Evolving Sparsity: Leveraging Token Importance Dynamics for Efficient LLM Decoding with Sparse Attention

Ruizi Han[1,2], Miao Zhang[1*], Ziyue Qiao[2*], Liqiang Nie[1]
[1] Harbin Institute of Technology (Shenzhen)
[2] Great Bay University
25B951126@stu.hit.edu.cn, zhangmiao@hit.edu.cn
zyqiao@gbu.edu.cn, nieliqiang@gmail.com

Abstract

Efficient long-context inference remains a major challenge for large language models (LLMs), as the cost of attention computation during auto-regressive decoding grows linearly with the context length. Recent sparse attention methods attempt to reduce the computational burden by selecting a subset of tokens at each step, while most rely on static importance scores that are repeatedly computed over the entire cache, overlooking the relational dynamics of the decoding process. In this work, we revisit sparse attention in LLMs and propose to model token importance as a dynamic process that evolves over decoding steps and propagates through model layers. To efficiently measure token importance, we propose two lightweight mechanisms: (1) Cross-Step Accumulation, which incrementally maintains long-term, query-agnostic importance via decayed accumulation of sparse attention scores, avoiding recomputing the importance of decoded tokens; and (2) Cross-Layer Propagation, which leverages the model's intrinsic Retrieval Heads to compute query-aware indices and efficiently propagate them across layers; Together, these mechanisms preserve both stable context memory and adaptive query relevance while reduce redundant computation. We evaluate our approach on PG-19, RULER, LongBench, and mathematical reasoning benchmarks using models employing Multi-Head and Grouped-Query Attention. Under varying KV cache budgets, our method consistently outperforms prior sparse attention baselines, approaches full attention performance in most settings, and achieves speedups of up to 5.36× for attention latency and 2.33× for end-to-end decoding. Our code is available at: https://github.com/iLearn-Lab/ACL26-EvoSparse.

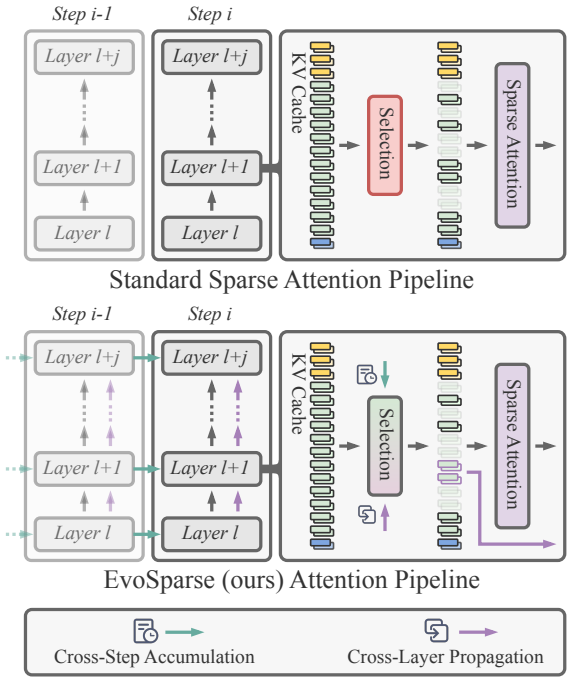

Figure 1: Pipeline overview. Unlike standard methods that perform independent selection across steps and layers, EvoSparse exploits intrinsic correlations through Cross-Step Accumulation and Cross-Layer Propagation, thereby enhancing both effectiveness and efficiency.

1 Introduction

Large Language Models (LLMs) such as GPT (Achiam et al., 2023), LLaMA (Grattafiori et al., 2024), and Gemini (Team et al., 2023) have demonstrated remarkable capabilities in reasoning, knowledge retrieval, and generation across a wide range of tasks (Bai et al., 2023; Shaham et al., 2023; An et al., 2023; Zhang et al., 2024). A key factor enabling these abilities is the model's capacity to process long sequences of tokens. However, as context lengths grow, the computational and memory demands of the attention mechanism scale quadratically in standard architectures, creating a substantial bottleneck for long-context modeling (Fu, 2024). The challenge is further amplified

* Co-corresponding Authors.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 11554–11566

during auto-regressive decoding, where each new query token requires attending over the entire preceding context.

Fortunately, LLMs exhibit a form of inherent sparsity (Deng et al., 2024): only a subset of tokens typically contributes meaningfully to attention outputs. Motivated by this, various decoding-focused sparse attention techniques have been developed, such as Loki (Singhania et al., 2024), Sparq Attention (Ribar et al., 2023), and Quest (Tang et al., 2024). While effective in reducing costs, these methods typically treat token selection as an independent, stateless operation at each step and layer (Figure 1). This design overlooks two critical intrinsic properties of attention computation that we observe in modern LLMs: (1) ***Temporal Consistency***: Token importance is not transient; significant tokens often serve as Long-term Anchors, and attention patterns exhibit Short-term Reuse between adjacent steps. (2) ***Head-Level Capability Gap***: While prior research has identified the existence of specialized Retrieval Heads (Wu et al., 2024), we observe a critical behavioral divergence during decoding. These specialized heads accurately locate key information, whereas the vast majority of Standard Heads often fail to capture dependencies when operating independently, attending instead to irrelevant noise. Existing methods failure to leverage these characteristics results in suboptimal selection quality and redundant computations, limiting the trade-off between performance and efficiency.

Based on these observations, we propose **EvoSparse**, a framework that models token importance as a continuously evolving process rather than a static snapshot (Figure 1). Our approach is instantiated through two synergistic mechanisms: (1) ***Cross-Step Accumulation***: Addressing the Temporal Consistency, we introduce a lightweight accumulation mechanism that updates token importance via an exponential moving average. This captures Long-term Anchors by aggregating attention mass and enforces Short-term Reuse by maintaining stability between steps, providing a robust global context view. (2) ***Cross-Layer Propagation***: Addressing the Capability Gap, we utilize the indices identified by the few Retrieval Heads to guide the Standard Heads in subsequent layers. This mechanism ensures that the Standard Heads, which typically lack intrinsic retrieval precision, align with high-quality retrieval signals, avoiding the overhead of redundant and error-prone searches. Together, these mechanisms convert sparse attention into a dynamic system that improves selection accuracy while reducing redundant computation.

We conduct a comprehensive evaluation on diverse long-context benchmarks, including language modeling on PG-19 (Rae et al., 2019), RULER (Hsieh et al., 2024), LongBench (Bai et al., 2023), and challenging mathematical reasoning tasks. Our method achieves the best balance between performance and efficiency: it consistently outperforms existing sparse attention baselines, such as Quest and TidalDecode (Yang et al., 2024b), particularly in low-resource regimes. Moreover, our optimized implementation delivers substantial speedups, with attention latency improved by up to 5.36× and end-to-end latency by up to 2.33×.

2 Related Work

Long-Context LLMs. Scaling LLMs to long contexts is constrained by the quadratic complexity of attention. To extend effective context windows, prior work has focused on continuous training with long-text corpora (Chen et al., 2023; Xiong et al., 2023; Fu et al., 2024) and advanced positional extrapolation methods like RoPE (Su et al., 2024) and YaRN (Peng et al., 2023). Parallel efforts employ Retrieval-Augmented Generation (RAG) to offload memory burdens to external storage (Tworkowski et al., 2023; Mohtashami and Jaggi, 2023; Xu et al., 2023). Despite these advances, the intrinsic computational cost of the attention mechanism remains a prohibitive barrier, necessitating methods that directly reduce its algorithmic complexity.

Sparse Attention. Sparse attention mitigates quadratic complexity by processing subsets of tokens, generally categorized into eviction-based and selection-based approaches.

Eviction-based methods maintain a fixed budget by permanently discarding tokens. H_2O (Zhang et al., 2023) and StreamingLLM (Xiao et al., 2023) prioritize "Heavy Hitters" or sink/local tokens, respectively. DuoAttention (Xiao et al., 2024) attempts to mitigate long-range loss by preserving full attention on Retrieval Heads. However, eviction inherently risks the irreversible loss of context in compressed heads, potentially discarding information critical for future generation.

Selection-based methods dynamically choose tokens from the full KV cache at each step. Quest (Tang et al., 2024) and Sparq (Ribar et al., 2023) estimate importance via query-aware metrics, whereas Loki (Singhania et al., 2024) employs

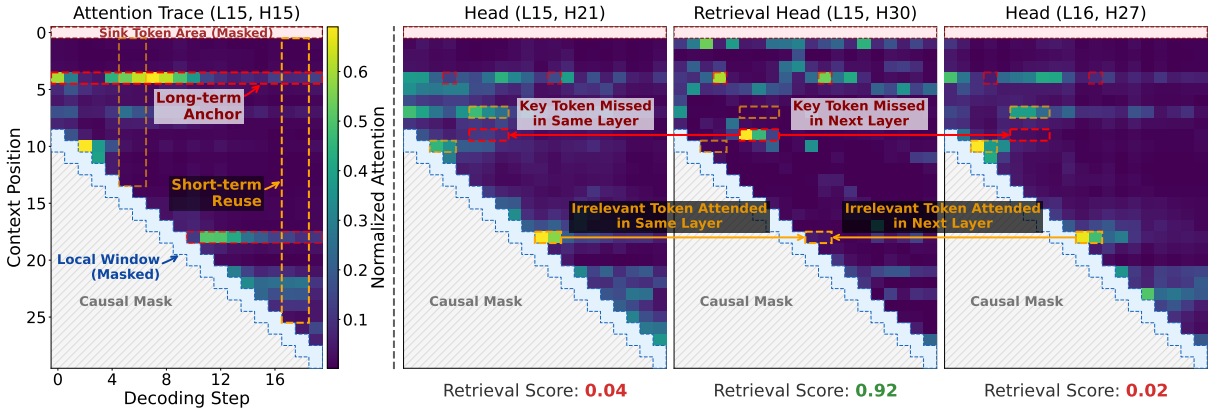

Figure 2: Attention dynamics in Llama-3-8B-1M. Sink and local tokens are masked to isolate intermediate dependencies. Left: The trace exhibits clear temporal stability via Long-term Anchors and Short-term Reuse. Right: A functional gap is observed; while the Retrieval Head pinpoints key tokens, Standard Heads in adjacent layers fail to capture these dependencies, attending instead to irrelevant noise.

low-rank approximations to identify critical tokens. Despite retaining the full cache, these methods typically treat decoding steps independently, neglecting the temporal stability of token importance.

3 Background and Motivation

3.1 Retrieval Heads in LLMs

Recent studies identify a specialized subset of attention mechanisms termed Retrieval Heads (Wu et al., 2024; Xiao et al., 2024), which are pivotal for locating "Needle-in-a-Haystack" information (Kamradt, 2023). Following Wu et al. (2024), we quantify the retrieval capability via the retrieval score, defined as the token-level recall rate of valid copy-paste operations where the head attends to the correct token in the context needle. Throughout this paper, we refer to the remaining Non-Retrieval Heads as Standard Heads.

Although essential for reasoning, these Retrieval Heads constitute only a small fraction of the total model capacity. The core challenge for efficient inference is thus to preserve these critical heads while pruning the redundant computation of Standard Heads. Existing approaches, however, often fall short by treating all heads uniformly (Tang et al., 2024) or relying on static allocation (Xiao et al., 2024), neglecting the dynamic functional interplay between them.

3.2 Empirical Observations

To explore intrinsic redundancy in attention computation, we analyze the attention patterns of Llama-3-8B-1M (Pekelis et al., 2024) on the PG-19 dataset (Rae et al., 2019). As shown in Figure 2, we mask out Sink tokens and Local windows prior to visualization. Since LLMs disproportionately allocate attention mass to these areas (Xiao et al., 2023), this masking is vital to unveil the model's true distribution over the intermediate context, which is the cornerstone of long-text understanding.

Anchors and Reuse. Figure 2 (Left) reveals significant temporal stability. First, we observe Long-term Anchors, where specific key tokens consistently trigger high activation across multiple steps, manifesting as persistent horizontal lines. Second, we observe Short-term Reuse, where the attention distribution of the current query remains highly similar to that of its immediate predecessor. These observations imply that token importance is not transient; rather, it accumulates and persists over the decoding trajectory.

The Gap Between Heads. Figure 2 (Right) highlights a functional gap. A specialized Retrieval Head (e.g., L15, H30) accurately assigns high probability to ground-truth key tokens. In contrast, Standard Heads in the same or subsequent layers (e.g., L15, H21; L16, H27) often fail to capture these dependencies when operating independently, attending instead to irrelevant noise. However, since the tokens identified by the Retrieval Head are semantically significant, this suggests that the critical retrieval signals captured by these heads can be propagated to guide the Standard Heads.

4 Method

Based on insights from Section 3.2, we propose EvoSparse (Figure 3), a novel sparse attention framework designed to capture both the temporal persistence of key tokens and the high-quality retrieval signals inherent in Retrieval Heads.

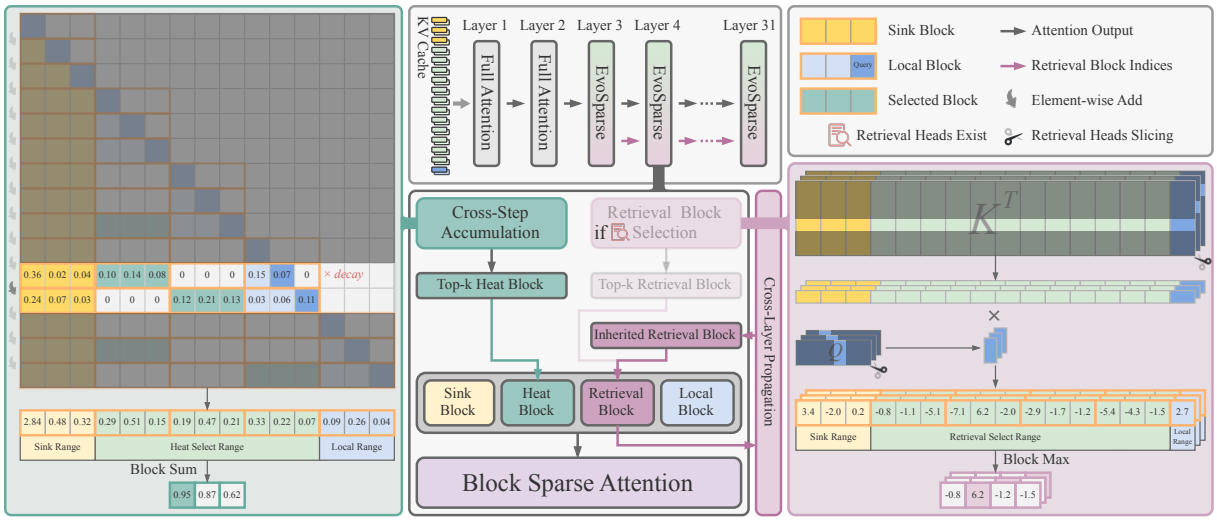

Figure 3: Overview of EvoSparse. The framework integrates two key mechanisms: (1) Cross-Step Accumulation, which employs an exponential moving average to preserve long-term anchors while ensuring short-term temporal stability; and (2) Cross-Layer Propagation, which distributes high-quality retrieval indices to guide Standard Heads, thereby enhancing their focus while maintaining efficiency.

4.1 Cross-Step Accumulation: Exploiting Temporal Stability

Motivated by the Long-term Anchors and Short-term Reuse observed in Section 3.2, we introduce Cross-Step Accumulation (Figure 3, Left) to structurally enforce temporal consistency.

We maintain a global importance vector $\mathbf{h}^t \in \mathbb{R}^N$, termed Heat, which provides a query-agnostic summary of token importance throughout the generation process. The Heat value for each token i at step t is defined as the exponentially decayed sum of its historical sparse attention scores:

$$h_i^t = \sum_{\tau=1}^{t} \lambda^{t-\tau} s_i^\tau, \quad \forall i \in 1, \ldots, N \tag{1}$$

where $\lambda \in (0, 1)$ is a decay factor and s_i^τ represents the sparse attention score at step τ. This formulation naturally encapsulates two key dynamics of token salience:

By isolating the most recent terms of the summation, the update can be viewed as $h_i^t = s_i^t + \lambda s_i^{t-1} + \ldots$. The heavy weighting of scores from proximal steps allows the Heat vector to capture the temporal correlation between adjacent decoding steps, effectively modeling Short-term Reuse of the immediate context.

The summation over the entire trajectory $\tau \in [1, t]$ facilitates the emergence of Long-term Anchors. Tokens that are consistently attended to across many steps will accrue significant Heat through accumulation, even if their instantaneous attention scores s_i^t are relatively moderate, thereby identifying them as globally significant anchors.

Finally, we identify the most salient context by selecting the top-k_h tokens based on their accumulated Heat:

$$\mathcal{I}_{\text{heat}}^t = \text{TopK}\left(\{h_i^t\}_{i=1}^N, k_h\right) \tag{2}$$

This mechanism ensures that the model preserves a stable, query-agnostic representation of globally important tokens to guide generation.

4.2 Cross-Layer Propagation: Bridging the Head Gap

To bridge the Functional Gap (Figure 2, Right), where Standard Heads fail to independently locate key dependencies, we introduce Cross-Layer Propagation (Figure 3, Right). This mechanism treats Retrieval Heads as global experts, broadcasting their high-quality indices to guide the attention of Standard Heads.

For a layer l containing a set of Retrieval Heads $\mathcal{H}_{\text{ret}}^l$, we aggregate the most relevant tokens based on query-key similarity:

$$\mathcal{I}_{\text{retrieval}}^l = \bigcup_{h \in \mathcal{H}_{\text{ret}}^l} \text{TopK}\left(\frac{q_h^l (K_h^l)^\top}{\sqrt{d}}, k_r\right), \tag{3}$$

where k_r denotes the retrieval budget.

Crucially, these indices serve Standard Heads in both the current and subsequent layers, enabling them to bypass computationally expensive global

search and directly focus on semantically significant regions. The propagated index set for a target layer l' is defined recursively:

$$\mathcal{I}^{l'}_{\text{prop}} = \begin{cases} \mathcal{I}^{l'}_{\text{retrieval}} & \text{if } l' \text{ is a retrieval layer,} \\ \mathcal{I}^{l_{\text{last}}}_{\text{retrieval}} & \text{otherwise,} \end{cases} \tag{4}$$

where l_{last} denotes the nearest preceding retrieval layer. This strategy aligns the focus of Standard Heads with expert retrieval signals, effectively rectifying the attention misalignment.

4.3 Unified Sparse Attention

At decoding step t and layer l, EvoSparse synthesizes these components into a unified framework. The final sparse index set $\mathcal{I}^{t,l}$ is constructed by integrating the indispensable Sink $\mathcal{I}_{\text{sink}}$ and Local $\mathcal{I}_{\text{local}}$ tokens with the temporally stable tokens $\mathcal{I}^t_{\text{heat}}$ and the propagated retrieval tokens $\mathcal{I}^l_{\text{prop}}$:

$$\mathcal{I}^{t,l} = \mathcal{I}_{\text{sink}} \cup \mathcal{I}_{\text{local}} \cup \mathcal{I}^t_{\text{heat}} \cup \mathcal{I}^l_{\text{prop}}. \tag{5}$$

To maintain a constant computational overhead, the parameter k_h is dynamically determined by subtracting the number of retrieval-based tokens from a fixed total token budget k, formulated as $k_h = k - |\mathcal{I}_{\text{sink}}| - |\mathcal{I}_{\text{sink}}| - |\mathcal{I}^l_{\text{prop}}|$. It should be noted that while k_h and k_r are defined here at the token-level for conceptual simplicity, the actual implementation of EvoSparse operates at the block-level to optimize hardware IO efficiency.

The attention computation is then explicitly restricted to this selected subset:

$$\text{Attn}(q^l, K^l, V^l) = \text{softmax}\left(\frac{q^l (K^l_{\mathcal{I}^{t,l}})^\top}{\sqrt{d}}\right) V^l_{\mathcal{I}^{t,l}}. \tag{6}$$

This design ensures complementary coverage: Cross-Layer Propagation provides immediate, query-aware precision, while Cross-Step Accumulation ensures long-term context stability. This synergy enables EvoSparse to approximate Full Attention performance with significantly reduced computational overhead.

4.4 Complexity Analysis

To quantify the efficiency gains of EvoSparse, we analyze its computational complexity (FLOPs) and memory IO cost relative to standard attention. Let N denote the sequence length, d the head dimension, L the number of layers, and H the number of KV heads per layer. We define k as the sparse budget and N_{ret} as the total number of top selected Retrieval Heads across the model.

Computational Complexity. While Full Attention scales linearly ($4dNLH$), EvoSparse restricts this growth to a minimal scope. Linear complexity is confined to the first two dense layers ($8dNH$) and the small set of N_{ret} Retrieval Heads. In contrast, the vast majority of heads operate on a fixed budget k, incurring only constant $O(k)$ cost. Since the linear components constitute a negligible fraction of the model capacity ($2 \ll L$ and $N_{ret} \ll LH$), the aggregate complexity for long sequences ($N \gg k$) is effectively dominated by the constant sparse term rather than the linear baseline.

Memory Access Analysis. Given that KV cache IO is the primary bottleneck in decoding, we analyze the cost ratio of EvoSparse relative to Full Attention ($C_{\text{full}} = 2dNLH$):

$$\frac{C_{\text{evo}}}{C_{\text{full}}} \approx \underbrace{\frac{2}{L}}_{\substack{\text{Dense} \\ \text{Floor}}} + \underbrace{\frac{k}{N}\left(1 - \frac{2}{L}\right)}_{\text{Sparsity}} + \underbrace{\frac{N_{ret}}{2HL}}_{\substack{\text{Retrieval} \\ \text{Overhead}}}. \tag{7}$$

This decomposition reveals three structural advantages: (1) The dense layers impose a minimal constant floor (e.g., $\approx 6\%$ for $L = 32$); (2) As N grows, the sparsity term dominates, effectively decoupling IO cost from linear scaling; and (3) The retrieval overhead scales inversely with model capacity (LH), remaining negligible ($N_{ret} \ll LH$) even in GQA (Ainslie et al., 2023) settings.

5 Experiments

5.1 Setups

Tasks and Models. We evaluate EvoSparse on a diverse benchmark suite: (1) PG-19 (Rae et al., 2019) for language modeling; (2) LongBench (Bai et al., 2023) and RULER (Hsieh et al., 2024) for long-context understanding and retrieval robustness; and (3) Mathematical Reasoning (e.g., AIME 24) to assess long-chain CoT capabilities. Correspondingly, we employ Llama-2-7B-32K (Grattafiori et al., 2024), Llama-3-8B-1M (Pekelis et al., 2024), and Qwen2.5-Math-7B/72B (Yang et al., 2024a), covering both MHA (Vaswani et al., 2017) and GQA (Ainslie et al., 2023) architectures.

Baselines and Implementation. We compare against Full Attention and training-free sparse baselines: Quest (Tang et al., 2024), TidalDecode (Yang et al., 2024b), and StreamingLLM (Xiao et al., 2023). Retrieval Heads are detected following Wu et al. (2024). Consistent with prior work, we keep

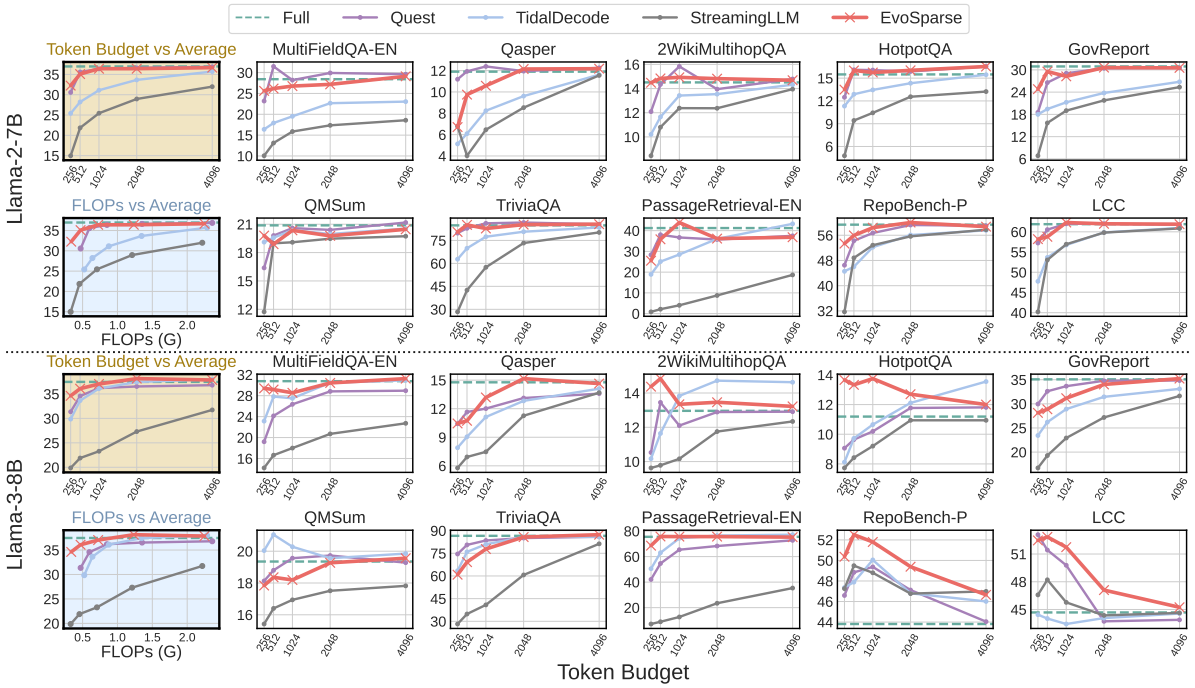

Figure 4: Evaluation on LongBench tasks under varying token budgets {256, 512, 1024, 2048, 4096}.

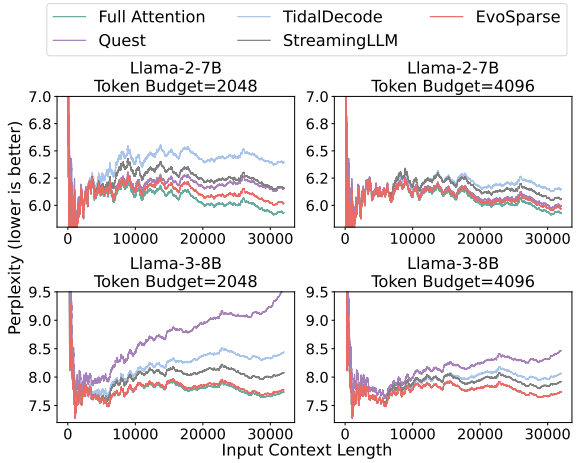

Figure 5: Perplexity of different methods across varying context lengths from 0 to 32k tokens.

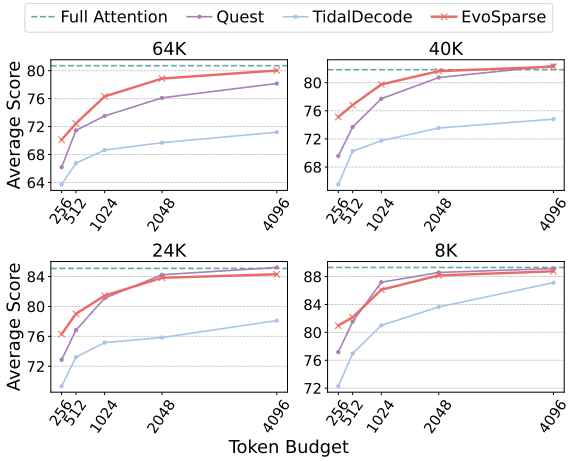

Figure 6: Performance comparison on the RULER benchmark across varying context lengths from 8k to 64k tokens.

the first two layers dense to preserve stability (Tang et al., 2024). Detailed hyperparameters are listed in Table 6.

5.2 Performance Evaluation

5.2.1 Language Modeling on PG-19

Figure 5 demonstrates that EvoSparse achieves perplexity (PPL) comparable to Full Attention across diverse models and token budgets. In contrast, baselines exhibit distinct failure modes: Quest suffers from noise introduced by ineffective heads; TidalDecode shows instability across architectures (dropping on LLaMA-2 due to layer mismatch); and StreamingLLM degrades consistently due to indiscriminate token dropping. EvoSparse avoids these pitfalls, preserving modeling quality even under constrained conditions.

5.2.2 Long-context Robustness on RULER

We evaluate retrieval robustness on RULER (Hsieh et al., 2024) using Llama-3-8B-1M (Pekelis et al., 2024) across contexts ranging from 8K to 64K. As illustrated in Figure 6, EvoSparse demonstrates superior resilience compared to sparse baselines. In high-sparsity regimes, our method maintains a significant lead, validating the efficacy of Cross-Layer Propagation in accurately pinpointing critical

Method	AIME 24	AMC 23	SAT Math	CN Middle School	Gaokao Cloze	Avg.
Avg Length	*1451.7*	*1162.0*	*2311.4*	*1981.6*	*974.0*	*1576.14*
Full Attention	16.7	65.0	96.9	72.3	68.6	63.90
StreamingLLM (512)	3.3	27.5	84.4	72.3	57.6	49.02
Quest (512)	10.0	45.0	90.6	69.3	59.3	54.84
TidalDecode (512)	10.0	52.5	87.5	73.3	66.9	58.04
EvoSparse (512)	13.3	55.0	90.6	72.3	69.5	**60.14**
StreamingLLM (256)	3.3	10.0	43.8	52.5	27.1	27.34
Quest (256)	3.3	35.0	75.0	66.3	44.1	44.74
TidalDecode (256)	3.3	40.0	71.9	69.3	63.6	49.62
EvoSparse (256)	6.7	45.0	78.1	64.4	67.8	**52.40**

Table 1: Performance comparison on mathematical reasoning tasks using Qwen2.5-Math-7B (Yang et al., 2024a).

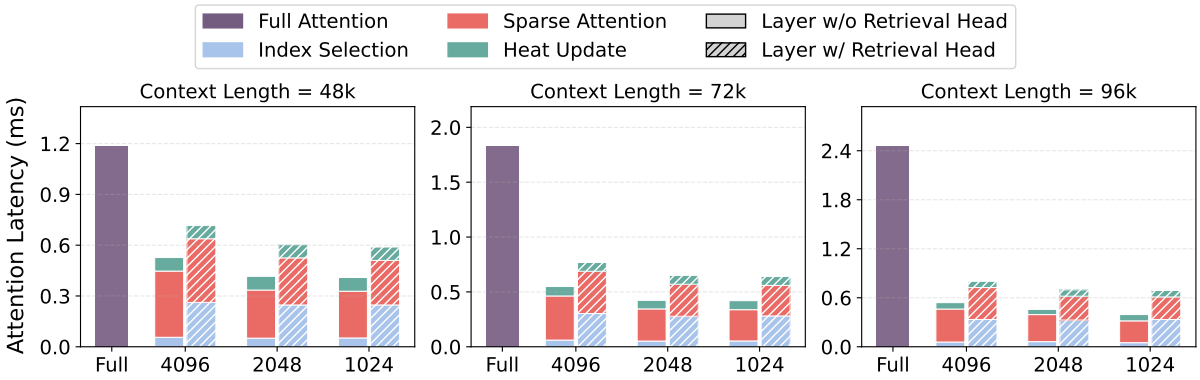

Figure 7: Latency breakdown of a single attention layer.

evidence amidst massive noise. In low-sparsity regimes, it rapidly converges to the Full Attention upper bound, achieving near-lossless performance as the budget increases.

5.2.3 General Long-Context Capabilities on LongBench

Figure 4 summarizes results across 10 representative LongBench tasks (Bai et al., 2023). EvoSparse achieves the best average performance across both Llama-2 and Llama-3 models. At higher budgets, it matches the accuracy of Full Attention; at lower budgets, it substantially outperforms all baselines. While Quest struggles with small budgets and TidalDecode relies on architecture-specific tuning, our approach maintains strong retrieval accuracy across different models and sparsity levels, highlighting its generalization capability.

5.2.4 Long-Chain Reasoning on Mathematical Benchmarks

Distinct from retrieval tasks, mathematical reasoning necessitates maintaining a coherent Chain of Thought (CoT) throughout extensive generation. The critical challenge lies in preserving self-generated intermediate steps within a limited cache. As shown in Table 1, evaluations on Qwen2.5-Math-7B (Yang et al., 2024a) demonstrate that EvoSparse achieves superior accuracy under strict token budgets compared to baseline methods. This result stems from Cross-Step Accumulation, which anchors pivotal intermediate conclusions that instantaneous scoring often overlooks, thereby preventing the logic chain breakage common in prior sparse attention mechanisms.

5.3 Efficiency Evaluation

We benchmark EvoSparse on Llama-3-8B-1M (Pekelis et al., 2024) using a single NVIDIA RTX 5090 (BF16). We implement core operators via Triton (Tillet et al., 2019) and compare against a PyTorch SDPA baseline. Figures 7 and 9 report attention and end-to-end latencies, respectively. Under a 96K context with a 2048-token budget, EvoSparse delivers significant acceleration. Standard sparse layers achieve a 5.36× speedup relative to full attention. Even layers containing Retrieval Heads maintain a 3.52× speedup despite selection overhead. Since Retrieval Heads constitute a minority

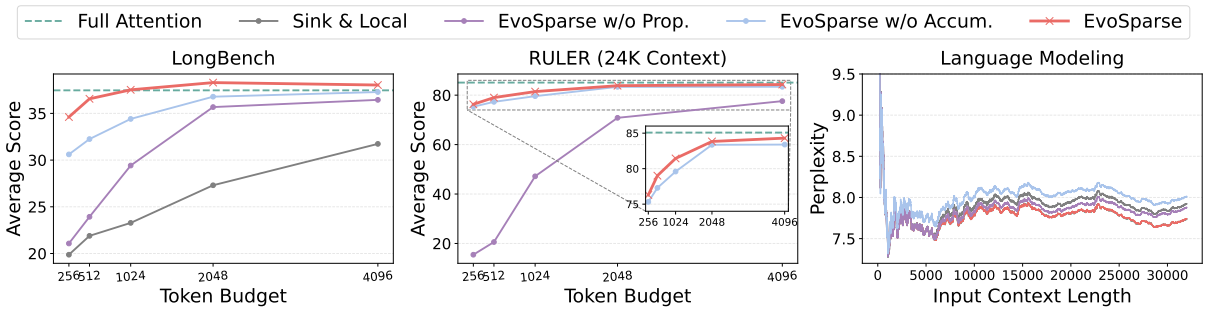

Figure 8: Ablation study of EvoSparse components on Llama-3-8B-1M (Pekelis et al., 2024).

Method	AIME 24	AMC 23	SAT Math	CN Middle	Gaokao Cloze	Avg.
Sink & Local (16 + 496)	3.3	27.5	84.4	72.3	57.6	49.02
EvoSparse (512, w/o Prop.)	3.3	40.0	75.0	70.3	58.5	49.42
EvoSparse (512, w/o Accum.)	3.3	50.0	71.9	67.3	61.9	50.88
EvoSparse (512)	13.3	55.0	90.6	72.3	69.5	**60.14**
Sink & Local (16 + 240)	3.3	10.0	43.8	52.5	27.1	27.34
EvoSparse (256, w/o Prop.)	3.3	20.0	46.9	56.4	46.6	34.64
EvoSparse (256, w/o Accum.)	3.3	47.5	56.2	65.3	61.9	46.84
EvoSparse (256)	6.7	45.0	78.1	64.4	67.8	**52.40**

Table 2: Ablation study of EvoSparse components on Qwen2.5-Math-7B (Yang et al., 2024a).

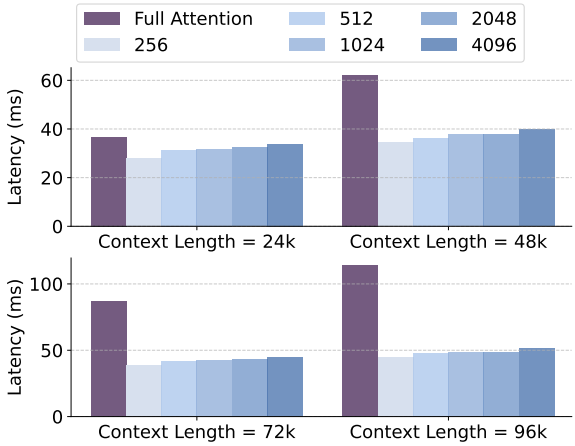

Figure 9: End-to-end decoding latency of EvoSparse.

of the model, these gains translate into a system-level end-to-end decoding speedup of 2.33×.

Scaling with Batch Size. Beyond single-batch performance, we further evaluate the effectiveness of EvoSparse in batched inference scenarios. As illustrated in Figure 10, EvoSparse consistently maintains high efficiency, and its relative advantage over full attention becomes even more pronounced as the batch size increases. For instance, with a 1024-token budget, the latency reduction compared to full attention grows from approximately 13.3% at BS=1 (from 28.34ms to 24.56ms) to 50.8% at BS=4 (from 19.62ms to 9.66ms). This trend suggests that by significantly reducing KV cache access, EvoSparse effectively alleviates the memory bandwidth bottleneck that typically intensifies with larger batch sizes, making it highly suitable for high-throughput deployment.

5.4 Ablation Study

We evaluate the impact of the two components of EvoSparse, namely Cross-Step Accumulation (Accum.) and Cross-Layer Propagation (Prop.), across language modeling, retrieval, and reasoning tasks.

Impact of Cross-Layer Propagation. The Prop. mechanism is indispensable for tasks requiring precise information extraction. Removing it causes catastrophic failures in low-budget regimes. On LongBench (Figure 8, Left) and RULER (Figure 8, Middle), removing Prop. results in a substantial performance gap compared to the full model. Similarly, on Mathematical Reasoning (Table 2), performance plummets significantly (e.g., Math accuracy drops from 52.4% to 34.6% at budget 256). This consistent degradation across benchmarks confirms that without explicit expert guidance, Standard Heads fail to independently locate sparse evidence within noisy contexts.

Impact of Cross-Step Accumulation. In contrast, Accum. acts as the primary driver for general coherence. Removing it leads to the sharpest degradation in PG-19 perplexity (Figure 8, Right), whereas removing Prop. has a comparatively minor effect.

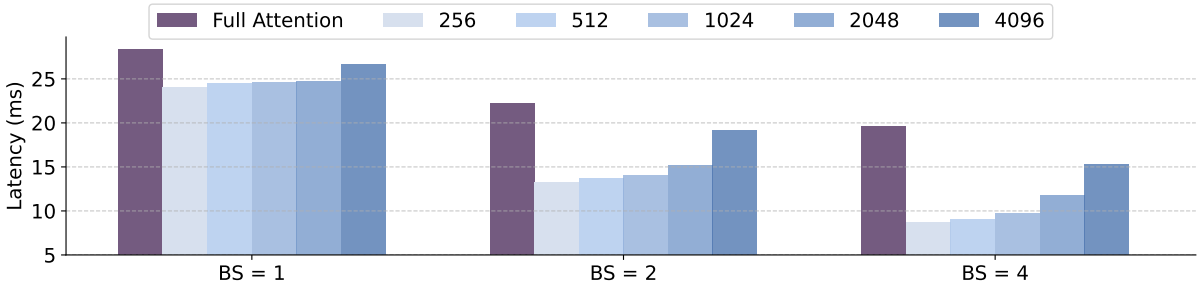

Figure 10: End-to-end decoding latency across different batch sizes, evaluated at a 16K context length.

Method	AIME 24	AMC 23	SAT Math	CN Middle	Gaokao Cloze	Avg.
Full Attention	23.3	62.5	96.9	78.2	74.6	67.10
Quest (512)	3.3	50.0	65.6	69.3	56.8	49.00
TidalDecode (512)†	20.0	67.5	87.5	77.2	71.2	64.68
EvoSparse (512)	20.0	67.5	96.9	79.2	72.0	**67.12**

† For TidalDecode, we report the best performance (Layer 33, ∼40% depth) via a grid search for the reselection layer.

Table 3: Performance comparison on mathematical reasoning tasks using Qwen2.5-Math-72B (Yang et al., 2024a).

This suggests that while Prop. locates specific needles, Accum. maintains the broader global context stability essential for language modeling.

5.5 Scalability to Larger Model Scales

To further investigate the scalability of EvoSparse and the impact of model capacity on sparse attention, we conduct additional experiments on the Qwen2.5-Math-72B (Yang et al., 2024a). As shown in Table 3, the performance gap previously observed at the 7B scale effectively vanishes at the 72B scale. EvoSparse achieves an average score of 67.12, which is on par with Full Attention (67.10).

This phenomenon suggests that the sensitivity to KV cache sparsity in smaller models (e.g., 7B) is likely due to their limited parameter redundancy. In contrast, larger models possess sufficient capacity to maintain strong robustness in long-chain reasoning tasks even with a constrained KV budget.

6 Conclusion

We presented EvoSparse, a unified framework that exploits information redundancy across layers and decoding steps. By integrating Cross-Layer Propagation for precision and Cross-Step Accumulation for stability, it effectively reconciles efficiency with performance. Evaluations across PG-19, LongBench, RULER, and reasoning tasks confirm that EvoSparse significantly outperforms baselines under strict constraints while matching full attention in high-budget regimes. Achieving up to 5.36× faster attention computation and 2.33× end-to-end speedup, it establishes a robust solution for long-context inference.

Limitations

Despite the promising results, we acknowledge several limitations. First, the reported wall-clock speedups rely on custom Triton kernels designed for efficient block-sparse execution; while these demonstrate significant gains on NVIDIA GPUs, deploying EvoSparse on hardware platforms that lack flexible sparse support or different memory hierarchies may require further engineering adaptation. Second, while we focus on MHA and GQA, EvoSparse's compatibility with Multi-Head Latent Attention (MLA)—as used in DeepSeek—remains unexplored. Integrating our sparsity approach with MLA to further reduce overhead is a key future direction.

Acknowledgments

Miao Zhang was partially sponsored by the National Natural Science Foundation of China under Grant 62306084 and U23B2051, Shenzhen College Stability Support Plan under Grant GXWD20231128102243003, and Shenzhen Science and Technology Program under Grant ZDSYS20230626091203008 and KJZD20230923115113026. The work of Ziyue Qiao is partially supported by the National Natural Science Foundation of China (No. 62406056), the Guangdong Basic and Applied Basic Research Foundation (No. 2024A1515140114)

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*.

Chenxin An, Shansan Gong, Ming Zhong, Xingjian Zhao, Mukai Li, Jun Zhang, Lingpeng Kong, and Xipeng Qiu. 2023. L-eval: Instituting standardized evaluation for long context language models. *arXiv preprint arXiv:2307.11088*.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, and 1 others. 2023. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2023. Longlora: Efficient fine-tuning of long-context large language models. *arXiv preprint arXiv:2309.12307*.

Yichuan Deng, Zhao Song, and Chiwun Yang. 2024. Attention is naturally sparse with gaussian distributed input. *CoRR*.

Yao Fu. 2024. Challenges in deploying long-context transformers: A theoretical peak performance analysis. *arXiv preprint arXiv:2405.08944*.

Yao Fu, Rameswar Panda, Xinyao Niu, Xiang Yue, Hannaneh Hajishirzi, Yoon Kim, and Hao Peng. 2024. Data engineering for scaling language models to 128k context. *arXiv preprint arXiv:2402.10171*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. 2024. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*.

Greg Kamradt. 2023. Llmtest_needleinahaystack: Doing simple retrieval from llm models at various context lengths to measure accuracy.

Amirkeivan Mohtashami and Martin Jaggi. 2023. Landmark attention: Random-access infinite context length for transformers. *arXiv preprint arXiv:2305.16300*.

Leonid Pekelis, Michael Feil, Forrest Moret, Mark Huang, and Tiffany Peng. 2024. Llama 3 gradient: A series of long context models.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2023. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*.

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. 2019. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*.

Luka Ribar, Ivan Chelombiev, Luke Hudlass-Galley, Charlie Blake, Carlo Luschi, and Douglas Orr. 2023. Sparq attention: Bandwidth-efficient llm inference. *arXiv preprint arXiv:2312.04985*.

Uri Shaham, Maor Ivgi, Avia Efrat, Jonathan Berant, and Omer Levy. 2023. Zeroscrolls: A zero-shot benchmark for long text understanding. *arXiv preprint arXiv:2305.14196*.

Prajwal Singhania, Siddharth Singh, Shwai He, Soheil Feizi, and Abhinav Bhatele. 2024. Loki: Low-rank keys for efficient sparse attention. *Advances in Neural Information Processing Systems*, 37:16692–16723.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. 2024. Quest: Query-aware sparsity for efficient long-context llm inference. *arXiv preprint arXiv:2406.10774*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Philippe Tillet, Hsiang-Tsung Kung, and David Cox. 2019. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pages 10–19.

Szymon Tworkowski, Konrad Staniszewski, Mikołaj Pacek, Yuhuai Wu, Henryk Michalewski, and Piotr Miłoś. 2023. Focused transformer: Contrastive training for context scaling. *Advances in neural information processing systems*, 36:42661–42688.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Wenhao Wu, Yizhong Wang, Guangxuan Xiao, Hao Peng, and Yao Fu. 2024. Retrieval head mechanistically explains long-context factuality. *arXiv preprint arXiv:2404.15574*.

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. 2024. Duoattention: Efficient long-context llm inference with retrieval and streaming heads. *arXiv preprint arXiv:2410.10819*.

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

Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, and 1 others. 2023. Effective long-context scaling of foundation models. *arXiv preprint arXiv:2309.16039*.

Peng Xu, Wei Ping, Xianchao Wu, Lawrence McAfee, Chen Zhu, Zihan Liu, Sandeep Subramanian, Evelina Bakhturina, Mohammad Shoeybi, and Bryan Catanzaro. 2023. Retrieval meets long context large language models. *arXiv preprint arXiv:2310.03025*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, and 1 others. 2024a. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

Lijie Yang, Zhihao Zhang, Zhuofu Chen, Zikun Li, and Zhihao Jia. 2024b. Tidaldecode: Fast and accurate llm decoding with position persistent sparse attention. *arXiv preprint arXiv:2410.05076*.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, and 1 others. 2024. Infinity bench: Extending long context evaluation beyond 100k tokens. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15262–15277.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, and 1 others. 2023. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710.

A Appendix

A.1 Hyperparameter Sensitivity

We investigate the sensitivity of EvoSparse to three critical hyperparameter groups: (1) the Prop. configuration (number of Retrieval Heads and top_k blocks), (2) the sparsity granularity (block size B), and (3) the temporal decay factor λ.

Impact of Prop. Configuration. Figure 1 visualizes the performance heatmaps varying the number of Retrieval Heads (N_{ret}) and the top_k KV blocks per retrieval head (k_{blk}). We observe a general trend of monotonic improvement: increasing either the number of Retrieval Heads or the number of retained blocks consistently boosts performance, indicated by the darker colors in the upper-right corners. However, the method exhibits strong robustness; for tasks like LongBench and RULER, acceptable performance is achieved even with moderate settings (e.g., $N_{ret} = 10$). Notably, for the Math Reasoning task (right), which operates under a strict budget, the sensitivity is higher. Reaching the peak accuracy of 60.14% requires a sufficient number of Retrieval Heads ($N_{ret} \geq 10$) to accurately pinpoint distributed evidence, confirming that "expert guidance" is resource-dependent in challenging reasoning scenarios.

Impact of Block Size. We analyze the effect of block size granularity (B) and observe distinct behaviors across different task types (Figure 2).

On LongBench and RULER (Figure 2 Left & Middle): For tasks requiring information retrieval and reasoning, a clear trade-off emerges regarding granularity. At strictly limited token budgets (e.g., 256 tokens), a smaller block size ($B = 16$, red line) significantly outperforms larger blocks ($B = 64$, blue line). This is because smaller blocks provide finer granularity, allowing the model to precisely retain discrete, scattered evidence ("needles") without wasting the scarce budget on irrelevant neighbors contained within larger blocks. As the budget increases (> 1024), this granularity advantage diminishes, and larger blocks ($B = 64$) catch up, becoming competitive due to their better capture of local semantic continuity.

On Language Modeling: In contrast, the block size has negligible impact on PG-19 perplexity. As shown in Figure 2 (Right), the performance curves for different block sizes largely overlap. This suggests that for next-token prediction, the primary factor is whether the global Long-term Anchors are retained, rather than the precise segmentation (block size) of these regions. The model is robust enough to extract necessary context regardless of whether the retrieved history is fragmented into 16-token or 64-token chunks.

Impact of Accum. Decay Factor (λ). We examine the role of the temporal decay factor λ in the

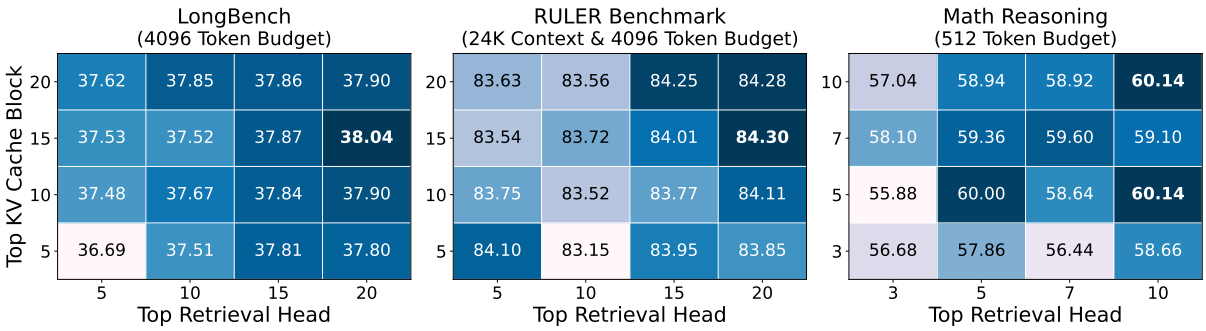

Figure 1: Hyperparameter sensitivity analysis of EvoSparse. We use Llama-3-8B-1M (Pekelis et al., 2024) for LongBench and RULER, and Qwen2.5-Math-7B (Yang et al., 2024a) for mathematical reasoning tasks.

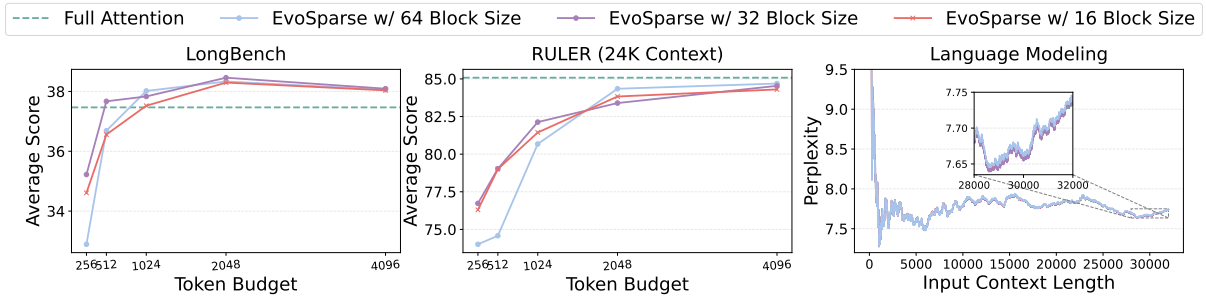

Figure 2: Hyperparameter sensitivity analysis of EvoSparse on block size using Llama-3-8B-1M (Pekelis et al., 2024).

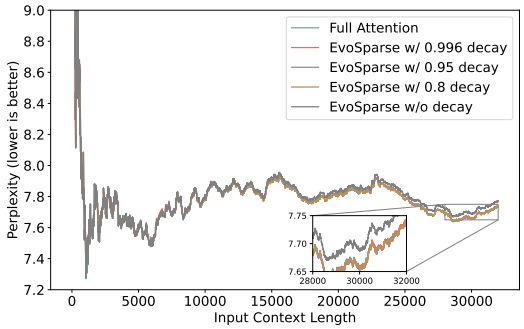

Figure 3: Hyperparameter sensitivity analysis of EvoSparse with different decay factors on Llama-3-8B-1M (Pekelis et al., 2024)

Cross-Step Accumulation mechanism (Table 4 and Figure 3). We specifically adopt $\lambda = 0.996$, which was chosen because it is the closest representable value to 1.0 under the BF16 precision format used in our experiments. This setting ensures the minimal necessary decay to prevent saturation while maximizing history retention.

The results indicate that the specific magnitude of λ (e.g., 0.8 vs. 0.996) has a marginal impact on performance, demonstrating the method's robustness. However, a crucial distinction exists between having decay ($\lambda < 1.0$) and no decay ($\lambda = 1.0$).

As shown in Table 4, removing the decay mechanism ($\lambda = 1.0$) leads to a noticeable performance drop (e.g., Avg accuracy drops from 60.14% to 56.14% on Math tasks). This degradation is further corroborated by the language modeling results in Figure 3. While models with decay (lines for $\lambda = 0.996, 0.95, 0.8$) tightly track the Full Attention baseline, the model without decay (grey line) exhibits a visible divergence, yielding higher perplexity as the context length grows.

We attribute these failures to a "Matthew Effect" in sparse selection. Without decay, the importance scores of early heavy-hitters accumulate indefinitely. These historical tokens eventually monopolize the limited sparse budget, preventing the model from attending to new, potentially critical information in the recent context. Introducing a decay factor effectively counteracts this saturation, allowing the model to update its focus and balance historical anchors with evolving context.

A.2 Hyperparameter Configuration

Table 6 details the specific settings for all evaluated methods. For EvoSparse, we employ a budget-aware scaling strategy, dynamically adjusting the number of active retrieval heads and retained blocks to optimize performance across varying sparsity levels.

Method	AIME 24	AMC 23	SAT Math	CN Middle School	Gaokao Cloze	Avg.
EvoSparse (512, $\lambda = 1.0$) *(w/o decay)*	6.7	47.5	84.4	75.2	66.9	56.14
EvoSparse (512, $\lambda = 0.8$)	6.7	52.5	93.8	74.3	67.8	59.02
EvoSparse (512, $\lambda = 0.95$)	6.7	50.0	90.6	75.2	69.5	58.40
EvoSparse (512, $\lambda = 0.996$)	13.3	55.0	90.6	72.3	69.5	60.14
EvoSparse (256, $\lambda = 1.0$) *(w/o decay)*	13.3	40.0	59.4	63.4	67.8	48.78
EvoSparse (256, $\lambda = 0.8$)	6.7	45.0	75.0	70.3	68.6	53.12
EvoSparse (256, $\lambda = 0.95$)	10.0	45.0	78.1	68.3	67.8	53.84
EvoSparse (256, $\lambda = 0.996$)	6.7	45.0	78.1	64.4	67.8	52.40

Table 4: Hyperparameter sensitivity analysis of EvoSparse with different decay factors on Qwen2.5-Math-7B (Yang et al., 2024a).

Method	AIME 24	AMC 23	SAT Math	CN Middle	Gaokao Cloze	Avg.
EvoSparse (512, $B = 16$)	13.3	55.0	90.6	72.3	69.5	60.14
EvoSparse (512, $B = 32$)	16.7	57.5	90.6	72.3	67.8	60.98
EvoSparse (512, $B = 64$)	3.3	45.0	84.4	72.3	68.6	54.72
EvoSparse (256, $B = 16$)	6.7	45.0	78.1	64.4	67.8	52.40
EvoSparse (256, $B = 32$)	3.3	47.5	59.4	55.4	55.9	44.30
EvoSparse (256, $B = 64$)	3.3	37.5	56.2	47.5	57.6	40.42

Table 5: Hyperparameter sensitivity analysis of EvoSparse on block size using Qwen2.5-Math-7B (Yang et al., 2024a).

Method	Budget	Local Window	Ret. Heads (N_{ret})	Top Blocks (k_{blk})	Specific Notes
EvoSparse	4096	128	20	15	-
	2048	128	15	10	-
	1024	128	10	7	-
	512	64	10	5	-
	256	64	5	5	-
Quest	All	-	-	-	Block Size = 16
TidalDecode	All	-	-	-	PPL: Re-sel. Layer = 9 Others†: Re-sel. Layer = 13
StreamingLLM	All	Budget - 16	-	-	Sink size = 16

Table 6: Hyperparameter settings for all methods. We report the configurations used for Language Modeling (PPL), LongBench, RULER, and Math Reasoning tasks. For EvoSparse, parameters scale with the token budget. Common Constants: Across all budgets, we fix Sink size = 16, Block size $B = 16$, and Decay factor $\lambda = 0.996$. N_{ret} denotes the number of selected top Retrieval Heads, and k_{blk} denotes the number of KV blocks retained per retrieval head.

† Others include LongBench, RULER, and Math tasks.