MATCH: Modulating Attention via In-Context Retrieval for Long-Context Transformers

Linrui Ma[♠*] **Chun Hei Lo**[♠*] **Xinyu Wang**[♣*] **Peng Lu**[♡] **Xihao Yuan**[◇]
Hanting Chen[◇] **Kai Han**[◇] **Xinghao Chen**[◇] **Chengjun Zhan**[◇] **Hanlin Xu**[◇]
Yichun Yin[◇] **Lifeng Shang**[◇] **Feng Wen**[♠] **Boxing Chen**[♠] **Yufei Cui**[♠†]

[♠] Huawei Canada [♣] McGill University [♡] Université de Montréal [◇] Huawei

Abstract

The quadratic computational cost of traditional attention mechanisms poses a major bottleneck to the scalability and practical deployment of large language models (LLMs), particularly in long-context scenarios. To improve efficiency, existing approaches often enforce rigid structural constraints such as local attention windows. However, these strategies typically lead to substantial performance degradation on tasks requiring precise long-range recall. In this work, we propose MATCH, a scalable and efficient framework that augments sparsified attention mechanisms with dynamically integrated in-context information through an efficient retrieval system. Empirical results show that MATCH significantly improves the performance of sparse-attention models on both synthetic and real-world natural-language tasks. These findings highlight the versatility of MATCH as a general approach for enhancing in-context retrieval capabilities while maintaining the efficiency benefits of sparse attention architectures.

1 Introduction

Large language models (LLMs) have demonstrated remarkable capabilities across a wide range of natural language processing tasks (OpenAI, 2023; Dubey et al., 2024; Abdin et al., 2024; DeepSeek-AI, 2024; Yang et al., 2025). Yet their scalability is fundamentally constrained by the *quadratic* computational complexity of the self-attention mechanism. This bottleneck becomes particularly acute when the context length is extended, as both memory and computation costs grow rapidly with the sequence length (Kwon et al., 2023; Liu et al., 2025). Consequently, practical deployments of LLMs are often forced to operate with truncated contexts, limiting their ability to fully exploit long-range dependencies in data. Numerous methods were developed in the hope of retaining inference efficiency and competitive performance as full attention while having light-weight memory consumption (Zhang et al., 2023b; Liu et al., 2023; Xiao et al., 2024b; Han et al., 2024; Li et al., 2024).

However, it is hard to reduce the memory footprint of the KV cache without sacrificing performance on challenging tasks. A common line of work seeks to mitigate this limitation by replacing full attention with sparse attention mechanisms (e.g., sliding window attention (Beltagy et al., 2020)) or generalized linear attention (Mamba (Gu and Dao, 2024; Dao and Gu, 2024)) to reduce computational and memory overhead from quadratic to linear or sub-quadratic complexity. While effective in enhancing computational efficiency, their local window patterns or fixed-size recurrent state significantly constrain the model's capacity to retrieve and integrate pertinent information across distant positions within a sequence (Arora et al., 2024; Xiao, 2025). Consequently, such limitations often lead to diminished performances on tasks that demand precise recall of contextual details.

In this work, we address this trade-off between efficiency and recall by augmenting sparse-attention LLMs with an **external retriever** designed to enhance long-range recalling capabilities. This retriever acts as a complementary module that processes the entire context in a computationally efficient manner, encoding salient information and making it accessible to the sparsified LLM. By integrating the retriever's outputs into the sparse-attention layers, our method preserves the efficiency benefits of sparsity while substantially improving the model's capacity to retrieve and reason over distant information.

Our contribution can be summarized as follows: First, we introduce a novel architecture that pairs sparse attention with an external retrieval-enhancing encoder and propose a retrieval-enhanced attention. Then, we conducted extensive

*Equal Contribution
†Corresponding author: yufei.cui@huawei.com

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 15165–15179

experiments on both synthetic ICL tasks and real-world long-context benchmarks and demonstrate substantial improvements over baseline sparse-attention models. Finally, we perform a comprehensive analysis of its efficiency, including throughput and memory footprint.

2 Related Work

Retrieval-Augmented Language Models. Retrieval-augmented language models enhance performance on knowledge-intensive tasks by incorporating a dedicated retrieval module that sources relevant textual information from an external knowledge base (Karpukhin et al., 2020; Guu et al., 2020; Lewis et al., 2020). This retrieved content is then combined with the original input and passed to the main model, allowing it to access information beyond its parametric memory (Roberts et al., 2020). Izacard et al. (2023) conduct a deeper analysis of how different loss functions influence retrieval module performance. Subsequent advancements have focused on refining passage selection (Asai et al., 2024; Ma et al., 2025), improving resilience to irrelevant or noisy retrievals (Yoran et al., 2024; Xu et al., 2024), and optimizing additional components of the retrieval pipeline (Lin et al., 2024).

Sparse Attention. Given the quadratic computational complexity of standard attention, sparse attention is chosen as a strategy to improve Transformer efficiency. Static sparse patterns include methods such as sliding window attention (SWA), dilated attention (Child et al., 2019; Shi et al., 2021; Ding et al., 2023), and other fixed sparsity schemes. SWA mechanisms are widely adopted in many modern large language model families, including Mistral (Jiang et al., 2023), Phi-3 (Abdin et al., 2024), Hymba (Dong et al., 2025), Gemma-3 (Team et al., 2025), and GPT-oss (Agarwal et al., 2025). Despite their efficiency gains, these methods typically have a limited receptive field and impose significant constraints on the flexibility of attention, particularly in attending to arbitrary token positions, thus underperforming full attention on copy-heavy tasks (Xiao, 2025).

KV Cache Compression. Recent research aim to improve the inference efficiency by reducing the KV cache usage during LLM decoding. SnapKV (Li et al., 2024) curates the KV cache by monitoring the accumulated attention scores and select significant tokens. Liu et al. (2023) propose selective dropping of low-attention KV pairs, while Liu et al. (2024) introduce quantization techniques for compact KV representations. H_2O (Zhang et al., 2023b) provides an adaptive token eviction strategy, improving the memory usage by balancing the recent and distant information. StreamingLLM (Xiao et al., 2024b) investigates and emphasize the inherent patterns of pre-trained attention, e.g. attention-sinks. By keeping the attention-sinks and local window patterns, it expands the context size of LLM on super-long inputs. PyramidKV (Cai et al., 2024) dynamically adjusts the KV cache consumption across different layers, which allocates more budgets for lower layers while tightening for upper layers. In this work, we focus on enhancing sparse attention for in-context retrieval, while KV cache compression addresses post-hoc efficiency in memory storage and reuse. Consequently, the two approaches are orthogonal: our method operates independently of KV cache compression and can be seamlessly combined with it for further efficiency gains.

3 Problem Formulation and Overview

This work seeks to boost the in-context recall capabilities of LLMs, with a particular emphasis on improving their performance in long-context tasks under constrained memory conditions. LLM inference generally comprises two stages: *pre-filling* and *decoding*. In the pre-filling stage, the model takes the user prompt as input and processes it in parallel to generate the initial hidden states.

We denote the user prompt as a sequence $x = (x_1, \ldots, x_N)$ and a model with a hidden dimension d. For the l-th attention layer, we utilize the trained model weights $\mathbf{W}_Q^l, \mathbf{W}_K^l, \mathbf{W}_V^l \in \mathbb{R}^{d\times d}$ for the query, key, and value projection matrices. In the standard *Full Attention* (FA) mechanism, every token x_i must attend to all preceding tokens in the sequence to capture global dependencies. The query, key, and value projections for a token at position i are computed as:

$$[\mathbf{q}_i, \mathbf{k}_i, \mathbf{v}_i] = \mathbf{h}_i^{l-1}[\mathbf{W}_Q^l, \mathbf{W}_K^l, \mathbf{W}_V^l], \quad (1)$$

where $\mathbf{h}_i^{l-1} \in \mathbb{R}^d$ is the input hidden state. The attention output $\mathbf{o}_i$ is the scaled dot-product across the entire sequence length N:

$$\mathbf{o}_i = \text{softmax}\left(\frac{\mathbf{q}_i(\mathbf{K}^l)^\top}{\sqrt{d_k}}\right)\mathbf{V}^l. \quad (2)$$

A primary challenge in deploying LLMs for long-context applications is the quadratic complexity of this operation. Because the softmax attention must be computed over all N tokens for each of the N positions, the computational requirements and memory consumption scale at $O(N^2)$. This memory bottleneck limits the effective context window size as the input length increases.

3.1 LLMs with Pre-Sparsified and Post-Sparsified Attentions

In order to balance the high performance and efficiency of LLMs, many works (e.g., Jiang et al. (2023); Abdin et al. (2024); Dong et al. (2025); Team et al. (2025); Agarwal et al. (2025)) replace full attention in the token-mixing layer with sparse attention (Beltagy et al., 2020). These models, referred to as *pre-sparsified* LLMs, are pre-trained with sliding window attention used in all or parts of the token-mixing layers. There is another line of work that focuses on *post-sparsified* LLMs with full attention during pre-training by identifying the inherent structures of the attention layers and evict unimportant tokens in the KV cache (Zhang et al., 2023b; Liu et al., 2023; Xiao et al., 2024b; Han et al., 2024; Li et al., 2024; Xiao et al., 2024a). Both the pre-sparsified and post-sparsified LLMs can work with a manageable KV cache buffer with a limited size, which is crucial to provide affordable, efficient and high-performing services for a wide range of real-world applications.

Although many works show that the sparse characteristics of attention matrices widely exist in pre-trained LLMs (Liu et al., 2023; Xiao et al., 2024b; Han et al., 2024), the sparse patterns are highly context-dependent. Namely, many tokens attend only to parts of the input sequence, and on which specific tokens it focuses is highly dependent on the surrounding context, which often differs markedly across various inputs. This context-sensitive structure is a crucial feature of models' behavior that enhances in-context recall. In such settings, the ability to selectively retrieve relevant information from long or complex sequences hinges on the model's capacity to dynamically adapt its attention based on nuanced contextual cues. Without this flexibility, the model risks overlooking key dependencies for accurate understanding and generation.

4 MATCH: Improving Sparsified Attention via External Retriever

In this section, we introduce our method, MATCH, designed to enhance the in-context recall capability of sparse attention mechanisms by incorporating a pre-trained external retriever.

At its core, MATCH augments a sparse attention model with dynamically generated token positions that are useful to next-token generation. These positions are identified via a retrieval system, which is detailed in §4.1. This allows the model to directly access and integrate information from arbitrary positions in the sequence, effectively extending its receptive field far beyond the constraints of local attention windows. This is particularly important for tasks that involve very long inputs, where maintaining both efficiency and global contextual information remains a significant challenge.

In §4.2, we demonstrate how the information from retrieval can be integrated into the attention computations. Essentially, we sparsify the attentions based on the retrieved positions and the original sparse patterns. With chunked pre-filling and limited KV recomputations during decoding, we can achieve memory- and time- efficient sub-quadratic attention computations.

4.1 In-Context Dense Search

MATCH incorporates an in-context search module that leverages an external retrieval system to dynamically identify the most salient context for each input token. This module should be designed so that it does not introduce high latency and memory overhead during generation. Below, we briefly describe the module.

Given an input sequence $x = (x_1, \ldots, x_N)$, we first partition the sequence into fixed-size context chunks of length U, which constitute the candidate pool for retrieval. For each token position, we construct a *query chunk* consisting of the immediate preceding tokens to provide a localized semantic representation. During pre-filling, we retrieve the top-k chunks using a bi-encoder over dense embeddings produced by a Sentence-BERT model (Reimers and Gurevych, 2019). Bi-encoders are fast and ensures low-latency pre-filling. During decoding, we retrieve the top-K chunks I' using the bi-encoder, rerank them using a cross-encoder, and select only the top-k chunks, where $K > k$. Fig. 1 illustrates the idea. While cross-encoders capture token-level interactions and generally offer higher

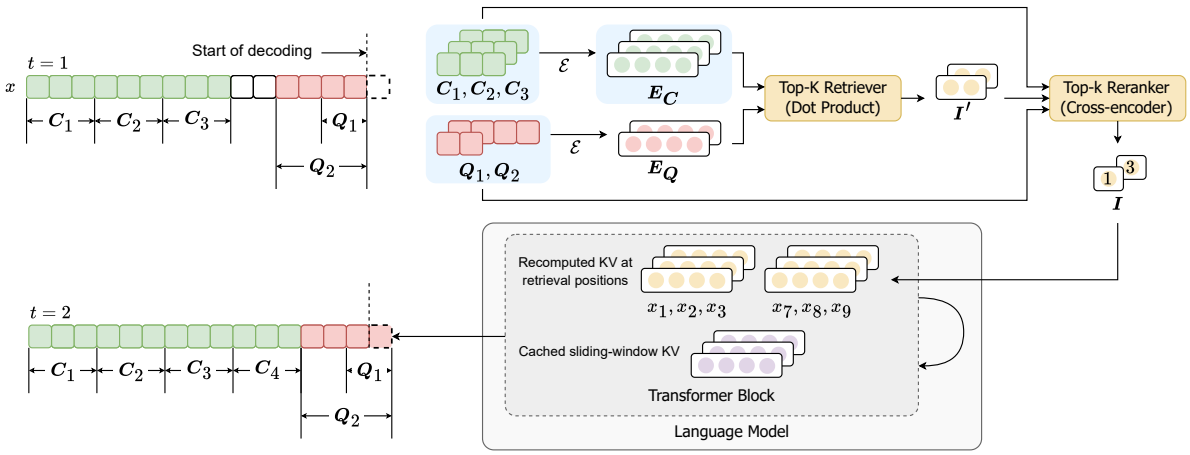

Figure 1: Illustration of one decoding step using MATCH, where $U = 3$, $K = 4$, and $k = 2$. $\boldsymbol{C}$ and $\boldsymbol{Q}$ are context chunks and query chunks respectively. $\mathcal{E}$ is a Sentence-BERT encoder. Objects with a blue background can be cached in memory and used for recomputation during the generation of each token.

retrieval quality, this hybrid pipeline helps balance precision and speed. For techniques on further improving efficiency and more discussion about the module, see Appendix A and Appendix C.1.

4.2 Augmented Attention Computation

We incorporate the retrieved results into the attention computation by unmasking the corresponding token positions, allowing each query token to attend not only to its causal local window but also to its retrieved tokens. As a result, we only need to maintain a constant-sized KV cache for the original sparse attention pattern. Moreover, the number of retrieved token positions remains fixed regardless of the sequence length. Therefore, the memory usage of sparse attention layers stays constant with respect to the overall context length, enabling better scalability to extremely long input sequences.

During inference, our method uses token positions identified from retrieval, which can be out of the sliding window of sparsified models, whose parameters may not be conditioned on during pre-training. Therefore, we adopt continual training on the models to adapt their parameters (see Appendix B.1). Below, we describe the detailed procedures of attention computation in pre-filling and decoding, as shown in Fig. 2.

4.2.1 Chunkwise Pre-Filling with Retrieval Information

When processing long sequential inputs, it is a common practice to use chunk-wise pre-filling, specifically, the input prompt is partitioned into fixed-size chunks to pre-fill the KV cache.

Concretely, given an original input sequence of length N, we pad it and divide it into n fixed-size chunks. During the pre-filling stage, each chunk is processed sequentially: the model computes and stores its corresponding key–value representations in the cache before proceeding to the next chunk.

During the pre-filling stage with long sequential inputs, for every query token we only need to conduct the attention operation with keys indicated by the custom attention mask matrices $\boldsymbol{\Lambda}$ as the following formulation:

$$\begin{bmatrix}\mathbf{Q} & \mathbf{K} & \mathbf{V}\end{bmatrix} = \begin{bmatrix}\mathbf{H}\end{bmatrix}\begin{bmatrix}\mathbf{W}_Q & \mathbf{W}_K & \mathbf{W}_V\end{bmatrix}, \quad (3)$$

$$\mathbf{S}_\mathbf{h}^\mathbf{l} = \mathrm{Softmax}\left(\mathbf{Q}\begin{bmatrix}\mathbf{K}^\top, \mathbf{K}_c^\top\end{bmatrix} + \boldsymbol{\Lambda}\right)\begin{bmatrix}\mathbf{V} \\ \mathbf{V}_c\end{bmatrix}, \quad (4)$$

where $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N \times d_{\text{head}}}$ are the query, key and value projections of one attention head, $(\mathbf{K}_c, \mathbf{V}_c)$ are cached key–value pairs and $\boldsymbol{\Lambda} \in \mathbb{R}^{\bar{N} \times \bar{N}}$ is the attention mask matrix, where $\bar{N} = n + n_c + n_r$, n_c is the size of cached KV, n_r is the number of retrieved KV. In our work, the attention mask $\boldsymbol{\Lambda}$ consists of three components: (1) a causal mask, which guarantees no future information leakages; (2) a structured mask for SWA, which provides local information and stability; and (3) a retrieval mask, which provides relevant information from arbitrary positions in the context.

4.2.2 Decoding with Retrieved KV Projection Recomputation

During each decoding step, the input sequence length is 1. The retriever identifies the correspond-

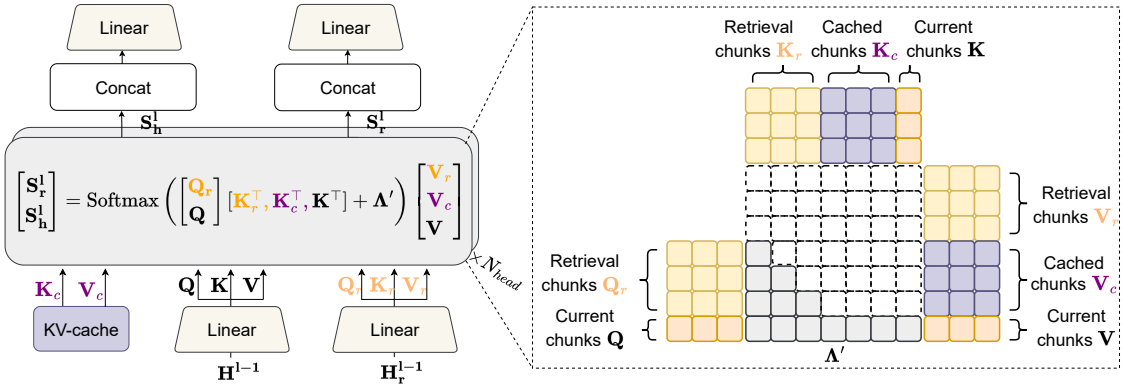

Figure 2: Illustration of the computation of MATCH in one attention head during decoding.

ing position from a compact cache of original inputs, and the associated original raw input is then provided to the model as an *auxiliary input*. In the subsequent forward pass, this auxiliary input participates in the attention computation synchronously with the current token and serves as the *reconstructed* KV cache for its offseted position, as shown on the right hand side of Fig. 2. Specifically, in each layer of the attention computation, we have:

$$\begin{bmatrix} \mathbf{Q}_r & \mathbf{K}_r & \mathbf{V}_r \\ \mathbf{Q} & \mathbf{K} & \mathbf{V} \end{bmatrix} = \begin{bmatrix} \mathbf{H}_r \\ \mathbf{H} \end{bmatrix} \begin{bmatrix} \mathbf{W}_Q & \mathbf{W}_K & \mathbf{W}_V \end{bmatrix}, \quad (5)$$

$$\begin{bmatrix} \mathbf{S}_\mathbf{r}^\mathbf{l} \\ \mathbf{S}_\mathbf{h}^\mathbf{l} \end{bmatrix} = \text{Softmax}\left(\begin{bmatrix} \mathbf{Q}_\mathbf{r} \\ \mathbf{Q} \end{bmatrix} \left[\mathbf{K}_r^\top, \mathbf{K}_c^\top, \mathbf{K}^\top \right] + \mathbf{\Lambda}' \right) \begin{bmatrix} \mathbf{V}_r \\ \mathbf{V}_c \\ \mathbf{V} \end{bmatrix}, \quad (6)$$

where $\mathbf{\Lambda}'$ denotes the attention mask, shaped as illustrated on the right side of Fig. 2, and $\mathbf{H}, \mathbf{H}_r$ represent the hidden states corresponding to the main and auxiliary inputs, respectively.

By re-computing the approximate attention over the auxiliary inputs at each layer, we efficiently while effectively reconstruct the KV cache for past inputs of arbitrary depth, enabling the model to recover long-range dependencies that would otherwise be lost due to cache truncation.

Crucially, this resource-intensive KV recomputation is performed only once at the beginning of each decoding chunk. The resulting reconstructed KVs at each layer are then stored in a fixed-size temporary cache. As a result, all subsequent new-coming query tokens within the same chunk can directly reuse this cache without repeating the expensive re-computation, thereby ensuring both the efficiency and effectiveness of MATCH .

This mechanism allows the retrieval-biased sparse attention to dynamically integrate relevant information that was not originally present in its limited cache, thereby enhancing both contextual coherence and overall performance in long-context scenarios.

5 Experiments and Results

We evaluate MATCH's performance on both synthetic and real-world long-context benchmarks, including Multi-Query Associative Recall (MQAR) (Arora et al., 2024), Mechanistic Architecture Design (MAD) (Poli et al., 2024), LongBench (Bai et al., 2024), and Needle-in-a-Haystack (NIAH) (Ivgi et al., 2023). We follow the standard setup of models for MQAR and MAD. For LongBench and NIAH, we experiment with a post-sparsified Qwen3-8B-Base (Yang et al., 2025) and the pre-sparsified Phi-3-mini-4k-instruct (Abdin et al., 2024).

Retrieval Configurations. For LongBench and NIAH, we use all-MiniLM-L6-v2 (Wang et al., 2020) which comprises only 22.6M parameters as the embedding model, and bge-reranker-v2-m3 (Li et al., 2023; Chen et al., 2024) which comprises 568M parameters as the reranker. We set $U = 128$, $K = 64$, and $k \in \{4, 8\}$. For MQAR and MAD, since the data is not natural language, we simply use exact string matching as the retrieval method, and set $U \in \{1, 2, 4\}$, and $K = k = 1$.

5.1 Experiments on Synthetic Tasks

We first validated the effectiveness of the MATCH architecture on recall-intensive tasks using MQAR and MAD benchmarks. For details of the setup of synthetic experiments, see Appendix B.3.

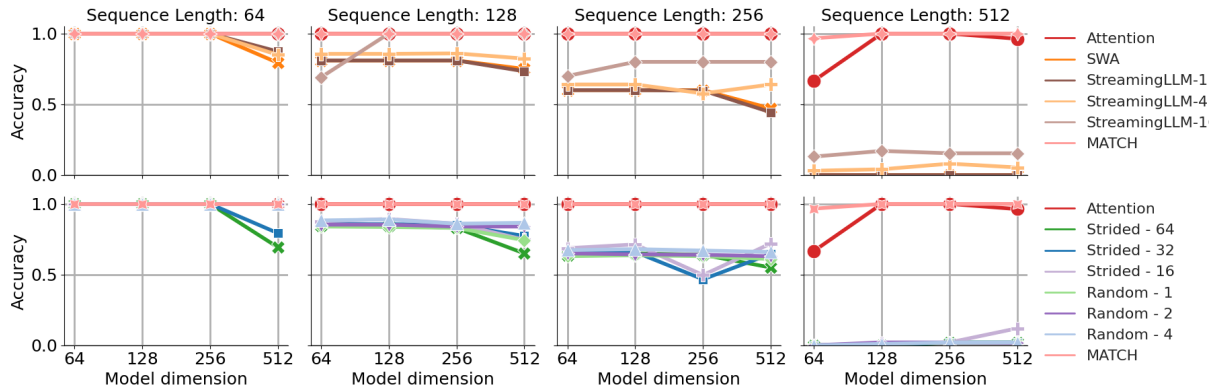

Figure 3: Results on MQAR. The top and bottom rows share identical experimental settings, differing only in the subjects being compared. Larger sequence lengths correspond to increased task difficulty.

Model	U	Fuzzy ICR	ICR	Noisy ICR	Avg.
Hymba		13.8	91.0	88.9	64.0
Hymba w/ MATCH	2	**80.3**	**99.0**	98.0	**92.4**
Hymba w/ MATCH	4	72.5	98.7	**98.1**	89.8
SWA		10.3	86.2	83.5	59.9
SWA w/ MATCH	2	**72.6**	**99.0**	**96.6**	**89.4**
SWA w/ MATCH	4	55.5	98.9	95.1	83.2

Table 1: Performance on three in-context retrieval tasks in MAD (Poli et al., 2024).

Multi-Query Associative Recall (MQAR). MQAR tests the associative recall capabilities of models. Fig. 3 compares MATCH against StreamingLLM, Strided (Zhang et al., 2023a), and Random, where the suffixes denote sink width, stride intervals, and random activation count respectively for the three methods. Despite requiring fewer activations, MATCH matches the performance of full attention and significantly outperforms other methods by a huge margin. Notably, at the sequence length of 512, MATCH maintains near-perfect accuracy whereas other sparse baselines almost degrade to random guessing.

Mechanistic Architecture Design (MAD). We conducted experiments on three synthetic in-context recall tasks proposed in the MAD suite. We evaluate two model types: Hymba model (Dong et al., 2025) and a transformer equipped with SWA. Table 1 shows that our proposed MATCH consistently boosts in-context retrieval (ICR) performance across all tasks and model variants. For both Hymba-based and SWA-based model, MATCH boosts the average performance by 45% to 50%, these gains hold across all three ICR tasks, with MATCH achieving the best or near-best performance in nearly all metrics. Overall, these results highlight the effectiveness of MATCH in context-recall–intensive scenarios.

5.2 Experiments on LongBench

Table 2 presents a comprehensive comparison between the performance of the sparsified baseline models with and without MATCH and the best-performing RAG results (see Appendix B.5) on the diverse downstream tasks of on LongBench. First, across all sparsity levels and model types, MATCH consistently improves overall average performance. The most pronounced improvements occur on synthetic tasks (up to +15.1), demonstrating that MATCH effectively recovers task-specific reasoning ability that is typically degraded by sparsification. Second, we also compare MATCH against retrieval-augmented generation (RAG) that uses the same retrieved results. Table 2 reports The results show that MATCH is mostly on par with or better than RAG, underpinning how MATCH can be a better alternative to incorporate retrieved contexts than RAG.

When $k = 4$, MATCH performs equally well with $k = 8$. The ablation studies presented in §6.1 show that performance drastically deteriorates under random retrieval. This shows that the retriever has high recall.

5.3 Experiments on NIAH

Table 3 presents the performance comparison of MATCH applied to both post-sparsified and pre-sparsified LLMs on the NIAH benchmark under varying context lengths (8K–128K). The "Single" and "Multi" columns respectively report single-instance and multi-instance retrieval accuracies. Across all context settings, our methods consistently outperform the corresponding sparsified

Model	SWA ratio	LongBench Task Types					Average
		Single-Doc. QA (SQ)	Multi-Doc. QA (MQ)	Summ. (SM)	Few-shot Learning(FS)	Synthetic Tasks(ST)	
Post-sparsified LLM							
Qwen3	0.8	41.6	34.7	20.7	61.1	12.7	34.2
Qwen3 w/ RAG	0.8	42.0	38.0	21.1	59.7	21.0	36.4
Qwen3 w/ MATCH ($k = 4$)	0.8	**43.0**	35.9	19.9	**61.4**	24.0	36.8
Qwen3 w/ MATCH ($k = 8$)	0.8	42.1	36.0	19.9	61.2	**26.0**	**37.0**
Qwen3	0.5	43.1	36.5	21.3	61.7	36.7	39.9
Qwen3 w/ RAG	0.5	44.1	39.2	20.2	60.8	42.0	41.2
Qwen3 w/ MATCH ($k = 4$)	0.5	**44.8**	39.1	20.7	61.7	50.8	**43.4**
Qwen3 w/ MATCH ($k = 8$)	0.5	44.7	38.0	20.6	**62.0**	**51.8**	**43.4**
Pre-sparsified LLM							
Phi-3	1	24.4	21.9	20.8	47.0	4.0	23.6
Phi-3 w/ RAG	1	27.5	27.0	20.1	46.4	8.5	25.9
Phi-3 w/ MATCH ($k = 4$)	1	**29.6**	26.8	**20.8**	46.9	7.3	26.3
Phi-3 w/ MATCH ($k = 8$)	1	28.6	**27.1**	20.3	46.9	**9.7**	**26.5**

Table 2: Results on LongBench. MATCH is applied to post-sparsified and pre-sparsified LLMs. The sparsity ratio denotes the proportion of attention layers replaced with SWA. RAG and MATCH utilize the same retrieval content.

Model	8K		16K		32K		64K		128K		Avg.	Avg. ≤32K
	Single	Multi	Single	Multi	Single	Multi	Single	Multi	Single	Multi		
Qwen3	87.0	91.9	80.0	84.1	72.7	58.0	90.0	24.4	47.0	13.6	60.3	78.7
Qwen3 w/ MATCH ($k = 4$)	**100.0**	92.4	**99.7**	81.5	**93.0**	**59.7**	84.7	**29.0**	**80.7**	**21.0**	**69.8**	**85.2**
Qwen3 w/ MATCH ($k = 8$)	99.0	**93.0**	99.0	82.3	91.3	55.4	77.3	26.6	71.0	19.9	67.5	84.2
Phi-3	24.3	18.2	14.3	8.2	3.7	4.2	–	–	–	–	11.7	11.7
Phi-3 w/ MATCH ($k = 4$)	78.0	43.6	75.3	40.5	75.3	**36.9**	–	–	–	–	53.8	53.8
Phi-3 w/ MATCH ($k = 8$)	**80.0**	**47.5**	**78.7**	**45.0**	**77.0**	36.7	–	–	–	–	**56.4**	**56.4**

Table 3: Results on NIAH across context lengths. '–' denotes out-of-memory runs.

baselines (Qwen3 and Phi-3). The improvements are particularly pronounced in long-context scenarios (32K–128K), demonstrating that MATCH effectively mitigates degradation in retrieval performance as sequence length increases. When excluding the out-of-memory runs of the 64K and 128K setups, our approach yields higher average scores, confirming its robustness and generalization across different sparsity configurations.

6 Analysis and Discussion

6.1 Ablation Studies

Method	LongBench Task Types					
	SQ	MQ	SM	FS	ST	Avg.
Base	41.6	34.7	20.7	61.1	12.7	34.1
w/ Random	40.0	34.7	19.6	60.1	9.5	32.8
w/ MATCH	**42.1**	**36.0**	19.9	**61.2**	**26.0**	**37.0**

Table 4: Results of ablation of retrieval.

To verify that MATCH's gains are attributed to the introduction of the retrieved context rather than the mere increased exposure of attention, we replace the retriever with a random index generator while keeping all other hyperparameters identical. As shown in Table 4, using Qwen3 with 80% SWA sparsity and $k = 8$, the random-retrieval model consistently underperformed both full MATCH and even the pure sparse-attention baseline, indicating that naïvely adding extra KV pairs is ineffective and even harmful. This confirms the necessity of MATCH's carefully designed retrieval pipeline.

6.2 Comparison with KV-cache compression techniques

We further compare MATCH against two prominent sparse attention baselines StreamingLLM (Xiao et al., 2024c) and FlexPrefill (Lai et al., 2025) (for details, see Appendix B.4). We applied these two methods directly to the standard Qwen3 model, adhering to the hyperparameter settings recommended in their original papers to ensure optimal performance. Tables 5 and 6 present the results on LongBench and NIAH respectively. We also report the Time-to-First-Token (TTFT) latency and GPU

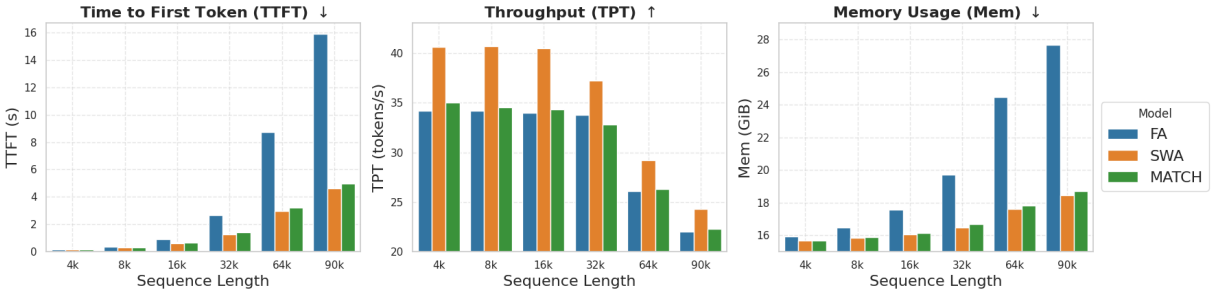

Figure 4: Performance comparison of attention mechanisms across sequence lengths: (a) Time to First Token (TTFT), (b) Throughput (Tokens Per Second), and (c) Memory consumption. Our method achieves competitive latency and throughput while maintaining significantly better memory efficiency than FA and comparable performance to SWA.

Method	LongBench Task Types						TTFT (s) ↓	Decoding Memory (GiB) ↓
	Single-Doc. QA ↑	Multi-Doc. QA ↑	Summ. ↑	Few-shot Learning ↑	Synthetic Tasks ↑	Avg. ↑		
StreamingLLM	42.5	34.2	24.7	59.3	33.7	38.9	1.8	19.8
FlexPrefill	43.3	37.9	24.8	58.5	26.5	38.2	1.6	19.7
MATCH	**44.7**	**38.0**	20.6	**62.0**	**51.8**	**43.4**	**1.4**	**16.7**

Table 5: Results on LongBench, time to first token (TTFT), and memory required for decoding by different attention sparsification methods.

Model	8K		16K		32K		64K		Average
	Single	Multi	Single	Multi	Single	Multi	Single	Multi	
StreamingLLM	70.7	65.4	50.0	38.3	49.7	22.0	60.7	14.8	43.6
FlexPrefill	93.7	76.9	96.0	74.4	98.3	66.8	84.7	39.7	75.2
MATCH	**100.0**	**92.4**	**99.7**	**81.5**	93.0	59.7	**84.7**	29.0	**76.4**

Table 6: Results on NIAH by different attention sparsification methods.

memory footprint during a typical LongBench sample inference for all three approaches.

We apply MATCH to the 50%-sparsified Qwen3 with $k = 8$, and it achieves the highest overall average score of 43.4 on LongBench, outperforming both StreamingLLM and FlexPrefill. It delivers clear gains in most of the task types except summarization. In addition, MATCH records the lowest latency and decoding memory footprint at 32K context length, highlighting its time and space efficiency advantage. We also observe that MATCH with $k = 4$ mostly outperforms the two other methods on NIAH.

Overall, the results indicate that MATCH enhances sparsified LLMs by restoring their representational and reasoning capabilities, leading to systematic performance gains without compromising sparsity efficiency. This highlights the potential of MATCH as a general plug-in framework for boosting the fidelity and reliability of sparse long-context models.

6.3 Efficiency Analysis

To better understand the advantages brought about by MATCH in handling long sequences, we conduct an additional experimental study to evaluate its computational efficiency relative to the backbone models it augments. In Fig. 4, we present ablations using Qwen3 with $k = 8$, comparing the base models with those enhanced by our method to quantify the incremental overhead introduced across three metrics—Time to First Token, Throughput, and Memory Consumption—over sequence lengths ranging from 4K to 90K tokens. While the MATCH module introduces a marginal increase in each computational factor, it remains lightweight, and the overall overhead compared to SWA is negligible, particularly for long sequences. Moreover, the augmented system achieves substantial efficiency gains compared to transformer models with full attention on our tasks.

Specifically, MATCH achieves a TTFT of 4.98s at 90K tokens, representing a 68% reduction from FA (15.88s) and remaining closely comparable to

SWA (4.62s). Furthermore, throughput remains stable across increasing sequence lengths, closely tracking FA's scaling behavior, while memory usage is reduced by 32% relative to FA (18.72 vs. 27.67 GiB). These results highlight that MATCH delivers near-SWA efficiency while preserving the expressive functionality of full attention. Overall, the method maintains a favorable balance among latency, throughput, and memory consumption, demonstrating its practicality for scalable long-context inference without compromising performance fidelity.

7 Conclusion

In this work, we propose MATCH, a lightweight retrieval-based knowledge integration mechanism that enhances the ability of sparsified LLMs to leverage example-specific context. MATCH introduces a novel approach that treats the input context as a dynamic datastore for retrieval, integrating the retrieved information with the input during both training and inference. This design enables models to utilize contextual information more effectively while overcoming the limitations imposed by local attention or data-independent sparse patterns. Across a variety of synthetic and real-world data sets, sparse-attention LLMs augmented with MATCH achieve substantial performance improvements over their base counterparts, highlighting its effectiveness as a general and broadly applicable framework.

8 Limitations

In this work, we focus on in-context retrieval tasks. We believe that a concentrated examination of these tasks allows us to delve deeper into the nuances and intricacies involved, thereby providing more insightful and meaningful findings. However, examining a more diverse range of tasks may help us to further and more broadly assess the effectiveness of our method. Furthermore, our method is designed to improve the context-copy ability of sparse attentions, and it operates without optimizing layer selection. Investing more time and resources into these aspects may further strengthen the method. We will leave these for future work.

References

Marah I Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat S. Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Caio César Teodoro Mendes, Weizhu Chen, Vishrav Chaudhary, Parul Chopra, and 68 others. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *CoRR*, abs/2404.14219.

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, and 1 others. 2025. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*.

Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Zou, Atri Rudra, and Christopher Ré. 2024. Zoology: Measuring and improving recall in efficient language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 3119–3137. Association for Computational Linguistics.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *CoRR*, abs/2004.05150.

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

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *Preprint*, arXiv:2402.03216.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *Preprint*, arXiv:1904.10509.

Tri Dao and Albert Gu. 2024. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

DeepSeek-AI. 2024. Deepseek llm: Scaling open-source language models with longtermism. *Preprint*, arXiv:2401.02954.

Jiayu Ding, Shuming Ma, Li Dong, Xingxing Zhang, Shaohan Huang, Wenhui Wang, Nanning Zheng, and Furu Wei. 2023. Longnet: Scaling transformers to 1, 000, 000, 000 tokens. *CoRR*, abs/2307.02486.

Xin Dong, Yonggan Fu, Shizhe Diao, Wonmin Byeon, ZIJIA CHEN, Ameya Sunil Mahabaleshwarkar, Shih-Yang Liu, Matthijs Van keirsbilck, Min-Hung Chen, Yoshi Suhara, Yingyan Celine Lin, Jan Kautz, and Pavlo Molchanov. 2025. Hymba: A hybrid-head architecture for small language models. In *The Thirteenth International Conference on Learning Representations*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and 82 others. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Albert Gu and Tri Dao. 2024. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling, COLM 2024, Philadelphia, PA, United States, October 7-9, 2024*. OpenReview.net.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938. PMLR.

Chi Han, Qifan Wang, Hao Peng, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. 2024. Lm-infinite: Zero-shot extreme length generalization for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 3991–4008. Association for Computational Linguistics.

Maor Ivgi, Uri Shaham, and Jonathan Berant. 2023. Efficient long-text understanding with short-text models. *Trans. Assoc. Comput. Linguistics*, 11:284–299.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot learning with retrieval augmented language models. *J. Mach. Learn. Res.*, 24:251:1–251:43.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *CoRR*, abs/2310.06825.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6769–6781. Association for Computational Linguistics.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023*, pages 611–626. ACM.

Xunhao Lai, Jianqiao Lu, Yao Luo, Yiyuan Ma, and Xun Zhou. 2025. Flexprefill: A context-aware sparse attention mechanism for efficient long-sequence inference. In *The Thirteenth International Conference on Learning Representations*.

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Chaofan Li, Zheng Liu, Shitao Xiao, and Yingxia Shao. 2023. Making large language models a better foundation for dense retrieval. *Preprint*, arXiv:2312.15503.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024. Snapkv: Llm knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 37:22947–22970.

Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Richard James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. RA-DIT: retrieval-augmented dual instruction tuning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Qianli Liu, Zicong Hong, Peng Li, Fahao Chen, and Song Guo. 2025. Mell: Memory-efficient large language model serving via multi-gpu KV cache management. In *IEEE INFOCOM 2025 - IEEE Conference on Computer Communications, London, United Kingdom, May 19-22, 2025*, pages 1–10. IEEE.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. 2023. Scissorhands: Exploiting the persistence of importance hypothesis for LLM KV cache compression at test time. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. 2024. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Shengjie Ma, Chengjin Xu, Xuhui Jiang, Muzhi Li, Huaren Qu, Cehao Yang, Jiaxin Mao, and Jian Guo. 2025. Think-on-graph 2.0: Deep and faithful large language model reasoning with knowledge-guided retrieval augmented generation. In *The Thirteenth International Conference on Learning Representations*.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Michael Poli, Armin W. Thomas, Eric Nguyen, Pragaash Ponnusamy, Björn Deiseroth, Kristian Kersting, Taiji Suzuki, Brian Hie, Stefano Ermon, Christopher Ré, Ce Zhang, and Stefano Massaroli. 2024. Mechanistic design and scaling of hybrid architectures. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Qwen. 2024. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 5418–5426. Association for Computational Linguistics.

Han Shi, Jiahui Gao, Xiaozhe Ren, Hang Xu, Xiaodan Liang, Zhenguo Li, and James Tin-Yau Kwok. 2021. Sparsebert: Rethinking the importance analysis in self-attention. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 9547–9557. PMLR.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, and 1 others. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems*, volume 33, pages 5776–5788. Curran Associates, Inc.

Guangxuan Xiao. 2025. Why stacking sliding windows can't see very far. `https://guangxuanx.com/blog/stacking-swa.html`.

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. 2024a. 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. 2024b. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024c. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2024. RECOMP: improving retrieval-augmented lms with context compression and selective augmentation. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *CoRR*, abs/2505.09388.

Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. 2024. Making retrieval-augmented language models robust to irrelevant context. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Re, Clark Barrett, Zhangyang Wang, and Beidi Chen. 2023a. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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. 2023b. H2O: heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

A Details of Retrieval

To reduce the latency of pre-filling on long prompts, we perform retrieval not at every token position but only for the last m token positions, and at regular intervals so that the same retrieved chunks are shared within each interval of U tokens. Concretely, x is chunked as:

$$(x_1, \ldots, x_N) \xrightarrow{\text{chunk}} (\boldsymbol{C}_1, \ldots, \boldsymbol{C}_T),$$
$$\text{where } \boldsymbol{C}_i = (x_{(i-1)U+1}, \ldots, x_{iU}),$$
$$T = \lfloor N/U \rfloor.$$

We then perform exact searches over the dense embeddings of chunks using a bi-encoder, which can be efficiently parallelized via matrices:

$$\mathbf{E}_i = \mathcal{E}(\boldsymbol{C}_i) \quad i \in [1, T], \tag{7}$$
$$\mathbf{S} = (\mathbf{E}_{\lfloor \frac{N-m}{U} \rfloor:T} \mathbf{E}^\top) + \mathbf{M} \tag{8}$$
$$\text{where } \mathbf{M}_{ij} = \begin{cases} 0, & i > j \\ -\infty & \text{otherwise} \end{cases}.$$

Finally, the top-k chunks per token position $I' = \{\boldsymbol{s}_1, ..., \boldsymbol{s}_N\}$ are obtained by

$$I'_i = \begin{cases} \arg\text{topk}_j(\mathbf{S}_{\tau j}) & \tau \in [N-m+1, N] \\ \varnothing & \text{otherwise} \end{cases} \tag{9}$$

where $x_i \in \boldsymbol{C}_\tau$ and $\mathcal{E}$ is the Sentence-BERT encoder.

The above is performed once per input sequence, incurring one-off computational costs of $O(TE)$ for equation 7 where E is the complexity of a forward pass of the Sentence-BERT model, and $O(T^2D)$ for equation 8 where D is the embedding dimension.

During decoding, we perform multi-query search, allowing retrieval at different granularities. We denote P as the lengths of the set of queries. For example, given $P = \{64, 128\}, K = 64, k = 8$, two queries are formed by taking the preceding 64 and 128 tokens respectively. Then, the bi-encoder would retrieve $\lfloor K/|P| \rfloor = 64/2 = 32$ chunks, and $\lfloor k/2 \rfloor = 8/2 = 4$ chunks will be selected for each query after reranking. We further discuss the design choices of reranking in Appendix C.1. As in pre-filling, retrieval is also performed at intervals to minimize latency.

B Experimental Details

B.1 Adapting LLMs via Continual Training

Weights of LLMs pre-trained with full attention or sliding window attention (Jiang et al., 2023; Dubey et al., 2024; Qwen, 2024; Abdin et al., 2024; Team et al., 2025; Yang et al., 2025) are conditioned to operate under the respective attention setups. A direct replacement of original attention layers by MATCH's ones could introduce significant discrepancies between training and inference. Consequently, we propose adaptations via continual pretraining.

For LLMs with both post-sparsified and pre-sparsified SWA, we conduct a two-stage continual pre-training. For the first stage, we sampled 25B data from Cosmopedia and 25B from Fineweb-edu. We also collect data from several curated data sets, including 14B data extracted from the ProLong dataset, a long-QA retrieval dataset containing contexts, questions, and answer pairs. Long paragraphs of gathered results with corresponding answers were generated by an OpenAI chat agent. The second stage utilizes 1.5B Prolong short-mix data, 1.5B NIAH-like synthetic data, and 1B Narriative QA dataset.

B.2 Tasks of LongBench

The tasks of LongBench (Bai et al., 2024) we evaluated on are listed in Table 7.

B.3 Details of Experiment on Synthetic Tasks

B.3.1 MQAR Setup

In our MQAR experiment setting, all models were implemented as 2-layer sequence mixers, trained on 100K samples, and evaluated on a hold out test set of 3K samples, following standard MQAR experiment settings. Sequence lengths ranged from 64 to 512, containing 4 to 64 recall pairs, respectively. For each experiment, we performed a sweep over four learning rates from 10^{-4} to 10^{-2} and reported the best performance. All Sparse Attention models shares a window size of 32. We averaged the performance across three experimental runs with different random seeds.

Dataset Name	Context Type	Avg. Length (Tokens)	Evaluation Metrics
2WikiMQA	Multi-document	2.5K	F1
DuReader	Single-document	1.6K	Rouge-L
Gov_Report	Single-document	9.4K	Rouge-L
HotpotQA	Multi-document	10.8K	F1
LSHT	Single-document	13.5K	Accuracy
Multi-News	Multi-document	11.8K	Rouge-L
MultiFieldQA-en	Multi-document	4.2K	F1
MultiFieldQA-zh	Multi-document	4.3K	F1
Musique	Multi-document	11.8K	F1
NarrativeQA	Single-document	20.6K	F1
Passage Count	Single-document	9.1K	Accuracy
Passage Retrieval-en	Multi-document	10.8K	Accuracy
Passage Retrieval-zh	Multi-document	10.8K	Accuracy
Qasper	Single-document	4.9K	F1
QMSum	Multi-document	9.8K	Rouge-L
Samsum	Dialogue	0.6K	Rouge-L
TREC	Single-document	5.0K	Accuracy
TriviaQA	Multi-document	7.8K	F1
VCSum	Dialogue	10.3K	Rouge-L

Table 7: Tasks from LongBench on which we evaluate.

B.3.2 MAD Setup

For each task in MAD, we report the accuracy on a held-out test set, where a prediction is considered correct only if the entire output sequence is exactly matched. All models are trained from scratch on the target task using a 4-layer architecture with a vocabulary size of 16 for ICR and FuzzyICR and 32 for NoisyICR, respectively. Each model has a hidden size of 128, and for models augmented with MATCH, we tried two variants of context chunk size of 2 and 4.

B.4 Configurations of Sparse Attention Baselines

We follow the recommended configurations in the original papers of StreamingLLM and FlexPrefill. For StreamingLLM, we set `global_window` to be 1024 and `local_window` to be 2048. For FlexPrefill, we set `block_size` to be 128, γ to be 0.9, τ to be 0.1, and `min_budget` to be 512.

B.5 Comparing with Retrieval-Augmented Generation

We have tried different ways of performing RAG. This includes appending the retrieved chunks to the end of the input sequence and replacing parts of the texts in the sliding window with the chunks. We find that appending the chunks to the end yields substantially worse performance. We report the best-scoring RAG results on LongBench.

C Additional Results

In the following subsections, we present more experimental results evaluated on LongBench for understanding the effects of different components in MATCH. For brevity, we denote single-doc. QA, multi-doc. QA, summarization, few-shot learning, and synthetic tasks as SQ, MQ, SM, FS, and ST respectively in the column headers of the tables below.

C.1 Effect of Reranking

k	Rerank	LongBench Task Types					
		SQ	MQ	SM	FS	ST	Avg.
8	Yes	**42.1**	**36.0**	**19.9**	61.2	**26.0**	**37.0**
8	No	41.1	34.8	19.5	**61.4**	21.7	35.7
4	Yes	**43.0**	**35.9**	**20.0**	**61.4**	**24.0**	**36.9**
4	No	40.6	35.3	19.9	60.5	22.7	35.8

Table 8: Results of MATCH with and without the reranking step in the retriever.

Table 8 shows the results of applying MATCH to the Qwen3 model with and without reranking during decoding. It is seen that the models with reranking almost always outperform those that do not. Nevertheless, as the reranker adopts a cross-encoder architecture, it is costly in terms of memory and time to perform pairwise scoring during pre-filling when all query chunks in the input sequence have to undergo retrieval. For instance, if we set the hyperparameters to $P \in 64, 128$, $U = K = 128$, and $m = 3000$ (refer to Ap-

(U, P, k, m)	LongBench Task Types					
	SQ	MQ	SM	FS	ST	Avg.
(128, {64, 128}, 8, 1000)	42.1	36.0	19.9	61.2	26.0	37.0
(128, {64, 128}, 8, −4096)	41.7	35.8	19.9	60.9	27.3	37.1
(128, {128}, 4, 1000)	42.3	35.3	19.9	61.0	22.7	36.2
(128, {32, 64, 96, 128}, 8, 1000)	42.4	36.9	19.6	60.2	24.5	36.7
(64, {64}, 8, 1)	42.0	35.3	19.4	61.4	22.2	36.1
(64, {64}, 8, 1000)	41.8	35.6	19.3	61.4	23.7	36.4

Table 9: Results of MATCH with different configurations of hyperparameters

pendix A for details), it would take up to 4 seconds for pre-filling. While it is possible to also add reranking during pre-filling, we wish to make our framework practical across hyperparameters settings. Therefore, we choose to rely solely on the efficient matrix-based bi-encoder (equations 7, 8, and 9) during pre-filling for applicability of different hyperparameters and simplicity.

C.2 Sensitivity to Hyperparameters

Table 9 presents the results of MATCH under different hyperparameter configurations. Several observations can be drawn. First, performance remains relatively stable across configurations, indicating that MATCH is not overly sensitive to hyperparameter choices. Second, all configurations yield substantial improvements over the baseline (see Table 2).

From our experiments, we find that increasing m (from 0) generally yields better performances but only very marginally when m is large enough. In Table 9, $m = -4096$ means retrieval is performed from 4096th position onward, which generally covers more tokens than when $m = 1000$ for LongBench. From this empirical insight, while there may be opportunities to optimize over m, we do not expect significant benefits by making it dynamic. We set m to be 1000 for simplicity in this work. We also find that a larger chunk size (128 over 64) is also often better. Moreover, retrieval using multiple queries indeed can help increase the granularity and consequently boost performances, but it can bring adverse effects when too many queries are used.

C.3 Detailed MQAR Results

The detailed results for plotting Fig. 3 are shown in Table 10.

Method	Model Dimension (d)			
	64	128	256	512
Sequence Length = 64				
Attention	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
SWA	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	0.79 (0.36)
StreamingLLM-1	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	0.87 (0.11)
StreamingLLM-4	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	0.85 (0.26)
StreamingLLM-16	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
Strided - 64	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	0.69 (0.31)
Strided - 32	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	0.79 (0.36)
Strided - 16	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
Random - 1	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
Random - 2	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
Random - 4	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
MATCH	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
Sequence Length = 128				
Attention	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
SWA	0.81 (0.00)	0.81 (0.00)	0.81 (0.00)	0.75 (0.10)
StreamingLLM-1	0.81 (0.00)	0.81 (0.00)	0.81 (0.00)	0.73 (0.06)
StreamingLLM-4	0.86 (0.01)	0.86 (0.01)	0.86 (0.00)	0.82 (0.03)
StreamingLLM-16	0.69 (0.32)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
Strided - 64	0.85 (0.01)	0.85 (0.01)	0.83 (0.00)	0.65 (0.00)
Strided - 32	0.86 (0.01)	0.86 (0.01)	0.85 (0.00)	0.77 (0.09)
Strided - 16	0.88 (0.01)	0.89 (0.01)	0.86 (0.00)	0.75 (0.06)
Random - 1	0.84 (0.00)	0.84 (0.01)	0.83 (0.00)	0.74 (0.12)
Random - 2	0.85 (0.01)	0.85 (0.01)	0.84 (0.00)	0.84 (0.00)
Random - 4	0.88 (0.01)	0.89 (0.00)	0.86 (0.02)	0.87 (0.01)
MATCH	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
Sequence Length = 256				
Attention	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
SWA	0.60 (0.00)	0.60 (0.00)	0.60 (0.00)	0.47 (0.12)
StreamingLLM-1	0.60 (0.00)	0.60 (0.00)	0.60 (0.01)	0.44 (0.05)
StreamingLLM-4	0.64 (0.00)	0.64 (0.00)	0.58 (0.11)	0.64 (0.00)
StreamingLLM-16	0.70 (0.00)	0.80 (0.00)	0.80 (0.00)	0.80 (0.00)
Strided - 64	0.64 (0.01)	0.65 (0.00)	0.64 (0.02)	0.55 (0.14)
Strided - 32	0.65 (0.00)	0.66 (0.00)	0.47 (0.33)	0.65 (0.01)
Strided - 16	0.69 (0.01)	0.71 (0.01)	0.50 (0.36)	0.72 (0.02)
Random - 1	0.63 (0.00)	0.64 (0.01)	0.63 (0.00)	0.61 (0.03)
Random - 2	0.65 (0.00)	0.64 (0.01)	0.64 (0.00)	0.63 (0.00)
Random - 4	0.67 (0.01)	0.68 (0.00)	0.67 (0.00)	0.66 (0.02)
MATCH	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
Sequence Length = 512				
Attention	0.67 (0.58)	1.00 (0.00)	1.00 (0.00)	0.96 (0.06)
SWA	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
StreamingLLM-1	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
StreamingLLM-4	0.03 (0.00)	0.04 (0.03)	0.08 (0.00)	0.05 (0.04)
StreamingLLM-16	0.13 (0.00)	0.17 (0.00)	0.15 (0.04)	0.15 (0.04)
Strided - 64	0.00 (0.00)	0.00 (0.00)	0.02 (0.00)	0.02 (0.00)
Strided - 32	0.00 (0.00)	0.00 (0.01)	0.02 (0.00)	0.02 (0.00)
Strided - 16	0.00 (0.00)	0.01 (0.02)	0.02 (0.00)	0.12 (0.09)
Random - 1	0.00 (0.00)	0.01 (0.00)	0.02 (0.00)	0.02 (0.01)
Random - 2	0.00 (0.00)	0.02 (0.00)	0.02 (0.00)	0.02 (0.01)
Random - 4	0.00 (0.00)	0.01 (0.00)	0.02 (0.00)	0.02 (0.00)
MATCH	0.97 (0.06)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)

Table 10: Detailed MQAR performance comparison (accuracy) across different sequence lengths and model dimensions. The scores are the average of three runs with different seeds. The numbers in brackets are the standard deviations.