intensity. Finally, to overcome the branching overhead and parallelization bottlenecks of naive token-wise routing, STILL introduces a delayed, chunk-level selection strategy. By making routing decisions

collectively per chunk, this hardware-friendly approach maximizes throughput, effectively translating theoretical linear complexity into tangible wall-clock acceleration during both training and inference.

We comprehensively evaluate STILL across both standalone intra-layer modules and intra-inter layer hybrid frameworks. Remarkably, STILL achieves linear-time scalability using less than 0.4B training tokens. This minimal cost translates to massive efficiency gains, yielding an 80% average memory reduction and an 83% prefilling speed-up while smoothly scaling sequences beyond 64K. STILL consistently outperforms existing baselines on reasoning tasks (up to +10.5% on MMLU [18]) and powerfully restores extended-context capabilities on RULER [19] and BABILong [20], recovering 86.2% of full-attention performance with a minimal softmax token budget. Crucially, STILL overcomes the severe performance collapse that plagues baselines at 16K contexts. While prior methods largely fail, our model maintains robust long-range memory, highlighted by a flawless score of 100 on the 16K S-NIAH-1 benchmark. These results show that self-saliency-based token selection and norm-preserved linear attention can largely recover the long-context capability lost in existing linearization methods, thereby enabling effective linearization of pretrained LLMs in long-context regimes.

2 Preliminaries

Linearizing Pretrained LLMs with Intra-Layer Hybrid Attention. Linearizing pretrained large language models (LLMs) aims to reduce the quadratic complexity of Softmax attention while preserving the representational capacity of pretrained weights. Intra-Layer Hybrid Attention provides a practical framework by integrating Softmax attention and Linear Attention within the same Transformer layer, where only a subset of tokens is processed using *Softmax Attention* (SA) and the remaining tokens are routed to *Linear Attention* (LA). Formally, the attention output at time step t is obtained by linearly combining the two branches:

$$\mathbf{y}_t = \alpha \mathbf{y}_t^{\mathrm{SA}} + \beta \mathbf{y}_t^{\mathrm{LA}}. \tag{1}$$

This design allows the model to retain high-fidelity attention for selected tokens while achieving linear-time complexity for the majority of the sequence. When linearizing a pretrained models, Linear Attention is typically initialized by inheriting existing attention weights, followed by lightweight alignment or fine-tuning to mitigate distribution mismatch. Existing Intra-Layer Hybrid Attention methods commonly adopt Sliding Window Attention (SWA) as a static, position-based routing strategy, where tokens within a fixed local window are processed using Softmax attention to preserve fine-grained local interactions, while tokens outside the window are handled by Linear Attention to capture broader contextual information.

Linear Attention with Learnable Feature Maps. Prior works [11, 21] introduce *learnable feature maps* that enable linear attention to approximate the exponential similarity of softmax attention given a pretrained weight from standard attention. Hedgehog [17], as a representative design, employs a lightweight MLP to transform the input and constructs a non-negative feature map by concatenating the softmax activations of the projected features and their negations. However, recent studies [15, 14, 16] have shown that the exponential function in softmax attention is highly sensitive to vector norms, where the sharpness of the attention distribution is partly controlled by the input magnitude. Consequently, the *norm-aware* property of softmax kernels is semantically valid only when applied to feature norms consistent with those of the pretrained model. However, we observe that in Hedgehog, the MLP transformation preceding the softmax mapping can freely rescale feature norms during fine-tuning, breaking the norm consistency before and after the MLP and causing the subsequent softmax to operate on a distorted norm distribution.

3 Methodology

In the following, we present the STILL model. We first propose the Self-Saliency Score for In-Layer Hybrid Linear Attention, which determines whether long-range dependencies should be modeled by softmax attention or linear attention, avoiding coarse sliding-window-based partitioning. Then, we present a Norm-preserved Feature Map that explicitly retains feature norm information, facilitating better alignment of linear attention with the inherited model. Finally, we present the complete STILL model with a chunk-wise parallel implementation that accelerates both training and inference.

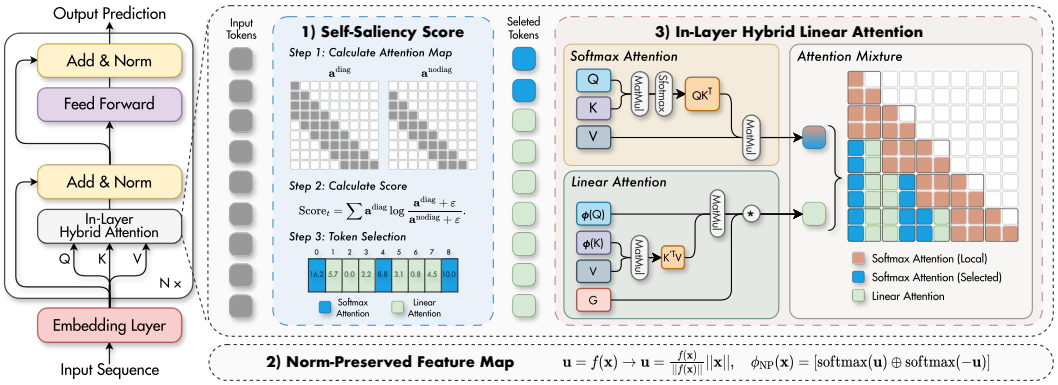


Figure 2: **Architecture of STILL for Intra-Layer Hybrid Linear Attention.** The diagram shows attention computation, chunk-level routing, and hybrid aggregation to outputs. STILL first computes the Self-Saliency Score within sliding-window attention for each token. Top-scoring tokens are routed to the softmax attention, while remaining tokens are processed via linear attention. Chunk-wise selecting replaces per-token selecting, enabling parallel training and inference.

3.1 Selecting Tokens by Self-Saliency Score

The core of in-layer hybrid attention lies in how to route the past context tokens into softmax attention (SA) and linear attention (LA) when computing the attention of the current token. Existing in-layer hybrid attention methods typically rely on sliding-window heuristics to route tokens to SA, which are inherently *position-based* and therefore fail to capture token-specific importance. To enable content-aware token selection, we introduce a *Self-Saliency Score*, which is computed for each token based on its attention weights within a local window. Formally, given an input sequence of length N, we denote the query and key representations for each head as $\{\mathbf{q}_t\}_{t=1}^{N} \in \mathbb{R}^{N\times d}$ and $\{\mathbf{k}_t\}_{t=1}^{N} \in \mathbb{R}^{N\times d}$, where $\mathbf{q}_t, \mathbf{k}_t \in \mathbb{R}^d$. We consider a sliding-window attention (SWA) with window radius w. For each query token $\mathbf{q}_t$, we define its local window index set as

$$\mathcal{W}_t = \{j \mid \max(1, t-w+1) \le j \le \min(N, t)\}. \tag{2}$$

For each token t, we measure the sensitivity of its local attention distribution to the self-attention term by comparing two SWA distributions: SWA with the diagonal term (Eq. (3)) is defined as,

$$\mathbf{a}_{t,j}^{\text{diag}} = \frac{\exp(\mathbf{q}_t \mathbf{k}_j^\top)}{\sum_{m\in\mathcal{W}_t} \exp(\mathbf{q}_t \mathbf{k}_m^\top)}, \quad j \in \mathcal{W}_t, \tag{3}$$

and SWA without the diagonal term Eq. (4) is defined as,

$$\mathbf{a}_{t,j}^{\text{nodiag}} = 0 \;\; (t=j), \qquad \mathbf{a}_{t,j}^{\text{nodiag}} = \frac{\exp(\mathbf{q}_t \mathbf{k}_j^\top)}{\sum_{m\in\mathcal{W}_t\setminus\{t\}} \exp(\mathbf{q}_t \mathbf{k}_m^\top)} \;\; (j \in \mathcal{W}_t \setminus \{t\}). \tag{4}$$

Finally, we leverage the two attention distributions to compute the **Self-Saliency Score** as

$$\text{Score}_t = \sum_{j\in\mathcal{W}_t} \mathbf{a}_{t,j}^{\text{diag}} \log \frac{\mathbf{a}_{t,j}^{\text{diag}} + \epsilon}{\mathbf{a}_{t,j}^{\text{nodiag}} + \epsilon}. \tag{5}$$

The Self-Saliency Score is then used to determine whether the token, when serving as a context token outside the local window, should be routed to the SA or LA branch, allowing the routing decision to adapt to token importance rather than solely rely on position. A higher Self-Saliency Score indicates that a token relies strongly on its self-attention term and is likely important under full-context attention. As shown in Fig. 2, Top-k tokens are routed to the SA branch, while the other tokens are handled by LA branch, enabling efficient and content-aware hybrid attention.

To evaluate our Self-Saliency Score, we compare score distribution under sliding-window (local) and full-context (global) attention. As shown in Fig. 3, the score distribution is clear: most tokens have near-zero scores, suitable for LA, while only a few reach high scores for SA processing. Importantly,

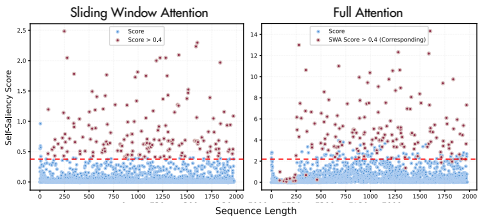


Figure 3: **Self-Saliency Score Distribution.** Red points mark top-scoring tokens routed to SA under sliding-window attention (left). Highlighting these exact tokens globally (right) demonstrates strong local-global consistency.

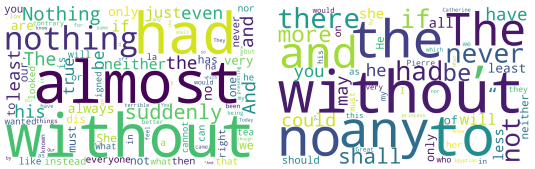


Figure 4: **Word Cloud of Tokens Most Frequently Selected for the SA Branch by Self-Saliency Score.** The selected set is dominated by *functional* operators (e.g., negation and quantification), suggesting that SA prioritizes precise, context-dependent interactions.

tokens identified as salient locally largely overlap with those selected under full-context attention, demonstrating that global importance can be reliably approximated using only local information. *More visualizations across different layers, heads, and window sizes are provided in Appendix A.5, all confirming this conclusion.* This consistency enables an efficient, content-aware routing strategy: token saliency is first estimated with sliding-window attention, after which only a small set of globally important tokens is routed to softmax attention, while the majority are handled efficiently by linear attention. To further justify our token routing strategy, we establish the following theoretical property:

Proposition 1 (Local-Global Consistency)**.** *Let p_{local} and p_{global} be the local and global self-attention probabilities. For $\epsilon \to 0^+$, the self-saliency score,* Score_t*, satisfies:*

1. ***Monotonicity**: $\frac{\partial \mathrm{Score}_t}{\partial p_{local}} > 0$ for all $p_{local} \in (0, 1)$.*
2. ***Rank Preservation**: Score_t is rank-preserving with respect to the expected global self-attention probabilities, i.e.:*

$$\mathrm{Score}_i > \mathrm{Score}_j \Rightarrow \mathbb{E}[p^{(i)}_{global}] > \mathbb{E}[p^{(j)}_{global}]. \tag{6}$$

Thus, local Top-λ selection based on Score_t is a probabilistically optimal proxy for global routing. Full proofs are in Appendix A.3.

For a qualitative view of what the score prioritizes, we visualize the most frequently selected SA tokens as a word cloud in Fig. 4; the prominence of **negation** (*not, no, never*), **modality** (*will, would, may*), and **quantification** (*all, any, nothing*) tokens supports our hypothesis that these high-scope operators particularly benefit from high-fidelity softmax attention.

3.2 Norm-Preserved Feature Map (NP-Map)

A critical yet often overlooked challenge in model linearization is the preservation of the norm-aware property intrinsic to the Softmax kernel. As established in NaLaFormer [14], the exponential function in standard attention naturally incorporates vector magnitudes into the attention score, allowing the model to distinguish tokens based on their representational intensity.

While Hedgehog [17] feature maps employ a softmax-based feature map to approximate this property, they typically introduce an additional MLP before the mapping to transform features. We observe that this MLP, while flexible, inevitably distorts the pre-trained norm distribution. Consequently, even though the subsequent softmax mapping is technically norm-aware, it operates on misaligned norms.

To address this limitation, we propose the **Norm-Preserved Feature Map** (NP-Map), a novel learnable feature map designed to enforce norm-invariance during the MLP transformation. Instead of allowing the MLP to freely scale the features, our approach explicitly decouples the feature's direction from its magnitude, recovering the original pre-trained norm while utilizing the MLP to only refine the directional components. Formally, for a given input $\mathbf{x} \in \{\mathbf{q}, \mathbf{k}\}$ derived from the pre-trained checkpoint, the NP-Map is defined as,

$$\mathbf{u} = f(\mathbf{x}) \to \mathbf{u} = \frac{f(\mathbf{x})}{\|f(\mathbf{x})\|}\|\mathbf{x}\|, \quad \phi_{\mathrm{NP}}(\mathbf{x}) = [\mathrm{softmax}(\mathbf{u}) \oplus \mathrm{softmax}(-\mathbf{u})], \tag{7}$$

where $f(\cdot)$ represents the learnable MLP as used in Hedgehog. By re-injecting the original norm $\|\mathbf{x}\|$ into the transformed feature $\mathbf{u}$, NP-Map ensures that the attention mechanism remains strictly

aligned with the pre-trained representational intensity. This allows the linear model to satisfy the non-negativity constraint while effectively inheriting the fine-grained semantic information from the pretrained Transformer. Furthermore, as gated attention [5, 22] effectively enhances the rank of linear attention while maintaining linear complexity with minimal computational overhead, we incorporate an additional gating mechanism. The complete linear attention computation is formulated as follows:

$$\mathbf{y}_t = \frac{\phi_{\text{NP}}(\mathbf{q}_t) \sum_{i=1}^{t} \phi_{\text{NP}}(\mathbf{k}_i)^\top \mathbf{v}_i}{\phi_{\text{NP}}(\mathbf{q}_t) \sum_{j=1}^{t} \phi_{\text{NP}}(\mathbf{k}_j)^\top} \odot \mathbf{g}_t. \tag{8}$$

3.3 STILL: Selecting Tokens for In-Layer Hybrid Linear Attention to Linearize LLMs

In this section, we propose **STILL**, an intra-layer hybrid linear attention method that combines self-saliency-based token selection with a norm-preserved learnable feature map. As illustrated in Fig. 2(a), STILL maintains a fixed-budget Softmax branch for recent and salient historical tokens, while the remaining historical context is summarized by linear attention. The same routing rule is used for recurrent decoding and chunk-wise parallel computation during training and prefilling.

Fixed-Budget Routing. To ensure bounded memory and linear complexity, STILL adopts a configuration-level fixed Softmax KV-cache budget m per attention head, which is independent of the runtime sequence length N. Among the m cache slots, W are reserved for the most recent tokens in the sliding window, while the remaining $m - W$ slots are allocated to salient historical tokens. Given a configured maximum context length $N_{\max}$ and chunk size C, we set

$$T_{\max} = \lceil N_{\max}/C \rceil, \qquad \lambda = \lfloor (m - W)/T_{\max} \rfloor, \tag{9}$$

where λ denotes the number of salient tokens retained from each completed chunk. For any sequence length $N \le N_{\max}$ with $T = \lceil N/C \rceil$ completed chunks, the number of historical salient tokens is bounded by $T\lambda \le T_{\max}\lambda \le m - W$.

Therefore, the total Softmax cache, including both the W local tokens and the historical salient tokens, never exceeds m tokens per head. All routing decisions are made independently per attention head, with each head selecting and retaining its own salient tokens; we omit head indices for brevity. Once selected, a salient token is retained permanently in the SA cache and is neither rescored nor evicted; unselected tokens are instead accumulated into the recurrent LA state.

Recurrent Form. In the decoding stage, STILL retain a small sliding-window attention for recent tokens to preserve local context. When a token moves out of the sliding window, we aim to the self-saliency score to decide whether it is routed to SA or LA. However, directly performing this routing decision at each decoding step would incur non-negligible computational overhead and low parallelism. To improve decoding efficiency, we therefore propose a delayed selection strategy, where token routing is performed at the granularity of chunks rather than at each time step.

We index the decoding step as $t = (a+1)T + b$, where T denotes the chunk size, a is the index of the second latest completed chunk, and b indicates the position of the current token within the ongoing chunk. At each decoding step, we compute the self-saliency score for the current token, as defined in Eq. (5). When $b = 1$, we rank the tokens in the most recently completed chunk a according to their S3 scores and select the top-λ tokens as *SA tokens*.

$$\mathbf{M}_a^{\mathcal{SA}} = \text{Top}_\lambda\big(\text{Score}_{a[1]:a[T]}\big), \quad \mathbf{M}_a^{\mathcal{LA}} = \{1, \ldots, T\} \setminus \mathbf{M}_a^{\mathcal{SA}}, \tag{10}$$

where $\mathbf{M}_a^{\mathcal{SA}}$ denotes the indices of the top-λ scores, and $\mathbf{M}_a^{\mathcal{LA}}$ denotes the complementary set of indices. Based on the selection results, tokens are divided into *salient* and *unsalient* ones. High-scoring and locally relevant key–value pairs are appended to the salient cache $\mathbf{h}_t$. The remaining tokens are written into the KV state $(\mathbf{s}_t, \mathbf{z}_t)$ and processed by the linear attention branch. Formally, we have:

$$\begin{aligned} \mathbf{h}_t = &\{(\mathbf{k}_i, \mathbf{v}_i) \mid \max(0, aT) < i \le \min(N, (a+1)T + b)\} \\ &\cup \begin{cases} \mathbf{h}_{t-1} \cup \left\{(\mathbf{k}_{a[i]}, \mathbf{v}_{a[i]}) \mid i \in \mathbf{M}_a^{\mathcal{SA}}\right\}, & a \ge 1, b = 1, \\ \varnothing, & a < 1, \\ \mathbf{h}_{t-1}, & o.w., \end{cases} \end{aligned} \tag{11}$$

$$\mathbf{s}_t = \begin{cases} \mathbf{s}_{t-1} + \sum_{i \in \mathbf{M}_a^{\mathcal{L}\text{-}\mathcal{A}}} (\phi_{\text{NP}}(\mathbf{k}_{a[i]}))^\top \mathbf{v}_{a[i]}, & a \geq 1, b = 1, \\ \mathbf{0}, & a < 1, \\ \mathbf{s}_{t-1}, & o.w., \end{cases} \tag{12}$$

$$\mathbf{z}_t = \begin{cases} \mathbf{z}_{t-1} + \sum_{i \in \mathbf{M}_a^{\mathcal{L}\text{-}\mathcal{A}}} (\phi_{\text{NP}}(\mathbf{k}_{a[i]}))^\top, & a \geq 1, b = 1, \\ \mathbf{0}, & a < 1, \\ \mathbf{z}_{t-1}, & o.w.. \end{cases} \tag{13}$$

Finally, computes the in-layer hybrid linear attention output token $\mathbf{y}_t$ as

$$\mathbf{y}_t = \frac{\mathcal{N}_{\text{SA}} + \mathcal{N}_{\text{LA}}}{\mathcal{D}_{\text{SA}} + \mathcal{D}_{\text{LA}}}, \tag{14}$$

$$\text{SA}: \begin{cases} \mathcal{N}_{\text{SA}} = \sum_{(\tilde{\mathbf{k}}, \tilde{\mathbf{v}}) \in \mathbf{h}_t} \exp(\mathbf{q}_t \tilde{\mathbf{k}}^\top) \tilde{\mathbf{v}}, \\ \mathcal{D}_{\text{SA}} = \sum_{(\tilde{\mathbf{k}}, \tilde{\mathbf{v}}) \in \mathbf{h}_t} \exp(\mathbf{q}_t \tilde{\mathbf{k}}^\top), \end{cases} \quad \text{LA}: \begin{cases} \mathcal{N}_{\text{LA}} = \phi_{\text{NP}}(\mathbf{q}_t) \mathbf{s}_t \odot \mathbf{g}_t, \\ \mathcal{D}_{\text{LA}} = \phi_{\text{NP}}(\mathbf{q}_t) \mathbf{z}_t. \end{cases} \tag{15}$$

In this way, STILL effectively routes appropriate tokens to softmax attention (SA) and linear attention (LA), leveraging the complementary strengths of both mechanisms within a unified computation.

Chunk-Wise Parallel Form. To improve parallelism during training and pre-filling while maintaining a unified training–inference formulation, our method can be implemented in a chunk-wise parallel manner. We divide the input sequence of length N into $T = \lceil N/C \rceil$ chunks, each containing C tokens. For each chunk, we first compute the self-saliency scores for all tokens and perform a local–global consistent selection within the chunk based on these scores. The selected tokens are then routed to softmax attention (SA), while the remaining tokens are assigned to linear attention (LA). Both SA and LA are computed efficiently in a chunk-wise parallel fashion, and their outputs are finally combined to produce the layer output. This chunk-wise formulation enables parallel computation within each chunk while preserving the same routing and memory-update behavior as in decoding. The overall procedure is summarized in Appendix A.4.

Complexity Analysis. The softmax attention branch requires $\mathcal{O}(Nmd)$ computations where m represents the salient tokens size. The linear attention branch processes the remaining tokens with a complexity of $\mathcal{O}((N-m)dd')$. Furthermore, the delayed selection strategy contributes $\mathcal{O}(T \log(C))$ to the total complexity. Summarizing these components, the overall complexity of STILL is given in Eq. (16), confirming that it scales linearly with the sequence length N.

$$\underbrace{\mathcal{O}(Nmd)}_{\text{SA}} + \underbrace{\mathcal{O}((N-m)dd')}_{\text{LA}} + \underbrace{\mathcal{O}(T \log(C))}_{\text{Selecting}} = \mathcal{O}(N) \quad \text{(Total)}. \tag{16}$$

4 Experiments

We evaluate STILL across three critical dimensions. We first assess its representational capacity on standard reasoning and challenging long-context tasks, including a demonstration of its compatibility within an intra-inter hybrid framework. Next, we quantify its hardware efficiency via memory and latency analyses. Finally, we validate its structural robustness across diverse backbone architectures, with detailed ablation studies deferred to Appendix A.7.

4.1 Experimental Setups

Training Strategy and Datasets. Following previous work [11], our training strategy employs a two-stage pipeline consisting of Attention Transfer and Low-rank Linearization, which including two stages. Stage 1 freezes the teacher and trains the learnable feature/gating projections in NP-Map via MSE. Stage 2 performs end-to-end LoRA [23] fine-tuning. Both stages are conducted on the Cleaned Alpaca dataset [24] for 20M tokens with a sequence length of 1024.

Evaluation Settings. We evaluate all models using the lm-evaluation-harness framework [25], following its standard evaluation protocols. All models are evaluated without task-specific finetuning. For each benchmark, we report the primary metric defined by the evaluation harness, such as accuracy or normalized accuracy, and ensure consistent evaluation settings across all compared methods.

Table 1: **Comparison of the Commonsense and General Reasoning Tasks**. Comparison of STILL with baseline methods under Llama 3 8B and Llama 3.1 8B models. STILL consistently demonstrates strong performance across benchmarks.

MODEL	TRAINING TOKENS	PIQA acc ↑	ARC-E acc ↑	ARC-C acc_n ↑	HELLA. acc_n ↑	WINO. acc ↑	MMLU (5-shot) ↑	AVG. (w. MMLU)	AVG. (wo. MMLU)
Transformer									
Llama 3 8B [2]	15000+	79.5	80.0	53.3	79.1	73.1	65.3	71.8	73.0
Llama 3.1 8B [2]	15000+	81.1	81.7	55.1	79.3	73.9	68.0	74.2	73.2
Subquadratic									
Mamba2 8B [32]	3500	79.8	75.9	48.1	77.1	71.6	48.7	67.0	70.6
Griffin 7B [33]	300	81.0	75.4	47.9	78.6	72.6	39.3	65.8	71.1
Falcon3-Mamba 7B [34]	7300	80.6	78.9	57.1	80.1	73.7	55.1	70.9	74.1
Hybird									
StripedHyena-Nous 7B [35]	-	78.8	77.2	40.0	76.4	66.4	26.0	60.8	67.8
Zamba 7B [36]	1000	81.4	74.5	46.6	80.2	76.4	57.7	69.5	71.8
RecurrentGemma 9B [37]	2000	80.6	78.9	57.1	80.1	73.7	55.1	70.9	74.1
Linearized from Llama 3 8B									
T2R-LoLCATs [21]	0.04	62.0	42.1	24.7	32.7	48.3	23.2	38.8	42.0
Hedgehog-LoLCATs [17]	0.04	77.4	71.1	40.6	50.7	54.3	24.2	53.1	58.8
SUPRA [30]	20	78.9	75.1	46.5	71.7	65.8	40.9	63.2	67.6
LoLCATs [11]	0.04	80.0	78.8	52.2	75.6	**73.3**	43.9	67.3	72.0
Liger-GLA [12]	0.02	80.3	81.1	52.5	76.3	72.0	43.4	67.6	72.4
STILL (Ours)	0.04	**81.3**	**81.7**	**54.8**	**78.1**	73.2	**59.7**	**71.5**	**73.8**
Linearized from Llama 3.1 8B									
Llamba [31]	12	80.9	82.5	54.6	77.6	73.3	60.0	71.5	73.8
LoLCATs [11]	0.04	80.9	82.6	54.7	76.6	**73.6**	50.8	69.9	73.7
Liger-GLA [12]	0.02	80.5	81.8	55.6	75.4	69.5	46.9	68.3	72.6
STILL (Ours)	0.04	**81.3**	**83.0**	**56.7**	**79.0**	73.4	**61.3**	**72.5**	**74.7**

4.2 Commonsense and General Reasoning Tasks

Settings. Settings. We evaluate STILL's commonsense and general reasoning on PIQA [26], ARC-E/C [27], HellaSwag [28], WinoGrande [29], and MMLU [18] under standard zero-shot settings (five-shot for MMLU) [30, 11, 12]. Using Llama 3 and 3.1 8B [31] as backbones, we compare against original Transformers, from-scratch subquadratic, inter-layer hybrid, and linearized baselines.

Results. Tab. 1 shows STILL achieves the best average performance (with and without MMLU), consistently outperforming linearized baselines. Notably, under the Llama 3.1 8B backbone, STILL improves MMLU accuracy over Liger [12] from 46.9 to 61.3, strongly preserving the backbone's reasoning capabilities, and impressively surpasses the original full-attention model's average performance by 1.5. Extended baseline comparisons and results across various pretrained LLM backbones are provided in Appendix Tab. 8 and Tab. 15, respectively.

4.3 Long-Context Benchmarks

To evaluate the capability of STILL in handling extended dependencies, we conduct experiments on two challenging long-context tasks, RULER [19] and BABILong [20]. These benchmarks assess the model's ability to maintain performance as the sequence length increases.

4.3.1 RULER Benchmark

We firstly evaluate STILL's long-context retrieval and memory capabilities on the RULER benchmark, covering both *Single Needle-in-a-Haystack (S-NIAH)* retrieval tasks and the *extended RULER* benchmark that requires stronger memory and state tracking.

S-NIAH Settings. We evaluate S-NIAH [19] up to 4K context lengths using the Llama 3.1 8B as the pretrained backbone. We compare STILL against sliding window attention (SWA) only, LoLCATs [11] and Liger [12] (window sizes: 128, 256, 512). To match baseline cached tokens, STILL keeps the total cache (selected tokens + sliding-window) at the baseline window size.

S-NIAH Results. Tab. 2 shows STILL consistently outperforms SWA and linearized baselines under equal or smaller cache budgets, with the gap widening at longer contexts where these baselines largely fail. Notably, at 4K context length, STILL achieves 86.2% on S-NIAH, whereas LoLCATs and Liger largely fail. Extended model configurations and comprehensive results for the NIAH task are detailed in Appendix Tab. 16.

Full RULER Settings. We evaluate STILL on the extended RULER benchmark [19] at a 4K context length across eight sub-tasks requiring stronger memory and state tracking. Using the Llama 3.1 8B backbone, we compare against Mamba2 [32], LoLCATs [11], and Liger [12], with the latter two using an 896 window size. STILL operates with a 128 window size and a strict 896 total token cap.

Table 2: **Comparison of the Single Needle-in-a-Haystack (S-NIAH) Tasks.** Accuracy across 0.5K–4K contexts under the same cache budgets.

S-NIAH	0.5K	1K	2K	4K
SWA	100	25.2	14.6	5.8
LoLCATs [11]	100	65.6	24.6	8.8
Liger-GLA [12]	73.4	1.0	0.0	0.0
STILL (Ours)	**100**	**99.6**	**89.0**	**86.2**
—Improved	+0.0	+34.0	+64.4	+77.4

Table 3: **Full RULER Benchmark Results.** Comprehensive performance comparison at a 4K context length, evaluating long-range memory and state tracking abilities across multiple tasks.

Model	MK	MQ	MV	CWE	FWE	HQA	SQA	VT	Avg.
Mamba2 [32]	19.9	49.1	35.0	32.9	**76.6**	31.8	35.5	**76.5**	44.7
LoLCATs [11]	4.9	3.3	3.6	3.0	13.6	14.2	14.0	0.7	7.2
Liger-GLA [12]	0	0	0	0.2	0.7	6.4	9.1	0	2.1
STILL (Ours)	**21.5**	**52.2**	**51.3**	**36.7**	76.3	**32.2**	**49.6**	63.7	**47.9**

Full RULER Results. Tab. 3 shows that STILL achieves the highest overall average score of 47.9, exceeding LoLCATs and Liger by a large margin. These tasks typically require understanding the entire context, indicating that STILL exhibits stronger long-context memory and state-tracking capabilities under tight computational constraints.

4.3.2 BABILong Benchmark

Settings. We evaluate long-context reasoning on BABILong [20], which extends bAbI [38] QA by embedding facts within long irrelevant contexts to test distant evidence integration. Using the Llama 3.1 8B backbone, we compare STILL against SWA and LoLCATs across tasks QA1–QA5 under a strict 256-token total cache budget.

Results. Tab. 4 shows that STILL consistently achieves the best accuracy across all context lengths. At 4K, STILL improves QA1 from 3 for LoLCATs and 1 for SWA to 10, and improves QA5 from 6 for both LoLCATs and SWA to 24, demonstrating stronger long-context reasoning and evidence aggregation. Full results are provided in Appendix Tab. 17.

Table 4: **BABILong Benchmark Results.** Performance comparison on BABILong across increasing context lengths (0K∼4K), evaluating long-context reasoning performance.

Model	QA1				QA4				QA5			
	0K	1K	2K	4K	0K	1K	2K	4K	0K	1K	2K	4K
SWA	93	2	1	1	74	0	0	0	68	18	11	6
LoLCATs [11]	100	22	5	3	65	21	9	1	62	42	17	6
STILL (Ours)	**100**$_{+0}$	**45**$_{+23}$	**22**$_{+17}$	**10**$_{+7}$	**82**$_{+8}$	**59**$_{+28}$	**30**$_{+21}$	**9**$_{+8}$	**83**$_{+15}$	**77**$_{+35}$	**45**$_{+28}$	**24**$_{+18}$

4.4 Architectural Compatibility Analysis: Intra-Inter Layer Hybrid Strategy

Results. As shown in Tab. 5, STILL demonstrates absolute superiority on the RULER benchmark. STILL perfectly solves S-NIAH-1 up to 16K with a score of 100, and shows remarkable robustness on S-NIAH-2 at 16K, surpassing LoLCATs by 65.4. Consequently, STILL yields a RULER score of 27.20 at 16K, outperforming Zebra-Llama [39] by 13.83, demonstrating its strong architectural compatibility within intra-layer hybrid frameworks.

Table 5: **Intra-Inter Layer Hybrid Results on the RULER Benchmark.** All models are linearized from Llama 3.2 and evaluated over context lengths ranging from 4K to 16K.

Model	S-NIAH-1			S-NIAH-2			S-NIAH-3			RULER		
	4K	8K	16K	4K	8K	16K	4K	8K	16K	4K	8K	16K
MambaInLlama [40]	-	-	-	-	-	-	-	-	-	38.75	21.55	3.88
Zebra-Llama [39]	-	-	-	-	-	-	-	-	-	35.75	24.80	13.37
SWA	9.4	8.2	9.0	16.4	10.2	10.6	11.4	4.4	7.2	6.00	3.96	4.66
LoLCATs [11]	84.8	81.6	36.2	86.8	33.0	8.6	54.6	16.8	3.4	33.99	17.93	7.59
STILL (Ours)	**100**	**100**	**100**	**100**	**99.0**	**74.0**	**70.4**	**67.2**	**25.8**	**53.24**	**35.66**	**27.20**

4.5 Ablation of Token-Routing Metrics

Because Self-Saliency is related to diagonal self-attention, we ask whether diagonal attention alone is sufficient for routing. We therefore compare it with simpler selection criteria under the same routing and cache settings, covering both short-context reasoning and long-context retrieval.

Short-context reasoning. Table 10 reports the average performance over PIQA, ARC-E, ARC-C, HellaSwag, WinoGrande, and MMLU. Replacing Self-Saliency with the diagonal self-attention weight reduces the average score only slightly by 0.5% This modest gap is expected because these benchmarks contain relatively short contexts, where retaining sparse long-range evidence is not the primary bottleneck. Thus, diagonal attention already captures much of the dominant routing trend.

Table 6: Routing-metric ablation on short-context reasoning. Δ denotes the difference from Self-Saliency.

Routing Metric	Avg. ↑	Δ
Self-Saliency	**72.5**	–
Diagonal attention	72.0	-0.5

Table 7: Routing-metric ablation on long-context S-NIAH-1 retrieval under the same 128 cache budget.

Routing Metric	0.5K ↑	1K ↑	Avg. ↑
Self-Saliency	**63.2**	**22.0**	**42.6**
Diagonal attention	$35.6_{-27.6}$	$10.6_{-11.4}$	23.1
$\mathbf{q_t}^\top \mathbf{k_t}$	$35.4_{-27.8}$	$10.2_{-11.8}$	22.8
Key norm	$18.6_{-44.6}$	$7.1_{-14.9}$	12.9
Uniform per chunk	$17.8_{-45.4}$	$6.8_{-15.2}$	12.3
Random selection	$18.6_{-44.6}$	$7.0_{-15.0}$	12.8

Long-context retrieval. The difference becomes substantially larger when routing is the key bottleneck. Table 11 compares Self-Saliency with diagonal attention, the query–key inner product, key norm, uniform per-chunk selection, and random selection on S-NIAH-1 under the same strict 128-token cache budget. Self-Saliency achieves 63.2 and 22.0 accuracy at 0.5K and 1K contexts, respectively, yielding an average of 42.6. In comparison, diagonal attention and $q_t^\top k_t$ obtain averages of only 23.1 and 22.8, while key-norm, uniform, and random routing remain around 12–13. These results show that although diagonal attention provides a reasonable proxy in ordinary short-context settings, it is not sufficient when accurate sparse retention of remote evidence is required. Self-Saliency is therefore particularly beneficial in the routing-sensitive long-context regime.

4.6 Efficiency Comparison

To assess the efficiency of STILL, we measure peak GPU memory usage and end-to-end latency on a single NVIDIA H200 GPU, varying the sequence length from 1K to 64K. Results demonstrate that STILL achieves a favorable memory–speed trade-off.

Prefilling. Fig. 5 illustrates prefilling efficiency. Softmax attention hits memory bottlenecks beyond 16K tokens, whereas STILL scales smoothly to 64K, requiring only 17% of the Softmax runtime at 16K. Notably, our delayed selection strategy ensures STILL adds only marginal latency and memory overheads over LoLCATs, a minimal trade-off for its massive recovery of long-context accuracy.

Decoding. Fig. 6 shows decoding efficiency. While Softmax scales poorly and OOMs beyond 32K tokens, STILL and LoLCATs maintain near-constant memory up to 64K, consuming a mere 20% of the Softmax memory at 16K. Although STILL exhibits slightly higher latency than LoLCATs, it achieves stable throughput with a flat memory profile, proving its practicality for extended generation.

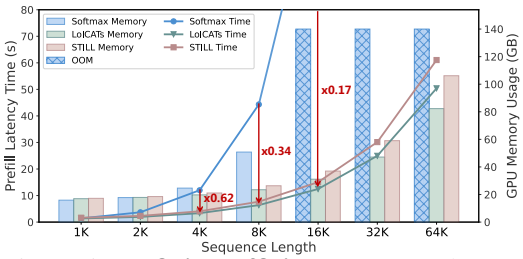


Figure 5: **Prefilling Efficiency.** STILL demonstrates favorable prefill latency and memory consumption from 1K to 64K.

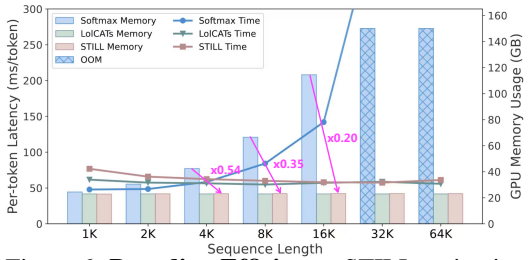


Figure 6: **Decoding Efficiency.** STILL maintains constant GPU memory while achieving lower latency for long-context decoding.

5 Conclusion

In this paper, we propose STILL, an intra-layer hybrid attention framework for efficiently linearizing pretrained LLMs. STILL leverages a self-saliency score to route the context tokens into softmax attention and linear attention. To preserve pretrained representational geometry, we introduce a norm-preserved learnable feature map that explicitly maintains norm consistency during linearization, and adopt a unified chunk-wise and delayed selection formulation to improve efficiency and parallelism. Together, these designs enable linear-time attention while largely retaining the reasoning and long-context capabilities of the original pretrained models. Through comprehensive evaluations across both intra-layer and intra-inter hybrid architectures, we demonstrate that STILL not only recovers long-context capabilities and improves short-sequence performance, but also delivers substantial memory savings and acceleration for both prefill and decoding.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant Nos. 62372132 and 62606255, and in part by the Key Research Project of Pengcheng Laboratory under Grant No. PCL2025A14.

References

[1] Tom B. Brown, Benjamin Mann, Nick Ryder, et al. Language models are few-shot learners. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[2] Llama Team. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pages 5998–6008, 2017.

[4] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 5156–5165, 2020.

[5] Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.

[6] Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[7] Aonian Li, Bangwei Gong, Bo Yang, et al. Minimax-01: Scaling foundation models with lightning attention. *CoRR*, abs/2501.08313, 2025.

[8] Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, et al. Kimi linear: An expressive, efficient attention architecture. *CoRR*, abs/2510.26692, 2025.

[9] An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report. *CoRR*, abs/2505.09388, 2025.

[10] DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026.

[11] Michael Zhang, Simran Arora, Rahul Chalamala, Benjamin Frederick Spector, Alan Wu, Krithik Ramesh, Aaryan Singhal, and Christopher Ré. Lolcats: On low-rank linearizing of large language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

[12] Disen Lan, Weigao Sun, Jiaxi Hu, Jusen Du, and Yu Cheng. Liger: Linearizing large language models to gated recurrent structures. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2025.

[13] Yanhong Li, Songlin Yang, Shawn Tan, Mayank Mishra, Rameswar Panda, Jiawei Zhou, and Yoon Kim. Distilling to hybrid attention models via kl-guided layer selection. *CoRR*, abs/2512.20569, 2025.

[14] Weikang Meng, Yadan Luo, Liangyu Huo, Yingjian Li, Yaowei Wang, Xin Li, and Zheng Zhang. Norm$\times$direction: Restoring the missing query norm in vision linear attention. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2026.

[15] Dongchen Han, Yifan Pu, Zhuofan Xia, Yizeng Han, Xuran Pan, Xiu Li, Jiwen Lu, Shiji Song, and Gao Huang. Bridging the divide: Reconsidering softmax and linear attention. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[16] Qihang Fan, Huaibo Huang, Yuang Ai, and Ran He. Rectifying magnitude neglect in linear attention. *CoRR*, abs/2507.00698, 2025.

[17] Michael Zhang, Kush Bhatia, Hermann Kumbong, and Christopher Ré. The hedgehog & the porcupine: Expressive linear attentions with softmax mimicry. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

[18] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[19] Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: what's the real context size of your long-context language models? *CoRR*, abs/2404.06654, 2024.

[20] Yuri Kuratov, Aydar Bulatov, Petr Anokhin, et al. Babilong: Testing the limits of llms with long context reasoning-in-a-haystack. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[21] Jungo Kasai, Hao Peng, Yizhe Zhang, et al. Finetuning pretrained transformers into rnns. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 10630–10643, 2021.

[22] Zihan Qiu, Zekun Wang, Bo Zheng, et al. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free. *CoRR*, abs/2505.06708, 2025.

[23] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.

[24] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model, 2023.

[25] Gao Leo, Tow Jonathan, Abbasi Baber, et al. The language model evaluation harness, 2025.

[26] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 7432–7439, 2020.

[27] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/1803.05457, 2018.

[28] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4791–4800, 2019.

[29] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 8732–8740, 2020.

[30] Jean Mercat, Igor Vasiljevic, Sedrick Keh, Kushal Arora, Achal Dave, Adrien Gaidon, and Thomas Kollar. Linearizing large language models. *CoRR*, abs/2405.06640, 2024.

[31] Aviv Bick, Tobias Katsch, Nimit Sharad Sohoni, Arjun Desai, and Albert Gu. Llamba: Scaling distilled recurrent models for efficient language processing. *CoRR*, abs/2502.14458, 2025.

[32] Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.

[33] Soham De, Samuel L. Smith, Anushan Fernando, et al. Griffin: Mixing gated linear recurrences with local attention for efficient language models. *CoRR*, abs/2402.19427, 2024.

[34] Jingwei Zuo, Maksim Velikanov, Dhia Eddine Rhaiem, Ilyas Chahed, Younes Belkada, Guillaume Kunsch, and Hakim Hacid. Falcon mamba: The first competitive attention-free 7b language model. *CoRR*, abs/2410.05355, 2024.

[35] Poli Michael, Wang Jue, Massaroli Stefano, Quesnelle Jeffrey, Carlow Ryan, Nguyen Eric, and Thomas Armin. Stripedhyena: Moving beyond transformers with hybrid signal processing models, 2023.

[36] Paolo Glorioso, Quentin Anthony, Yury Tokpanov, James Whittington, Jonathan Pilault, Adam Ibrahim, and Beren Millidge. Zamba: A compact 7b SSM hybrid model. *CoRR*, abs/2405.16712, 2024.

[37] Aleksandar Botev, Soham De, et al. Recurrentgemma: Moving past transformers for efficient open language models. *CoRR*, abs/2404.07839, 2024.

[38] Jason Weston, Antoine Bordes, Sumit Chopra, and Tomás Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[39] Mingyu Yang, Mehdi Rezagholizadeh, Guihong Li, Vikram Appia, and Emad Barsoum. Zebra-llama: Towards extremely efficient hybrid models. *CoRR*, abs/2505.17272, 2025.

[40] Junxiong Wang, Daniele Paliotta, Avner May, Alexander M. Rush, and Tri Dao. The mamba in the llama: Distilling and accelerating hybrid models. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[41] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, et al. Mistral 7b. *CoRR*, abs/2310.06825, 2023.

[42] Zhen Qin, Weixuan Sun, Hui Deng, Dongxu Li, Yunshen Wei, Baohong Lv, Junjie Yan, Lingpeng Kong, and Yiran Zhong. cosformer: Rethinking softmax in attention. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.

[43] Han Cai, Junyan Li, Muyan Hu, Chuang Gan, and Song Han. Efficientvit: Lightweight multi-scale attention for high-resolution dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 17256–17267, 2023.

[44] Dongchen Han, Xuran Pan, Yizeng Han, Shiji Song, and Gao Huang. Flatten transformer: Vision transformer using focused linear attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5938–5948, 2023.

[45] Weikang Meng, Yadan Luo, Xin Li, Dongmei Jiang, and Zheng Zhang. Polaformer: Polarity-aware linear attention for vision transformers. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

[46] Weikang Meng, Yadan Luo, Liangyu Huo, Yingjian Li, Yaowei Wang, Zheng Zhang, and Heng Tao Shen. Polaformer++: Polarity-aware linear attention with channel-wise spikiness. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2026.

[47] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *CoRR*, abs/2307.08621, 2023.

[48] Zhen Qin, Dong Li, Weigao Sun, Weixuan Sun, Xuyang Shen, Xiaodong Han, Yunshen Wei, Baohong Lv, Xiao Luo, Yu Qiao, and Yiran Zhong. Transnormerllm: A faster and better large language model with improved transnormer. *CoRR*, abs/2307.14995, 2023.

[49] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark W. Barrett, Zhangyang Wang, and Beidi Chen. H2O: heavy-hitter oracle for efficient generative inference of large language models. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[50] Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: LLM knows what you are looking for before generation. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[51] Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. QUEST: query-aware sparsity for efficient long-context LLM inference. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 235, pages 47901–47911, 2024.

[52] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

[53] Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, and Wen Xiao. Pyramidkv: Dynamic KV cache compression based on pyramidal information funneling. *CoRR*, abs/2406.02069, 2024.

[54] Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. Duoattention: Efficient long-context LLM inference with retrieval and streaming heads. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

[55] Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[56] Yingfa Chen, Zhen Leng Thai, Zihan Zhou, Zhu Zhang, Xingyu Shen, Shuo Wang, Chaojun Xiao, Xu Han, and Zhiyuan Liu. Hybrid linear attention done right: Efficient distillation and effective architectures for extremely long contexts. *CoRR*, abs/2601.22156, 2026.

[57] Yuxian Gu, Qinghao Hu, Haocheng Xi, Junyu Chen, Shang Yang, Song Han, and Han Cai. Jet-nemotron: Efficient language model with post neural architecture search. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[58] Daniel Goldstein, Eric Alcaide, Janna Lu, and Eugene Cheah. RADLADS: rapid attention distillation to linear attention decoders at scale. *CoRR*, abs/2505.03005, 2025.

[59] Difan Deng, Andreas Bentzen Winje, Lukas Fehring, and Marius Lindauer. Neural attention search linear: Towards adaptive token-level hybrid attention models. *CoRR*, abs/2602.03681, 2026.

[60] Jeffrey Li, Alex Fang, Georgios Smyrnis, et al. Datacomp-lm: In search of the next generation of training sets for language models. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[61] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *CoRR*, abs/2312.00752, 2023.

[62] Bo Peng, Daniel Goldstein, Quentin Anthony, Alon Albalak, et al. Eagle and finch: RWKV with matrix-valued states and dynamic recurrence. *CoRR*, abs/2404.05892, 2024.

[63] Cong Jiang, Xiaofeng Zhang, Fangzhi Zhu, XiaoWei Chen, Junxiong Zhu, and Zheng Zhang. Rethinking LLM reasoning: From explicit trajectories to latent representations. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026.

[64] Shibo Hao, Sainbayar Sukhbaatar, et al. Training large language models to reason in a continuous latent space. volume abs/2412.06769, 2024.

[65] Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

[66] Cong Jiang, Fangzhi Zhu, Xiaowei Chen, Junxiong Zhu, Bo Zheng, Yifeng Wang, and Zheng Zhang. Dmlora: Dynamic multi-subspace low-rank adaptation. In *Companion Proceedings of the ACM Web Conference (WWW)*, pages 1038–1042, 2025.

[67] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *Advances in neural information processing systems*, 35:10078–10093, 2022.

[68] Kunchang Li, Yali Wang, Yizhuo Li, Yi Wang, Yinan He, Limin Wang, and Yu Qiao. Unmasked teacher: Towards training-efficient video foundation models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19948–19960, October 2023.

[69] Weijun Zhuang, Yuqing Huang, Weikang Meng, Xin Li, Ming Liu, Xiaopeng Hong, Yaowei Wang, and Wangmeng Zuo. Cluster-wise spatio-temporal masking for efficient video-language pretraining. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 39197–39207, June 2026.

A Appendix

- A.1 **Limitations.**
- A.2 **Related Work.**
- A.3 **Theoretical Proof.** The proof of theorem.
- A.4 **Algorithm.** Chunk-wise parallel form with delayed selection strategy.
- A.5 **More Visualizations.** More visualizations of Self-Saliency Score Distribution
- A.6 **Experimental Setups.**
- A.7 **Ablation Study.** Ablation of components and different backbone models and scales.
- A.8 **Full Tables.** Full tables about the experimental results.
- A.9 **Future Work.**

A.1 Limitations

While STILL demonstrates strong performance in linearizing LLMs, we acknowledge a few practical constraints. First, due to computational limits, our empirical evaluations focus on models up to the 8B parameter scale (e.g., Llama 3/3.1 8B, Mistral 7B). Although our theoretical derivations for the intra-layer hybrid routing are scale-agnostic, verifying transferability to massive models (e.g., 70B+) remains future work. Second, STILL is currently tailored for text-based autoregressive models; extending it to multimodal architectures is an open direction. Finally, while our fixed chunk and window configurations yield highly competitive results, exploring dynamic, task-adaptive window sizing could further optimize hardware efficiency.

A.2 Related Work

Linear Attention. As the central component of Transformer [3], self-attention effectively captures global dependencies among tokens. In decoder-only LLMs [2, 41], for each head, the current query $\mathbf{q}_t$ attends to all past key–value (KV) pairs, denoted as $\{\mathbf{k}_i\}_{i=1}^{t} \in \mathbb{R}^{t\times d}$ and $\{\mathbf{v}_i\}_{i=1}^{t} \in \mathbb{R}^{t\times d}$. The attention scores are computed via dot product, normalized using softmax under causal masking, and used to aggregate the values. Formally, the output $\mathbf{y}_t$ is given by

$$\mathbf{y}_t = \frac{\sum_{i=1}^{t} \exp(\mathbf{q}_t \mathbf{k}_i^\top / \sqrt{d}) \mathbf{v}_i}{\sum_{j=1}^{t} \exp\left(\mathbf{q}_t \mathbf{k}_j^\top / \sqrt{d}\right)}. \tag{17}$$

While softmax attention provides a powerful mechanism to model long-range dependencies, its computation requires evaluating all pairwise interactions between queries and keys, resulting in $\mathcal{O}(N^2 d)$ complexity. This quadratic cost becomes a significant bottleneck for both training and inference on long sequences.

To mitigate the efficiency bottlenecks of quadratic softmax attention, Linear Attention [4] is proposed to approximate the softmax via a designed feature map $\phi(\cdot) : \mathbb{R}^d \to \mathbb{R}^{d'}$ such that $\exp(\mathbf{q}\mathbf{k}^\top) \approx \phi(\mathbf{q})\phi(\mathbf{k})^\top$. As shown in Equation (18), by prioritizing the computation of the KV pairs, this approach reduces the complexity from $\mathcal{O}(N^2 d)$ to $\mathcal{O}(N d'^2)$.

$$\mathbf{y}_t = \frac{\sum_{i=1}^{t} \phi(\mathbf{q}_t)\phi(\mathbf{k}_i)^\top \mathbf{v}_i}{\sum_{j=1}^{t} \phi(\mathbf{q}_t)\phi(\mathbf{k}_j)^\top} = \frac{\phi(\mathbf{q}_t) \sum_{i=1}^{t} \phi(\mathbf{k}_i)^\top \mathbf{v}_i}{\phi(\mathbf{q}_t) \sum_{j=1}^{t} \phi(\mathbf{k}_j)^\top} = \frac{\phi(\mathbf{q}_t)\mathbf{s}_t}{\phi(\mathbf{q}_t)\mathbf{z}_t}, \tag{18}$$

$$\mathbf{s}_t = \mathbf{s}_{t-1} + \phi(\mathbf{k}_t)^\top \mathbf{v}_t, \quad \mathbf{z}_t = \mathbf{z}_{t-1} + \phi(\mathbf{k}_t)^\top. \tag{19}$$

The design of kernel functions generally considers two key factors: ensuring *non-negative feature maps*, e.g., through $\mathrm{ReLU}(\cdot)$ [42, 43] or $1 + \mathrm{ELU}(\cdot)$ [4], and promoting a *spikiness property* to sharpen the attention distribution. Existing methods such as Flatten Transformer [44], PolaFormer [45, 46], and NaLaFormer [14] address the latter by designing power-based feature maps that produce low-entropy, softmax-like attention weights, enhancing the selectivity of attention. In addition, Hedgehog [17] uses a learnable feature map trained with a loss to approximate the exponential function, producing attention distributions closer to softmax.

Beyond the aforementioned designs targeting attention similarity, in autoregressive architectures such as LLMs, linear attention can be reformulated as a recurrent computation over the KV state as $\mathbf{s}_t = \mathbf{k}_t^\top \mathbf{v}_t$, making the state update strategy crucial (shown in Equation (19)). RetNet [47] and TransnormerLLM [48] use a decay factor to mitigate attention dilution. GLA [5] introduces a gating mechanism on the KV state to selectively retain or forget information, while DeltaNet [6] leverages a delta-rule update to dynamically adjust key–value associations for more effective long-context modeling. However, these methods cannot reuse pre-trained Transformer weights, requiring costly retraining from scratch and limiting the use of existing large-scale models.

Linearized Transformer. Recently, several studies have explored leveraging existing pre-trained large language models (LLMs) to obtain linearized LLMs through low-cost training approaches. These methods [21, 30, 31] inherit pre-trained weights and linearize quadratic LLMs through uptraining, distillation, or fine-tuning. However, they still require large token budgets and high memory during adaptation. Consequently, some studies focus on achieving Transformer linearization with lower memory usage and fewer tokens. LoLCATs [11] linearizes pre-trained LLMs using only 40M tokens via kernel distillation and low-rank fine-tuning. Liger [12] further reuses the key projection weights to construct gating, avoiding the introduction of additional parameters, and achieves linearization with only low-rank fine-tuning. While these methods employ sliding-window attention to preserve local information sensitivity, they rely exclusively on linear attention for long-range modeling. This reliance significantly impairs the representation of long-range dependencies, leading to a pronounced degradation in performance on long-sequence tasks. Therefore, a method is needed to properly allocate dependencies between softmax attention and linear attention, thereby restoring performance on long-sequence tasks.

Token Selection and Sparse Attention Inference. Another line of research improves long-context inference by selectively retaining or retrieving salient tokens from the KV cache of pretrained softmax-attention models. Representative methods exploit accumulated attention statistics [49], observation-window attention and clustering [50], query-aware block retrieval [51], attention sinks [52], layer-wise budget allocation [53], or head-level functional specialization [54]; related work further sparsifies attention computation during prefilling [55]. These approaches primarily reduce the inference cost of an otherwise unchanged quadratic attention model by pruning, compressing, or selectively accessing its KV cache. In contrast, STILL targets *model linearization*: it converts pretrained softmax attention into a trainable intra-layer hybrid operator, where salient tokens retain high-fidelity softmax attention while the remaining context is continuously summarized by a recurrent linear-attention state. Moreover, Self-Saliency is computed causally from local attention, making the same routing mechanism applicable to both prefilling and autoregressive decoding.

Recent Hybrid Distillation Advances. Beyond intra-layer designs, recent work improves Transformer-to-hybrid distillation along orthogonal axes: layer-level softmax budget allocation [56], post-hoc architecture search over converted blocks [57], and rapid attention distillation pipelines [58]. Concurrent to our work, NAtS-L [59] routes chunks between softmax and gated delta-rule attention. STILL is complementary to these advances: our self-saliency routing and norm-preserved feature map operate *within* each converted layer and can be composed with layer-selection strategies.

A.3 Theoretical Proof

In this section, we provide the theoretical derivation demonstrating that the Self-Saliency Score effectively tracks the global self-attention probability and exhibits strong local-global consistency. This justifies our strategy of using local sliding-window computations to efficiently route tokens for global attention.

Lemma 1 (Monotonicity of the Saliency Score)**.** *Given an infinitesimally small smoothing parameter $\epsilon > 0$, the Self-Saliency Score Score_t is a strictly monotonically increasing function of the local self-attention probability p_{local}.*

Proof. The Self-Saliency Score measures the divergence between the local attention distributions with and without the current token's self-similarity. Let Z_t be the denominator of the softmax function for the local window, and $Z_{t\setminus t}$ be the denominator excluding the self-interaction term, such that:

$$Z_t = Z_{t\setminus t} + \exp(q_t k_t^\top) \tag{20}$$

By definition, the local self-attention probability is $p_{local} = a_{t,t}^{diag} = \frac{\exp(q_t k_t^\top)}{Z_t}$. The total probability mass allocated to the other context tokens is:

$$1 - p_{local} = \frac{Z_{t\setminus t}}{Z_t} \tag{21}$$

For any other context token ($j \neq t$), we can express its attention probability including the self-interaction term ($a_{t,j}^{diag}$) in terms of the probability without it ($a_{t,j}^{nodiag}$):

$$a_{t,j}^{diag} = \left(\frac{\exp(q_t k_j^\top)}{Z_{t\setminus t}}\right) \cdot \left(\frac{Z_{t\setminus t}}{Z_t}\right) = a_{t,j}^{nodiag} \cdot (1 - p_{local}) \tag{22}$$

Substituting this into the formulation of the Self-Saliency Score, and omitting the negligible ϵ in the non-diagonal terms for analytical simplicity as $\epsilon \to 0^+$, we obtain:

$$\mathrm{Score}_t \approx p_{local} \log \frac{p_{local}}{\epsilon} + \sum_{j \in \mathcal{W}_t \setminus \{t\}} (1 - p_{local}) a_{t,j}^{nodiag} \log(1 - p_{local}) \tag{23}$$

Since the probabilities sum to one ($\sum_{j \in \mathcal{W}_t \setminus \{t\}} a_{t,j}^{nodiag} = 1$), this simplifies to:

$$\mathrm{Score}_t \approx p_{local} \log \frac{p_{local}}{\epsilon} + (1 - p_{local}) \log(1 - p_{local}) \tag{24}$$

Taking the derivative with respect to p_{local} yields:

$$\frac{\partial \mathrm{Score}_t}{\partial p_{local}} = \log \frac{p_{local}}{\epsilon} + 1 - \log(1 - p_{local}) - 1 = \log \frac{p_{local}}{\epsilon(1 - p_{local})} \tag{25}$$

For an infinitesimally small $\epsilon > 0$ and any valid $p_{local} > 0$, the ratio $\frac{p_{local}}{\epsilon(1-p_{local})} > 1$. Therefore, the derivative is strictly positive ($\frac{\partial \mathrm{Score}_t}{\partial p_{local}} > 0$), proving that Score_t increases strictly monotonically with p_{local}. □

Remark: Practical applicability condition. Lemma 1 is established for $\epsilon \to 0^+$. With a finite ϵ, the monotone relationship between Score_t and p_{local} is conditional: when $p_{\mathrm{local}} \gg \epsilon$, the two quantities follow a similar trend and ϵ is negligible; when p_{local} is comparable to ϵ, the ϵ-smoothed terms dominate, and tokens are re-ranked according to the complete leave-one-out distribution rather than the diagonal term alone. In this regime ϵ acts as a soft gate: a token with a smaller $a_{t,t}^{\mathrm{diag}}$ can still obtain a high score if removing its self-interaction induces a larger change to the local attention distribution. This is precisely the regime in which Self-Saliency departs from — and empirically outperforms — the diagonal-attention baseline (Table 11).

Lemma 2 (Probabilistic Local-Global Consistency)**.** *Assuming that the attention logits for context tokens (i.e.,* $\exp(q_t k_j^\top)$ *for* $j \neq t$*) are independent and identically distributed (i.i.d.) samples from a context distribution with expectation* C*, the global self-attention probability* p_{global} *is a strictly monotonically increasing function of the local self-attention probability* p_{local}*.*

Proof. Let w be the local window size, N be the global sequence length, and $S_t = \exp(q_t k_t^\top)$. By the Law of Large Numbers, for sufficiently large sequence lengths, the local and global partition functions (excluding the query token itself) converge to the expectation C:

$$\frac{1}{w} Z_{local} \xrightarrow{p} C, \quad \frac{1}{N} Z_{global} \xrightarrow{p} C \implies Z_{global} \approx \frac{N}{w} Z_{local} \tag{26}$$

From the definition of $p_{local} = \frac{S_t}{S_t + Z_{local}}$, we can express the local partition function as $Z_{local} = S_t \frac{1 - p_{local}}{p_{local}}$.

The global self-attention probability is defined as $p_{global} = \frac{S_t}{S_t + Z_{global}}$. Substituting $Z_{global} = \frac{N}{w} Z_{local}$ into this formula, we obtain a direct analytical mapping between the local and global probabilities:

$$p_{global} = \frac{S_t}{S_t + \frac{N}{w} S_t \frac{1 - p_{local}}{p_{local}}} = \frac{1}{1 + \frac{N}{w} \left(\frac{1}{p_{local}} - 1 \right)} := g(p_{local}) \tag{27}$$

Letting $\gamma = \frac{N}{w}$ (where $\gamma \geq 1$), we compute the derivative of the mapping function $g(x)$:

$$g'(x) = \frac{\gamma}{x^2 \left(1 + \gamma \left(\frac{1}{x} - 1 \right) \right)^2} \tag{28}$$

For any valid probability $x \in (0, 1)$ and $\gamma > 0$, the derivative $g'(x) > 0$ strictly holds.

Since $g'(x) > 0$, the global self-attention probability is, in expectation, a strictly monotonic function of its local counterpart. Therefore, under the stated assumptions, the local Self-Saliency ranking provides a rank-preserving proxy for its global counterpart. □

A.4 Algorithm

Algorithm 1 Chunk-Wise Parallel Form

Input: $\mathbf{q}, \mathbf{k}, \mathbf{v} \in \mathbb{R}^{N\times d}$, gate $\mathbf{g} \in \mathbb{R}^{N\times d}$, feature maps ϕ_q, ϕ_k, total cache budget m, local-window budget W, configured maximum length $N_{\max}$, chunk size C, and small constant ϵ.

$T_{\max} \leftarrow \lceil N_{\max}/C \rceil$

$\lambda \leftarrow \lfloor (m-W)/T_{\max} \rfloor$

$T \leftarrow \lceil N/C \rceil$ $\triangleright\ T\lambda \le m - W$

Divide $\mathbf{q}, \mathbf{k}, \mathbf{v}, \mathbf{g}$ into T chunks of size $C \times d$.

Initialize masks $\mathbf{M}^{\text{down}}, \mathbf{M}^{\text{up}}, \mathbf{M}^{\text{diag}} \in \{0,1\}^{C\times C}$, where $\mathbf{M}^{\text{down}}_{t,s} = 1$ if $t \ge s$ else 0, $\mathbf{M}^{\text{up}}_{t,s} = 1$ if $t < s$ else 0, and $\mathbf{M}^{\text{diag}}_{t,s} = 1$ if $t > s$ else 0.

Define $\mathrm{Mask}(\mathbf{X}, \mathbf{M})_{i,j} = \begin{cases} \mathbf{X}_{i,j}, & \mathbf{M}_{i,j} = 1, \\ -\infty, & \mathbf{M}_{i,j} = 0. \end{cases}$

Compute Self-Saliency Score:

$\mathbf{s}^{\text{in}} \leftarrow \mathbf{q}\mathbf{k}^\top \in \mathbb{R}^{T\times C\times C}$.

$\mathbf{k}^{\rightarrow}_{[t,:,:]} \leftarrow \mathbf{k}_{[t-1,:,:]}$ for $t > 1$, and zeros for $t = 1$.

$\mathbf{s}^{\text{prev}} \leftarrow \mathbf{q}(\mathbf{k}^{\rightarrow})^\top \in \mathbb{R}^{T\times C\times C}$.

$\mathbf{a}^{\text{diag}} \leftarrow \mathrm{softmax}\left(\left[\mathrm{Mask}(\mathbf{s}^{\text{in}}, \mathbf{M}^{\text{down}}), \mathrm{Mask}(\mathbf{s}^{\text{prev}}, \mathbf{M}^{\text{up}})\right]\right)$.

$\mathbf{a}^{\text{nodiag}} \leftarrow \mathrm{softmax}\left(\left[\mathrm{Mask}(\mathbf{s}^{\text{in}}, \mathbf{M}^{\text{diag}}), \mathrm{Mask}(\mathbf{s}^{\text{prev}}, \mathbf{M}^{\text{up}})\right]\right)$.

$\mathbf{S} \leftarrow \mathrm{sum}\left(\mathbf{a}^{\text{diag}} \log \frac{\mathbf{a}^{\text{diag}}+\epsilon}{\mathbf{a}^{\text{nodiag}}+\epsilon}, \mathrm{dim} = -1\right) \in \mathbb{R}^{T\times C}$.

Local-Global Consistent Selection:

$\mathbf{M}^{\mathcal{SA}} \leftarrow \mathrm{Top}(\mathbf{S}, \lambda), \quad \mathbf{M}^{\mathcal{LA}} \leftarrow \overline{\mathbf{M}^{\mathcal{SA}}}$.

$\mathbf{k}^{\mathcal{SA}} \leftarrow \mathrm{Index}(\mathbf{k}, \mathbf{M}^{\mathcal{SA}}) \in \mathbb{R}^{T\times\lambda\times d}$,

$\mathbf{v}^{\mathcal{SA}} \leftarrow \mathrm{Index}(\mathbf{v}, \mathbf{M}^{\mathcal{SA}}) \in \mathbb{R}^{T\times\lambda\times d}$.

$\mathbf{k}^{\mathcal{LA}} \leftarrow \mathrm{Index}(\mathbf{k}, \mathbf{M}^{\mathcal{LA}}) \in \mathbb{R}^{T\times(C-\lambda)\times d}$,

$\mathbf{v}^{\mathcal{LA}} \leftarrow \mathrm{Index}(\mathbf{v}, \mathbf{M}^{\mathcal{LA}}) \in \mathbb{R}^{T\times(C-\lambda)\times d}$.

Linear Attention Branch:

$\mathbf{s}^{\text{chunk}} \leftarrow \phi_k(\mathbf{k}^{\mathcal{LA}})^\top \mathbf{v}^{\mathcal{LA}} \in \mathbb{R}^{T\times d'\times d}$.

$\mathbf{z}^{\text{chunk}} \leftarrow \mathrm{sum}\left(\phi_k(\mathbf{k}^{\mathcal{LA}}), \mathrm{dim} = 1\right) \in \mathbb{R}^{T\times d'}$.

$\mathbf{s} \leftarrow \mathrm{Cumsum}(\mathbf{s}^{\text{chunk}}, \mathrm{dim} = 1) \in \mathbb{R}^{T\times d'\times d}$.

$\mathbf{z} \leftarrow \mathrm{Cumsum}(\mathbf{z}^{\text{chunk}}, \mathrm{dim} = 1) \in \mathbb{R}^{T\times d'}$.

$\mathbf{s}^{\rightarrow}_{[t,:,:]} \leftarrow \mathbf{s}_{[t-2,:,:]}$ and $\mathbf{z}^{\rightarrow}_{[t,:]} \leftarrow \mathbf{z}_{[t-2,:]}$ for $t > 2$, and zeros otherwise.

Softmax Attention Branch:

$\mathbf{k}^{\mathcal{SA}}_{\text{hist}} \leftarrow \mathrm{Broadcast}\left(\mathrm{Flatten}(\mathbf{k}^{\mathcal{SA}})\right) \in \mathbb{R}^{T\times(T\lambda)\times d}$.

$\mathbf{v}^{\mathcal{SA}}_{\text{hist}} \leftarrow \mathrm{Broadcast}\left(\mathrm{Flatten}(\mathbf{v}^{\mathcal{SA}})\right) \in \mathbb{R}^{T\times(T\lambda)\times d}$.

Initialize mask $\mathbf{M}^{\text{Chunk}} \leftarrow \mathbf{0} \in \{0,1\}^{T\times(T\lambda)}$.

$\mathbf{M}^{\text{Chunk}}_{[t,\ 1:\max(0,t-2)\lambda]} \leftarrow 1, \quad \forall t \in \{1,\dots,T\}$.

$\mathbf{s}^{\mathcal{SA}}_{\text{hist}} \leftarrow \mathrm{Mask}\left(\mathbf{q}(\mathbf{k}^{\mathcal{SA}}_{\text{hist}})^\top, \mathbf{M}^{\text{Chunk}}\right)$.

$\mathbf{o}^{\mathcal{SA}} \leftarrow \exp\left(\left[\mathbf{s}^{\mathcal{SA}}_{\text{hist}}, \mathrm{Mask}(\mathbf{s}^{\text{prev}}, \mathbf{M}^{\text{up}}), \mathrm{Mask}(\mathbf{s}^{\text{in}}, \mathbf{M}^{\text{down}})\right]\right)$.

$\mathbf{v}^{\rightarrow}_{[t,:,:]} \leftarrow \mathbf{v}_{[t-1,:,:]}$ for $t > 1$, and zeros for $t = 1$.

$\widetilde{\mathbf{v}}^{\mathcal{SA}} \leftarrow \left[\mathbf{v}^{\mathcal{SA}}_{\text{hist}}, \mathbf{v}^{\rightarrow}, \mathbf{v}\right]$.

$\mathbf{y} \leftarrow \frac{\mathbf{o}^{\mathcal{SA}}\widetilde{\mathbf{v}}^{\mathcal{SA}} + \phi_q(\mathbf{q})\mathbf{s}^{\rightarrow} \odot \mathbf{g}}{\mathrm{sum}(\mathbf{o}^{\mathcal{SA}}, \mathrm{dim}=-1) + \phi_q(\mathbf{q})\mathbf{z}^{\rightarrow}} \in \mathbb{R}^{T\times C\times d}$.

Output: $\mathbf{y} \leftarrow \mathrm{Flatten}(\mathbf{y}) \in \mathbb{R}^{N\times d}$.

Budget certification and runtime lengths. The per-chunk quota λ is derived from the configured $N_{\max}$ so that the hard budget m is certified for every runtime length $N \le N_{\max}$ ($T\lambda \le T_{\max}\lambda \le m - W$). For $N \ll N_{\max}$ the salient cache is intentionally under-filled, trading utilization for a length-independent memory footprint. When the deployment length is known, λ can equivalently be re-derived from the actual chunk count at inference, $\lambda(N) = \lfloor (m-W)/\lceil N/C \rceil \rfloor$, filling the cache with no architectural or training change; the same re-derivation keeps the budget exact for $N > N_{\max}$, degrading gracefully in selection density rather than exceeding memory. All experiments use the certified static λ for reproducibility.

A.5 More Visualizations

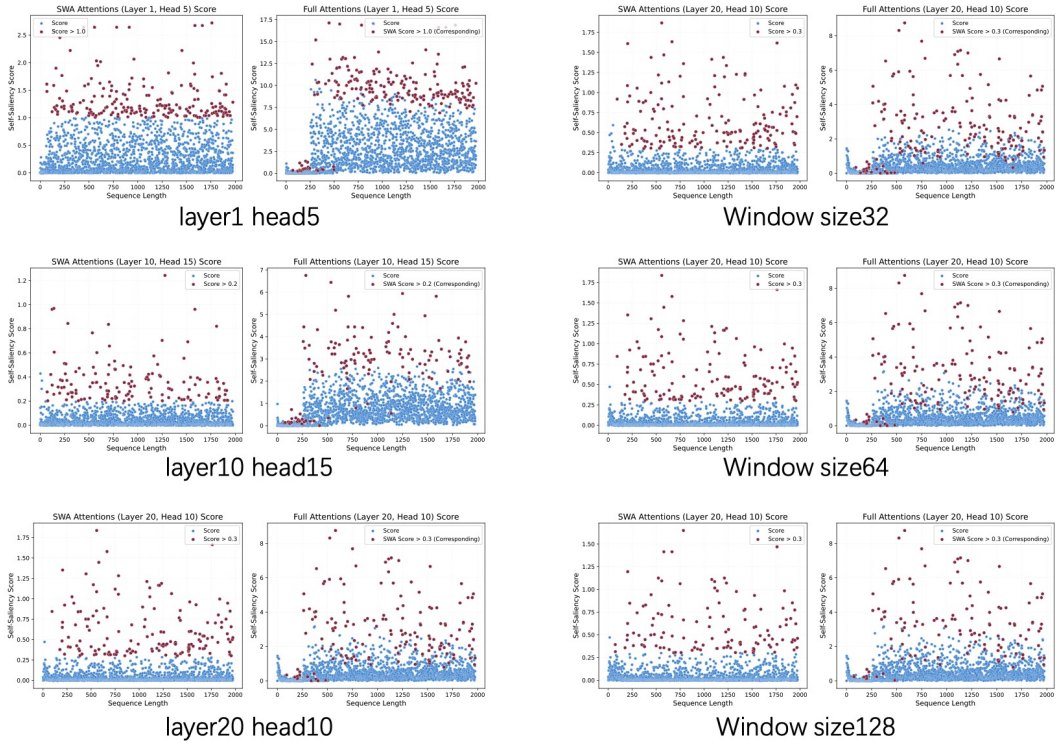


Figure 7: Self-saliency score comparison between local and global attention across different layers, heads, and window sizes.

A.6 Experimental Setups

A.6.1 Training Strategy and Datasets

To rigorously evaluate our proposed STILL module while strictly controlling computational variables against baselines, we introduce a primary lightweight training configuration, alongside an extended configuration to test architectural compatibility.

Primary Configuration (Intra-Layer Validation). For our core experiments (Section 4.2 and 4.3), we strictly adhere to the ultra-lightweight two-stage training procedure proposed in LoLCATs [11]. Specifically, during the *Attention Transfer* stage, we freeze the teacher model parameters and optimize the feature mapping ϕ and the gating projection $\mathbf{W}_g$ by minimizing the layer-wise mean squared error (MSE) between the attention maps of the original and proposed modules. In the *Low-rank Linearization* stage, we perform end-to-end training using Low-Rank Adaptation (LoRA) [23] applied to the query, key, value, and output projection matrices, as well as the gating mechanism ($\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v, \mathbf{W}_o, \mathbf{W}_g$). Both stages are conducted on the Cleaned Alpaca dataset [24] utilizing a minimal budget of only 20M tokens with a constant sequence length of 1024.

Extended Configuration (Architectural Compatibility). To demonstrate STILL's plug-and-play flexibility within macroscopic layer-routing architectures (evaluated in Section 4.4), we integrate our intra-layer mechanism with the KL-LS inter-layer strategy [13] on Llama 3.2 1B. Specifically, 25% of the layers are retained as full softmax attention, while in the remaining layers, STILL selects 12.5% of the tokens for local softmax attention. This dual-hybrid setup operates on the DCLM dataset [60] in two stages: an initial attention transfer phase (100M tokens, sequence length 512), followed by a full-parameter alignment step (250M tokens) adapted to a sequence length of 4096 to accommodate the structural changes introduced by the inter-layer routing.

We note that the routing score is length-invariant by construction: the Self-Saliency Score is computed entirely within a local window of $W \leq 512$ tokens (Eqs. 2–5), so its distribution at long test contexts

matches that at the 1024-token training sequences; what is learned from the training corpus is the fidelity of the LA branch (NP-Map) and the LoRA alignment, i.e., short-range behavior, while long-context capability emerges at inference from the accumulation of per-chunk routing decisions. Our use of Cleaned Alpaca at length 1024 follows the established linearization protocol of LoLCATs [11] and Liger [12], ensuring identical training data and token budgets across all compared methods in Secs. 4.2–4.3.

A.6.2 Evaluation Settings

Commonsense and General Reasoning Tasks Settings. We evaluate the commonsense reasoning and general language understanding capabilities of STILL on a diverse suite of standard benchmarks, including PIQA [26], ARC-Easy and ARC-Challenge [27], HellaSwag [28], WinoGrande [29], and MMLU [18]. Consistent with prior work [30, 11, 12], we evaluate all benchmarks in the zero-shot setting, with MMLU evaluated under the standard five-shot protocol. We select Llama 3 8B and Llama 3.1 8B [31] as backbone models, and compare our method against four categories of baselines: the original Transformer, subquadratic models trained from scratch, inter-layer hybrid models and linearized models.

S-NIAH Settings. We evaluate the S-NIAH tasks [19] with context lengths up to 4K, which assess a model's ability to retrieve a target span from long inputs. We compare STILL against sliding-window attention (SWA), LoLCATs and Liger-GLA with window sizes of 128, 256, and 512, using Llama 3.1 8B as the backbone model. For STILL, we set the local-window size to half of the corresponding baseline configuration and match the overall cache budget by ensuring that the sum of cached tokens and the sliding-window size does not exceed the baseline window size. We keep the training procedure of all models unchanged and only modify the sliding-window size during inference.

Full RULER Settings. We further evaluate STILL on the extended RULER benchmark [19] at a 4K context length, which requires stronger memory and state tracking beyond NIAH-style retrieval. It includes multi-key retrieval (MK), multi-query retrieval (MQ), multi-value retrieval (MV), common word extraction (CWE), frequent word extraction (FWE), HotpotQA (HQA), SQuAD-QA (SQA), and variable tracking (VT). We compare STILL against Mamba2 [32], LoLCATs [11], and Liger [12], using Llama 3.1 8B as the backbone model. LoLCATs and Liger are evaluated with a sliding-window size of 896. STILL is evaluated with a local-window size of 128, and the total number of selected tokens is capped at 896.

BABILong Settings. To evaluate the long-context reasoning capability of STILL, we conduct experiments on BABILong [20], a long-context extension of the bAbI [38] question-answering tasks. BABILong embeds key supporting facts into very long irrelevant contexts, requiring models to integrate evidence across distant positions for reasoning rather than relying on simple keyword matching. We compare STILL with SWA and LoLCATs under a total cache budget of 256, using Llama 3.1 8B as the backbone model. Experiments are conducted across tasks QA1–QA5.

A.7 Ablation Studies

We conduct ablation studies to examine the robustness of STILL under different backbone architectures and model scales, and to quantify the impact of our key design components.

A.7.1 Different Backbone Architectures and Scales.

Tab. 8 summarizes results across different backbone families and model scales. STILL consistently outperforms all linearized baselines in overall performance. With the Llama 3.2 1B [2], STILL improves the average score including MMLU to 54.8 and raises MMLU from 22.4 with Liger to 29.8. Using the stronger Llama 3.2 3B [2], STILL further boosts MMLU to 52.4, compared to 32.1 for Liger, achieving an overall average of 65.5. For a linearized Mistral 7B [41], STILL attains an average score of 70.9 including MMLU, surpassing LoLCATs at 69.1 and Liger at 65.1.

Table 8: **Different Backbone Architectures and Scales Results.** STILL achieves the best performance across all models and scales.

MODEL	TRAINING TOKENS (B)	PIQA acc ↑	ARC-E acc ↑	ARC-C acc_n ↑	HELLA. acc_n ↑	WINO. acc ↑	MMLU (5-shot) ↑	AVG. (w. MMLU)	AVG. (wo. MMLU)
Linearized from Llama 3.2 1B									
Llama 3.2 1B [2]	9000	74.4	65.5	35.8	63.7	60.5	31.9	55.3	60.0
T2R [21]	0.04	69.2	58.2	29.9	42.6	54.1	23.3	46.2	50.8
Hedgehog [17]	0.04	70.1	55.8	29.8	47.7	50.7	23.0	46.2	50.8
LoLCATs [11]	0.04	74.1	63.7	**36.4**	51.2	58.2	23.1	51.1	56.7
Liger-GLA [12]	0.02	**75.0**	**65.4**	35.7	59.8	59.1	22.4	52.9	59.0
STILL	0.04	74.9	65.3	36.3	**62.6**	**59.8**	**29.8**	**54.8**	**59.8**
Linearized from Llama 3.2 3B									
Llama 3.2 3B [2]	9000	76.4	74.7	46.0	73.6	69.9	56.2	66.1	68.1
LoLCATs [11]	0.04	76.7	72.0	42.3	51.9	66.9	23.6	55.6	62.0
Liger-GLA [12]	0.02	**77.9**	74.0	43.9	70.3	66.3	32.1	60.7	66.5
(Ours)	0.04	77.6	**76.1**	**47.5**	**72.0**	**67.6**	**52.4**	**65.5**	**68.2**
Linearized from Mistral 7B									
Mistral 7B [41]	8000	82.1	80.9	53.8	81.0	74.0	62.4	72.4	74.4
SUPRA [30]	20	80.1	74.6	42.3	74.8	67.4	28.0	61.2	67.8
SUPRA [30]	100	80.4	75.9	45.8	77.1	70.3	34.2	64.0	69.9
LoLCATs [11]	0.04	81.5	81.7	54.9	79.2	71.0	46.2	69.1	73.7
Liger-GLA [12]	0.02	80.1	78.7	49.3	76.3	70.1	36.3	65.1	70.9
(Ours)	0.04	**81.7**	**82.7**	**55.6**	**80.0**	**71.6**	**53.8**	**70.9**	**74.3**

A.7.2 Module Ablation

We further conduct a module ablation study of STILL on the BABILong benchmark with a sequence length of 1K, as shown in Tab. 9. Removing either the gating mechanism or the non-parametric mapping (NP-Map) degrades performance by 5.8 and 4.9 points, respectively, and removing both causes an 8.3-point drop. When the self-saliency-based token selection strategy is further removed, the accuracy drops from 27.5 to 12.5, demonstrating that each component contributes meaningfully and that they are complementary.

Table 9: **Module Ablation Results.** Ablation study of key components in STILL.

SELF-SALIENCY SCORE	NP-MAP	GATE	ACC.
✓	✓	✓	27.5
✓	✓		21.7$_{-5.8}$
✓		✓	22.4$_{-4.9}$
✓			19.2$_{-8.3}$
			12.5$_{-15.0}$

A.7.3 Ablation of Token-Routing Metrics

To examine whether the proposed Self-Saliency Score provides a more effective routing signal than simpler alternatives, we compare it with several natural token-selection criteria under the same routing and cache configuration. We consider both short-context commonsense reasoning, where long-range cache routing is relatively unimportant, and S-NIAH retrieval, where a small number of relevant tokens must be retained among many distractors under a strict cache budget.

As shown in Tab. 10, replacing Self-Saliency with the diagonal self-attention weight causes only a modest 0.5-point degradation on standard commonsense reasoning tasks. This is expected because these benchmarks contain relatively short contexts, where retaining sparse long-range evidence is not the primary bottleneck. In this regime, diagonal attention already captures much of the dominant routing trend.

Table 10: **Routing-metric ablation on short-context commonsense reasoning.** We report the average performance over PIQA, ARC-E, ARC-C, HellaSwag, WinoGrande, and MMLU. Δ denotes the difference from Self-Saliency.

Routing Metric	**Avg.** $\uparrow$	Δ
Self-Saliency	**72.5**	–
Diagonal attention	72.0	-0.5

The distinction becomes substantially larger on S-NIAH (Tab. 11), where routing is central to preserving remote evidence under a limited cache budget. Self-Saliency achieves an average accuracy of 42.6, compared with 23.1 for diagonal attention and 22.8 for $\mathbf{q}_t^\top \mathbf{k}_t$, while norm-, uniform-, and random-based selection perform considerably worse. These results indicate that diagonal attention captures a strong local routing signal in ordinary short-context settings, but does not provide an equally effective substitute when accurate sparse long-range retention is required. Self-Saliency is particularly beneficial in this routing-sensitive regime.

Table 11: **Routing-metric ablation on long-context S-NIAH retrieval.** All methods use the same cache budget. Avg. denotes the mean accuracy over the 0.5K and 1K context lengths, and Δ denotes the difference from Self-Saliency.

Routing Metric	**0.5K** $\uparrow$	Δ	**1K** $\uparrow$	Δ	**Avg.** $\uparrow$
Self-Saliency	**63.2**	–	**22.0**	–	**42.6**
Diagonal attention	35.6	-27.6	10.6	-11.4	23.1
$\mathbf{q}_t^\top \mathbf{k}_t$	35.4	-27.8	10.2	-11.8	22.8
Key norm	18.6	-44.6	7.1	-14.9	12.9
Uniform per chunk	17.8	-45.4	6.8	-15.2	12.3
Random selection	18.6	-44.6	7.0	-15.0	12.8

A.7.4 Future KV utility

Self-Saliency is computed causally within a token's local window, whereas routing requires estimating whether the token will be attended by *future* queries after it becomes remote context. We directly measure this correspondence. Under the standard delayed-selection procedure, we select the Top-224 tokens with locally computed Self-Saliency, then run full attention and rank the eligible remote keys — excluding the sliding-window and delayed-selection regions — by the attention weights assigned by subsequent queries. As shown in Table 12, the causal local score recovers 62.9%–84.8% of the remote keys actually prioritized by future full-attention queries across representative heads in shallow (0), middle (23, 24), and deep (28) layers. We do not include accumulated or recent-received attention as a baseline: under delayed chunk routing it requires an additional causal observation horizon and a normalization for unequal exposure across positions, and maintaining such query-dependent statistics contradicts the online, state-free nature of our routing. Oracle selection with full attention is a non-deployable quadratic upper bound and is instead reflected by the overlap measurement itself. Together with the downstream retrieval results (Table 11), these results support Self-Saliency as an effective online proxy of future KV utility.

Table 12: Overlap between the Top-224 tokens selected by locally computed Self-Saliency and the Top-224 remote keys prioritized by future full-attention queries (Llama 3.1 8B).

Layer	Head	Selected	Overlap	Ratio
0	0	224	141	62.9%
0	1	224	145	64.7%
0	2	224	158	70.5%
0	3	224	190	84.8%
23	0	224	185	82.6%
23	1	224	188	83.9%
23	2	224	188	83.9%
23	3	224	143	63.8%
24	0	224	168	75.0%
24	1	224	163	72.8%
24	2	224	164	73.2%
24	3	224	150	67.0%
28	0	224	145	64.7%
28	1	224	180	80.4%
28	2	224	169	75.4%
28	3	224	150	67.0%

A.7.5 Sensitivity to Chunk Size and Delayed Selection

In STILL, the local window spans one chunk, so the chunk size C jointly controls the observation window of Self-Saliency and the routing delay. We vary C while keeping the total selection budget unchanged (16 selected remote tokens) and using the same checkpoint.

Routing quality. Table 13 shows that delayed selection introduces no systematic accuracy loss: performance improves from C=16 to C=64 and remains close at C=128, indicating that moderate chunk-level delay preserves routing quality and can even outperform finer-grained selection under the same budget. The lower performance at C=16 suggests that the local window should not be too small, both to preserve short-range generation quality and to support reliable saliency estimation.

Table 13: Commonsense reasoning accuracy under different chunk sizes C with a fixed selection budget.

C	PIQA	ARC-e	ARC-c	Hella.	Wino.	MMLU	Avg. w/ MMLU	Avg. w/o MMLU
16	80.5	81.3	53.4	77.2	71.0	54.2	69.6	72.7
64	81.4	82.8	56.4	79.0	73.1	60.8	72.3	74.5
128	81.3	82.7	56.2	79.0	71.1	62.1	72.1	74.1

Hardware efficiency. Table 14 reports training cost on a single NVIDIA A6000. Increasing C from 16 to 128 reduces training time by 35.4% and memory by 21.7%, while C=4 runs out of memory. Exact per-token routing corresponds to C=1: it breaks chunk-level parallelism, and in separate profiling it increases decoding selection latency by more than 3$\times$ and reduces training speed by more than 50%. Overall, C=64 provides the best balance between routing quality and hardware efficiency and is our default configuration.

Table 14: cost per step under different chunk sizes C.

C	Time (s/step)	Memory (GB)
4	OOM	OOM
8	20.3	46.4
16	14.7	37.4
32	11.3	32.9
64	9.7	30.6
128	9.5	29.3

A.8 Full Tables

Table 15: **Comparison of the Commonsense and General Reasoning Tasks**. Comparison of STILL with baseline methods under Llama 3 8B and Llama 3.1 8B models. STILL consistently demonstrates strong performance across benchmarks.

Model	Training Tokens (B)	PIQA acc ↑	ARC-e acc ↑	ARC-c acc_n ↑	Hella. acc_n ↑	Wino. acc ↑	MMLU (5-shot) ↑	Avg. (w. MMLU)	Avg. (wo. MMLU)
Transformer									
Llama 3 8B [2]	15000+	79.5	80.0	53.3	79.1	73.1	65.3	71.8	73.0
Llama 3.1 8B [2]	15000+	81.1	81.7	55.1	79.3	73.9	68.0	74.2	73.2
Subquadratic									
Mamba 8B [61]	1100	78.9	75.4	42.2	75.6	68.3	28.0	61.4	68.1
Mamba2 8B [32]	3500	79.8	75.9	48.1	77.1	71.6	48.7	67.0	70.6
RWKV-6 7B [62]	1420	78.7	76.8	46.3	75.1	70.0	-	69.4	69.4
TransNormerLLM 7B [48]	1400	80.1	75.4	44.4	75.2	66.1	43.1	64.1	68.2
Hawk 7B [33]	300	80.0	74.4	45.9	77.6	69.9	35.0	63.8	69.6
Griffin 7B [33]	300	81.0	75.4	47.9	78.6	72.6	39.3	65.8	71.1
Falcon3-Mamba 7B [34]	7300	80.6	78.9	57.1	80.1	73.7	55.1	70.9	74.1
Hybird									
StripedHyena-Nous 7B [35]	-	78.8	77.2	40.0	76.4	66.4	26.0	60.8	67.8
Zamba 7B [36]	1000	81.4	74.5	46.6	80.2	76.4	57.7	69.5	71.8
RecurrentGemma 9B [37]	2000	80.6	78.9	57.1	80.1	73.7	55.1	70.9	74.1
Linearized from Llama 3 8B									
T2R-LoLCATs [21]	0.04	62.0	42.1	24.7	32.7	48.3	23.2	38.8	42.0
Hedgehog-LoLCATs [17]	0.04	77.4	71.1	40.6	50.7	54.3	24.2	53.1	58.8
SUPRA [30]	20	78.9	75.1	46.5	71.7	65.8	40.9	63.2	67.6
LoLCATs [11]	0.04	80.0	78.8	52.2	75.6	**73.3**	43.9	67.3	72.0
Liger-GLA [12]	0.02	80.3	81.1	52.5	76.3	72.0	43.4	67.6	72.4
STILL (Ours)	0.04	**81.3**	**81.7**	**54.8**	**78.1**	73.2	**59.7**	**71.5**	**73.8**
Linearized from Llama 3.1 8B									
Llamba [31]	12	80.9	82.5	54.6	77.6	73.3	60.0	71.5	73.8
LoLCATs [11]	0.04	80.9	82.6	54.7	76.6	**73.6**	50.8	69.9	73.7
Liger-GLA [12]	0.02	80.5	81.8	55.6	75.4	69.5	46.9	68.3	72.6
STILL (Ours)	0.04	**81.3**	**83.0**	**56.7**	**79.0**	73.4	**61.3**	**72.5**	**74.7**

Table 16: **Full Results of the Comparison of the Single Needle-in-a-Haystack (S-NIAH) Tasks**. Accuracy on S-NIAH tasks from the RULER benchmark across context lengths from 0.5K to 4K. Methods are compared under different cache budgets and our approach evaluated under matched or smaller cache constraints.

Model	Cache Tokens	S-NIAH-1				S-NIAH-2			
		0.5K	1K	2K	4K	0.5K	1K	2K	4K
SWA	128	0	0	0	0	100	0	0	0
LoLCATs [11]	128	29.4	9.6	3.0	0	100	17.4	**7.2**	**4.2**
Liger-GLA [12]	128	28.4	0.2	0.2	0.2	100	1.6	0.6	1.0
STILL (Ours)	≤128	**63.2**	**22.0**	**5.8**	**1.4**	**100**	**33.6**	3.4	2.8
SWA	256	66.4	4.2	0.2	0	100	0	0.2	0
LoLCATs [11]	256	84.8	26.2	10.2	2.2	100	37.0	12.2	8.2
Liger-GLA [12]	256	83.8	0.0	0.0	2.8	100	1.0	4.6	0.8
STILL (Ours)	≤256	**94.0**	**86.6**	**31.0**	**10.6**	**100**	**63.0**	**27.2**	**9.4**
SWA	512	100	25.2	14.6	5.8	100	44.6	0.2	0
LoLCATs [11]	512	100	65.6	24.6	8.8	100	71.8	17.4	16.6
Liger-GLA [12]	512	73.4	1.0	0.0	0.0	100	2.6	4.0	0.0
STILL (Ours)	≤512	**100**$_{+0.0}$	**99.6**$_{+34.0}$	**89.0**$_{+64.4}$	**86.2**$_{+77.4}$	**100**$_{+0.0}$	**92.6**$_{+20.8}$	**73.8**$_{+56.4}$	**37.4**$_{+20.8}$

A.9 Future Work

In future work, we aim to further improve the effectiveness, efficiency, and generalization of linearized Transformers across diverse tasks and modalities. First, recent advances in latent reasoning [63, 64] motivate us to explore compact representations within linear attention states for efficient reasoning, while expressive memory update mechanisms [65] may enhance long-range information retention. Second, building on low-rank linearization methods [11, 12] and dynamic multi-subspace adaptation [66], we will investigate adaptive parameter-efficient strategies to mitigate performance degradation during Transformer linearization. Finally, we aim to extend linearized Transformers to multimodal settings by integrating efficient pretraining strategies [67, 68, 69] to support scalable multimodal representation learning and long-context understanding. Together, these directions could further advance the capabilities and applicability of linearized Transformers.

Table 17: **BABILong Benchmark Full Results.** Performance comparison on BABILong across increasing context lengths (0K∼4K), evaluating long-context reasoning performance.

MODEL	QA1				QA2				QA3				QA4				QA5			
	0K	1K	2K	4K	0K	1K	2K	4K	0K	1K	2K	4K	0K	1K	2K	4K	0K	1K	2K	4K
SWA	93	2	1	1	34	3	2	0	8	4	2	1	74	0	0	0	68	18	11	6
LoLCATs	100	22	5	3	25	10	4	1	30	17	13	4	65	21	9	1	62	42	17	6
STILL	**100**	**45**	**22**	**10**	**57**	**19**	**9**	**5**	**37**	**24**	**15**	**11**	**82**	**59**	**30**	**9**	**83**	**77**	**45**	**24**

Table 18: Extended S-NIAH-1 evaluation up to 8K on Llama 3.1 8B under the 512-token cache budget, STILL retains meaningful retrieval accuracy at 8K, while all baselines collapse to near zero.

S-NIAH-1	0.5K	1K	2K	4K	8K
SWA	100	25.2	14.6	5.8	3.0
LoLCATs	100	65.6	24.6	8.8	0.0
Liger-GLA	73.4	1.0	0.0	0.0	0.0
STILL (Ours)	100	99.6	89.0	86.2	36.2

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: See the last two paragraphs in Section 1 Introduction.

   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: See in the appendix.

   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: See in the APPENDIX.

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: See in section 4 and Appendix A.6.

   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 is contained in supplementary material and we only use Open Source Datasets.

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: See in section 4 and Appendix A.6.

   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: See in the Results part of each subsection in Section 4.

   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: See in Section 4 and APPENDIX.

   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: We have make sure to preserve anonymity.

   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: [N/A]

    Justification: This work focus on a general hybrid attention model with no direct societal impact.

    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: This work is a research on designing transformer models with no such risks.

    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: We use all assets properly according to their licenses, see in APPENDIX.

    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: [N/A]

    Justification: This work does not release new assets.

    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: This work does not have crowdsourcing experiments or research with human subjects.

    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: This work does not have crowdsourcing experiments or research with human subjects.

    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: This work only use LLM for writing and editing.

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.