CompressKV: Semantic-Retrieval-Guided KV-Cache Compression for Resource-Efficient Long-Context LLM Inference

Xiaolin Lin[1,*], Jingcun Wang[1], Olga Kondrateva[1], Yiyu Shi[2], Bing Li[3] and Grace Li Zhang[1]

[1]Technical University of Darmstadt, Darmstadt, Germany
[2]University of Notre Dame, Notre Dame, IN, USA
[3]Technical University of Ilmenau, Ilmenau, Germany

Abstract

Long-context large language model (LLM) inference is increasingly constrained by the memory footprint and decoding cost of key-value (KV) caches, limiting sustainable deployment on resource-constrained hardware. Existing KV cache eviction methods typically apply heuristic token scoring over all heads in GQA-based LLMs. These methods ignore the different functionalities of attention heads, leading to the eviction of critical tokens and thus degrading the performance of LLMs. To address this issue, we propose CompressKV, a resource-efficient KV-cache compression framework for GQA-based LLMs. Instead of aggregating attention scores from all heads, CompressKV identifies Semantic Retrieval Heads (SRHs) that capture both the initial and final tokens of a prompt and semantically important mid-context evidence, and uses them to select tokens whose KV pairs should be retained. Furthermore, CompressKV allocates cache budgets across layers according to offline estimates of layer-wise eviction error. Experiments on LongBench and Needle-in-a-Haystack show that CompressKV consistently outperforms existing KV-cache eviction methods across memory budgets. Notably, it preserves over 97% of full-cache performance using only 3% of the KV cache on LongBench question-answering tasks and achieves 90% accuracy with just 0.7% KV storage on Needle-in-a-Haystack. These results demonstrate an improved resource–performance trade-off for long-context LLM inference. Our code is publicly available at: https://github.com/TUDa-HWAI/CompressKV

Keywords

Large Language Models, Long-Context Inference, KV-Cache Compression, Resource-Efficient AI, Efficient Inference

1. Introduction

Recent advances in large language models (LLMs) [1, 2, 3, 4, 5] have boosted their long-context processing capabilities. However, with the increasing length of texts, the resulting key-value (KV) cache size grows linearly. The large KV cache leads to slow inference due to the attention calculation across past KV cache. In addition, the large KV cache requires substantial memory storage, which creates a major bottleneck in the deployment of long-context LLMs. Therefore, effective compression of KV cache is essential for optimizing the computational efficiency and model scalability.

State-of-the-art KV cache compression focuses on quantization, low-rank approximation, and KV cache eviction [6, 7, 8, 9, 10, 11, 12, 13]. Among them, KV-cache eviction—discarding KV pairs for unimportant tokens while retaining the rest—has attracted increasing attention.

Several criteria have been proposed to identify tokens for KV-cache eviction. For example, StreamingLLM [9] retains the first and last tokens and neglects potentially important tokens in the middle of the prompt. SnapKV [10] clusters recent attention scores within an observation window at the end of the prompt, either per head or per head group, to identify and retain the important tokens receiving the highest attention values. CAKE [13] extends SnapKV's method by adding the attention

SuRE'26: Workshop on Sustainability and Resource-Efficiency of Artificial Intelligence, August 17, 2026, Bremen, Germany
*Corresponding author.
xiaolin.lin@tu-darmstadt.de (X. Lin); jingcun.wang@tu-darmstadt.de (J. Wang); olga.kondrateva@tu-darmstadt.de (O. Kondrateva); yshi4@nd.edu (Y. Shi); bing.li@tu-ilmenau.de (B. Li); grace.zhang@tu-darmstadt.de (G. L. Zhang)

variance in an observation window to the eviction score, enabling it to capture tokens whose importance fluctuates over time.

While the above criteria work well in many KV-cache eviction scenarios, they overlook head heterogeneity: all heads are weighted equally, and eviction decisions are made from aggregated attention scores (typically the sum across heads within a group). In fact, attention heads exhibit different functionalities. For example, in Grouped Query Attention (GQA)-based LLMs [14], some attention heads, called Streaming Heads, exclusively focus on the beginning and the end of a prompt [9, 15]. When the attention heads within a GQA group are dominated by Streaming Heads, those heads have the largest influence on KV cache eviction, resulting in only the initial and last tokens' KV pairs being retained. As a result, crucial mid-context tokens may be evicted, degrading LLM performance.

Besides eliminating KV pairs for those unimportant tokens, state-of-the-art research also allocates specified memory budgets to layers. For example, [9, 10] allocates each layer to a fixed number of KV pairs without considering layer difference. [12, 11, 13] allocates KV cache budget across layers based on attention distributions or layer-wise statistics such as attention entropy or variance, which often require additional online computation cost. Moreover, attention distributions can vary significantly across models, which limits the generalization ability and effectiveness of these allocation strategies. Orthogonally, HeadKV [16] and AdaKV [17] extend to head-level budget allocation.

In this paper, we observe that certain attention heads are capable of retrieving important tokens within the text and attending to their surrounding semantic context. We refer to these heads as Semantic Retrieval Heads. Motivated by this observation, we identify such Semantic Retrieval Heads in each layer and use them to determine the crucial tokens and share a unified set of crucial token indices across all heads within that layer. This approach can substantially address the dominance of Streaming Heads in KV cache evictions, so that it can enhance the performance of GQA-based models. Furthermore, we analyze the cache eviction error of each layer individually and introduce a layer-adaptive KV cache allocation strategy. Our contributions are as follows:

(1) We introduce a Semantic-Retrieval–driven mechanism to address streaming-head dominance in GQA, preventing important tokens from being evicted. The identified Semantic Retrieval Heads then guide token importance and KV-cache eviction. Our experimental results demonstrate Semantic Retrieval Heads know what tokens are unimportant before generation.

(2) We estimate each layer's compression impact by computing the Frobenius norm of the difference between its attention-block outputs with the compressed cache and those with the full cache, during the decoding stage. Cache budgets are then proportionally assigned across layers, prioritizing layers with higher errors. Importantly, this analysis is performed offline and does not introduce any additional overhead during online inference.

(3) CompressKV is validated on multiple LLMs using LongBench and Needle-in-a-Haystack (NIAH). On LongBench, CompressKV maintains over 99% of full-cache performance with only 19% of KV budget and retains 97% of question-answering accuracy using just 3% of the cache. On Needle-in-a-Haystack retrieval benchmark, it achieves 90% of the baseline accuracy with only 0.7% of KV storage.

2. Background and Related Work

2.1. KV-Cache Compression for Budgeted Long-Context Inference

To alleviate the burden of KV cache storage, various KV cache compression methods, e.g., quantization [6], low-rank approximations [7], and KV cache eviction strategy have been proposed. In particular, KV cache eviction reduces cache size by removing KV cache pairs of unimportant tokens without retraining. There are different eviction strategies. For example, StreamingLLM [9] focuses solely on retaining the first and last tokens, which only addresses the Streaming Head scenario and neglects potentially important tokens in the middle of the sequence. To overcome this limitation, more advanced methods have been proposed [18, 19, 10, 20, 21]. A representative example is SnapKV [10], which clusters recent attention scores, either per head or per head group to identify important token and retain the KV cache pairs of such tokens. Besides, recent approaches, including PyramidKV [11], D2O [22],

Figure 1: Motivation. (a) The attention score distribution of a streaming head (SH). (b) The attention score distribution of a retrieval head (RH). (c) Streaming attention heads in a GQA group dominate the token eviction, indicating only the initial and final tokens are retained. The critical tokens are evicted.

and CAKE [13], dynamically allocate cache budgets based on attention statistics or modeled attention dynamics of all the layers in an LLM. Beyond layer-level allocation, HeadKV [16] and AdaKV [17] further enhance cache budget with head-level budget allocation. Their selection strategies for important tokens are an extended version of SnapKV's eviction strategy.

Despite their effectiveness, existing eviction pipelines have two limitations that are especially relevant to GQA-based LLMs. First, many prior KV cache eviction pipelines compute token importance via head-agnostic pooling (e.g., across heads within each GQA group) when selecting tokens for eviction, effectively treating all attention heads equally and ignoring their functional heterogeneity; Recent work [23, 24, 25, 26, 27, 28, 29, 30, 16] has shown that different attention heads have distinct roles. For example, some attention heads, called Streaming Heads in the state-of-the-art research, always focus on the beginning and the end of a prompt. For example, in Figure 1 (a), head 0 is such a Streaming Head since the attention scores of the initial token and the last tokens are larger than the remaining tokens. On the contrary, some attention heads, called Retrieval heads in [27], exhibit copy-and-paste behaviors for long-context scenarios. For example, in Figure 1 (b), head 1 is such a retrieval head since the attention scores of the correct answer "sandwich" are larger. HeadKV [16] further scores heads using retrieval and reasoning signals. In GQA-based LLMs, Streaming Heads tend to have larger effect than the other heads for KV cache eviction, which indicates only KV cache pairs corresponding to initial and last tokens are retained. This leads to the eviction of crucial tokens in the middle of a prompt and thus degrades the performance of LLMs. Figure 1 (c) illustrates such an example, where Streaming Heads including head0 and head1 dominate token eviction for KV cache compression.

Second, existing layer-adaptive allocation methods [12, 11, 13] often rely on attention distributions or layer-wise statistics such as entropy and variance. These signals can introduce additional online computation and may vary across models, making the resulting allocation less robust under fixed resource budgets. In contrast, CompressKV identifies Semantic Retrieval Heads offline and uses them to guide retrieval-aware token selection, while assigning layer-wise budgets according to offline eviction-error estimates. This design improves the accuracy–memory trade-off without requiring online layer-importance estimation or budget search during generation.

3. CompressKV

CompressKV includes three key components: (1) Identification of the attention heads that are capable of retrieving important tokens within the text and attending to their surrounding semantic context. (2) Important token selection driven by such identified heads. (3) Error-aware layer-adaptive cache allocation. In the following subsections, we will first explain our observations and insights into identification of attention heads with specified functionalities. Afterwards, we will take advantage of such heads to select tokens for KV cache eviction. Furthermore, different cache budgets will be allocated to different layers.

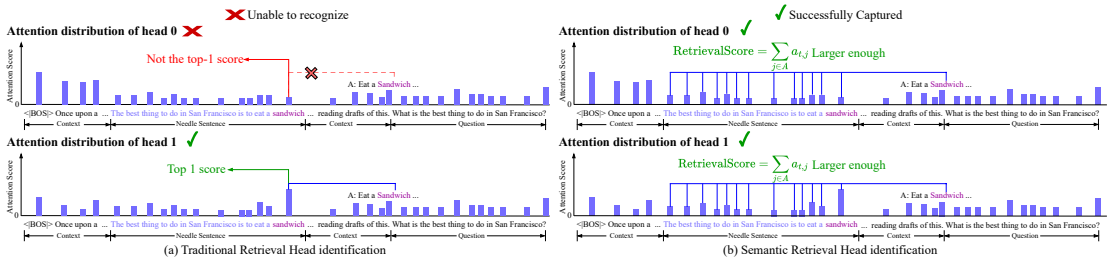

Figure 2: Illustration of Semantic Retrieval Head identification versus traditional Retrieval Head selection. Semantic Retrieval Heads capture attention over the entire answer span, addressing the limitations of traditional methods that rely solely on copy-and-paste behavior.

3.1. Observations and Insights

To prevent Streaming Attention Heads from dominating KV-cache eviction as illustrated in Figure 1 (c), we use Retrieval Heads—rather than all attention heads—to identify important tokens for KV-cache eviction. Importantly, most prior methods do not leverage retrieval heads for token-level eviction decisions; for example, HeadKV mainly uses retrieval-head signals for head-level KV budget allocation instead of selecting tokens to keep or evict. To this end, we first review how prior work identifies Retrieval Heads.

Previous work identifies Retrieval Heads using a strict top-1 rule, indicating that those attention heads, the highest attention score of which aligns exactly with the correct token answer during generation, are labeled as Retrieval Heads [27]. This identification technique emphasizes copy-and-paste behavior. [30] extends copy-and-paste identification by classifying both echo heads (copy-and-paste to the identical prior token) and induction heads (an extension that attends to the immediately preceding token) as Retrieval Heads. HeadKV [16] relaxes the strict top-1 criterion to a top-N hit: at each decoding step, a head is credited if the ground-truth answer token ranks within its top-k attention weights.

Although HeadKV are more relaxed than strict top-1, this criteria still remains peak-driven, privileging sharp attentions on the answer token. In long contexts where attention is sparse and skewed towards boundary tokens—top-1 rules yield low hit rates and can under-credit attention heads whose attention covers the answer span and its semantic neighborhood without placing a single sharp peak on the exact answer token. In HeadKV, if parts of the answer span do not appear within the top-k ranked positions, heads allocating substantial attention to these tokens may not be credited. For instance, in Figure 2 (a), head 0 fails to receive credit because the relevant tokens fall outside the top-k range despite providing coverage around the correct answer. Moreover, because the top-k threshold in HeadKV is tied to the answer length, when answers are short, e.g., only one or two tokens, this method returns back to the original strict top-1 regime.

To address this limitation, we introduce Semantic Retrieval Heads (SRH), a span-aggregation standard that credits attention heads for both copy-and-paste behaviours and deeper semantic dependencies. We then use such heads to identify important tokens for KV cache eviction, thereby preventing crucial mid-prompt evidence from being suppressed by streaming heads. For a visual comparison between Semantic Retrieval Heads and traditional Retrieval Heads, please refer to Section 4.7.

3.2. Semantic Retrieval Head Identification Standards

Instead of requiring exact top-k hits in the traditional Retrieval Head identification, we use a calibration dataset (following [27]; provided in our codebase) to evaluate each head h by aggregating its attention mass over the entire answer span whenever the model generates a correct answer token. Formally, we define the SRH score $S_{\mathrm{SRH}}(h)$ as

$$S_{\mathrm{SRH}}(h) = \sum_{t=1}^{N} \mathbf{1}_{\{y_t \in \mathcal{A}\}} \sum_{j \in \mathcal{A}} a_{t,j}^{(h)}. \tag{1}$$

where y_t is the generated token at step t, $\mathcal{A}$ is the answer span, and $a_{t,j}^h$ is head h's attention weight on the j-th token of $\mathcal{A}$. The higher the score of a head is, the more capable of capturing semantic information this head is.

Figure 2 (b) illustrates the concept of this new identification standard. By summing over the entire span, we can capture attention heads that contribute semantically relevant context even when they never achieve top-1 attention on a single token. Aggregation over multiple tokens enables the method to recognize heads that attend to semantic cues—such as "eat" or "a thing" around "sandwich"—rather than only pure copy-and-paste patterns. For example, head 0 in Figure 2 is considered as Semantic Retrieval Head in our new standard although it is not considered as Retrieval Head in the traditional identification methods. For a visual comparison between Semantic Retrieval Heads and traditional Retrieval Heads, please refer to Section 4.7.

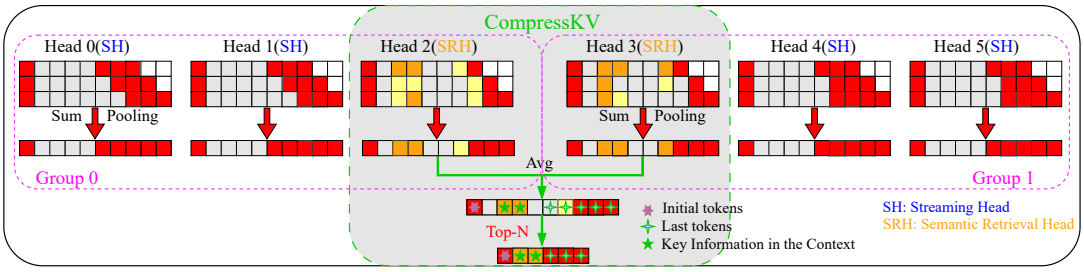

Figure 3: Illustration of the token selection driven by Semantic Retrieval Heads.

3.3. Token Selection Driven by Semantic Retrieval Heads

In GQA-based LLMs, for each layer, we will select the top-k Semantic Retrieval Heads with high scores defined with equation 1 as the criterion for selecting important tokens for KV cache eviction. All the attention heads within this layer share a common set of selected token indices determined by these top Semantic Retrieval Heads. This concept is illustrated in Figure 3, where a layer has two groups. In this example, Head 2 and Head 3 are top 2 Semantic Retrieval Heads. Following SnapKV, the attention score matrices of such heads are compressed by summing over the observation window and pooling across the token dimension. Afterwards, such compressed vectors are averaged. The tokens with the top N highest attention scores will be selected and their corresponding KV cache pairs will be retained. The KV cache pairs for the remaining tokens will be evicted to compress KV cache.

3.4. Error-Aware Layer-Adaptive Cache Allocation

To maximize memory efficiency under strict budget constraints, we propose an error-aware and layer-adaptive cache allocation strategy. Instead of relying on attention statistics as in the previous methods, this approach quantifies the compression error caused by KV cache compression, using full-cache outputs as the reference. We specifically focus on the extreme compression setting, where only a small fraction of tokens are retained in each layer's KV cache. For each layer l and decoding step t, let $\mathbf{O}_{\text{full},t}^l$ and $\mathbf{O}_{\text{comp},t}^l$ denote the attention outputs using the full and compressed KV caches, respectively:

$$\mathbf{O}_{\text{full},t}^l = \mathbf{W}_O^l \,\text{Attn}\Big(\mathbf{Q}_t^l,\ \mathbf{K}_{\text{full}}^l,\ \mathbf{V}_{\text{full}}^l\Big)\,, \tag{2}$$

$$\mathbf{O}_{\text{comp},t}^l = \mathbf{W}_O^l \,\text{Attn}\Big(\mathbf{Q}_t^l,\ \mathbf{K}_{\text{comp}}^l,\ \mathbf{V}_{\text{comp}}^l\Big)\,. \tag{3}$$

where $\mathbf{W}_O^{(l)}$ is the output projection matrix of layer l, $\mathbf{Q}_t^l$ is the query, $\mathbf{K}^l$ is the key, and $\mathbf{V}^l$ is the value representation at layer l. To evaluate the error incurred by compressing KV cache per layer, the error score for layer l is computed and normalized as:

$$e^{(l)} = \sum_{t=1}^{T} \frac{\|\Delta \mathbf{O}_t^l\|_F}{\|\mathbf{O}_{\text{full},t}^l\|_F + \epsilon}, \quad \tilde{e}^{(l)} = \frac{e^{(l)}}{\sum_k e^{(k)}} \tag{4}$$

where T is the total number of decoding steps,$\|\cdot\|_F$ denotes the Frobenius norm and ϵ is a small positive constant (e.g., 10^{-6}) to prevent division by zero.

Given the normalized per-layer error scores $\tilde{e}$ and total cache budget B_{total}, we first assign a minimum allocation m and a maximum allocation M to each layer to avoid a layer either has no memory budget or a large memory budget. The remaining budget is distributed in proportion to the error scores.

4. Experiments

Baselines and Models. We compare CompressKV with six KV-cache eviction baselines: StreamingLLM [9], SnapKV [10], PyramidKV [11], CAKE [13], HeadKV [16], and AdaKV [17]. All methods are evaluated with greedy decoding on Llama-3.1-8B-Instruct [3], Mistral-7B-Instruct-v0.3 [31], Qwen2.5-14B-Instruct, and Qwen2.5-32B-Instruct [4]. We further examine the orthogonality of CompressKV in Section 4.6, where we integrate it with head-level allocation, prefilling acceleration, and KV-cache quantization.

Evaluating Tasks. To evaluate CompressKV's performance under different memory budgets, we adopt two comprehensive benchmarks and one masking-based ablation analysis: (1) LongBench [32], which contains 16 long-context subtasks across single-document QA, multi-document QA, summarization, few-shot learning, synthetic tasks, and code completion. (2) Needle-in-a-Haystack(NIAH) [33], which measures the retrieval of a target answer hidden in extended text; and (3) an ablation of retrieval head types, following [27], where we selectively disable SRH and TRH to quantify their contributions. We also compare CompressKV with TRH vs. SRH under equal per-layer KV budgets, e.g., 256 tokens and report results separately.

Implementation Details. We evaluate all methods under average per-layer KV-cache budgets, denoted as $B_{\text{per-layer}}$, ranging from 128 to 2048 tokens. Given a total KV-cache budget B_{total} over L transformer layers, $B_{\text{per-layer}} = B_{\text{total}}/L$ denotes the average budget assigned to each layer. StreamingLLM and SnapKV use uniform layer-wise budgets, whereas PyramidKV, CAKE, and CompressKV redistribute budgets across layers under the same total memory constraint. HeadKV and AdaKV are applied at the GQA-group granularity to respect grouped-query attention. For fairness, all methods evict tokens only during prefilling and use the same local attention setting as SnapKV [10]: $\texttt{window\_size} = 8$ and $\texttt{kernel\_size} = 5$.

For CompressKV, we select the top four SRHs per layer, identified offline once per model using the calibration data from [27]. Layer-adaptive allocation is also computed offline by measuring normalized Frobenius-norm reconstruction errors between compressed-cache and full-cache attention-block outputs under minimal-size KV compression on LongBench. We constrain each layer's budget to $[m, M]$, with $m = 32$ and $M = 3 \times B_{\text{per-layer}}$, and allocate the remaining KV pairs proportionally to the normalized errors. During inference, the precomputed SRH sets and layer-wise budgets are fixed and reused for all samples, so CompressKV does not require online layer-importance estimation or additional dynamic profiling during generation.

4.1. Evaluation on LongBench Benchmark

Table 1 reports average LongBench scores under two representative KV-cache budgets: 256 tokens for tight memory settings and 1024 tokens for moderate compression. CompressKV consistently achieves the highest average performance across model families and scales, ranging from Llama-3.1-8B-Instruct(Llama-3.1-8B) and Mistral-7B-Instruct-v0.3(Mistral-7B) to larger Qwen2.5-14B-Instruct(Qwen2.5-14B) and Qwen2.5-32B-Instruct(Qwen2.5-32B) models. The gains are most pronounced under the tighter 256-token budget, showing that CompressKV is especially effective when KV-cache memory is severely constrained. These results also indicate that the benefit of CompressKV generalizes from 7B/8B-scale models to larger 14B/32B models. As illustrated in Figure 4, we further

Table 1
Average LongBench scores under fixed KV-cache budgets. FullKV is the uncompressed reference and does not depend on the budget. The Budget row indicates the average retained KV tokens per layer for compressed methods.

Method	Small-scale LLMs				Large-scale LLMs			
	Llama-3.1-8B		Mistral-7B		Qwen2.5-14B		Qwen2.5-32B	
	256	1024	256	1024	256	1024	256	1024
FullKV	49.08		47.82		49.80		48.57	
Budget	256	1024	256	1024	256	1024	256	1024
StreamingLLM	33.92	36.95	31.22	34.73	25.86	29.84	25.28	29.62
SnapKV	45.21	47.82	43.76	46.48	43.77	48.18	43.36	47.34
PyramidKV	44.36	47.65	43.06	45.96	42.71	47.70	42.11	46.98
CAKE	46.30	47.97	44.73	46.66	44.70	48.52	44.49	47.51
HeadKV	44.11	47.05	44.10	46.41	44.21	48.42	44.02	47.50
AdaKV	44.45	47.94	43.75	46.38	43.68	48.19	43.30	47.33
CompressKV	**46.71**	**48.24**	**45.43**	**46.96**	**45.37**	**48.69**	**44.73**	**47.78**

benchmark CompressKV on LongBench across KV-cache sizes from 128 to 2048 using Llama-3.1-8B-Instruct and Mistral-7B-Instruct-v0.3. CompressKV consistently outperforms all baselines across the full budget range, with the largest margins at small cache sizes. These results show that SRH-guided token selection and layer-adaptive budget allocation improve the memory–performance trade-off of long-context LLM inference, especially under strict memory constraints.

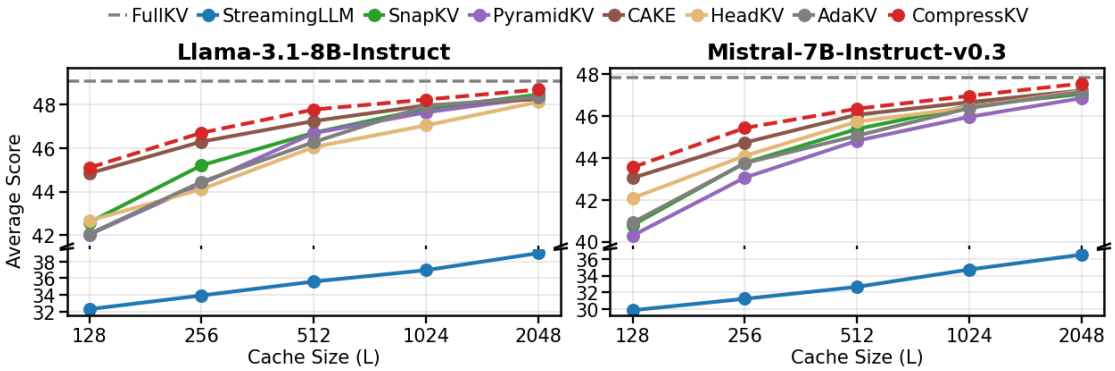

Figure 4: Average performance on 16 LongBench datasets under varying KV-cache budgets, compared with baseline methods.

4.2. Evaluation on Needle In A Haystack

Figure 5 presents average Needle-in-a-Haystack performance across KV budgets for Llama-3.1-8B-Instruct (8K–128K context) and Mistral-7B-Instruct-v0.3 (2K–32K)—showing CompressKV consistently surpasses competing methods at every budget. On Mistral-7B-Instruct-v0.3, CompressKV, HeadKV, and CAKE achieve near lossless compression with as few as 256 KV budget, highlighting their robustness. On Llama-3.1-8B-Instruct, AdaKV and HeadKV also underperform at low budgets, while CompressKV achieves nearly lossless performance at a 2048 KV budget (5% of the full cache) and still retains 90% of the original performance with only 256 KV budget (0.7% capacity). Together with the LongBench evaluation, these results show that CompressKV preserves general LLM performance across diverse long-context tasks while delivering efficient KV-cache compression.

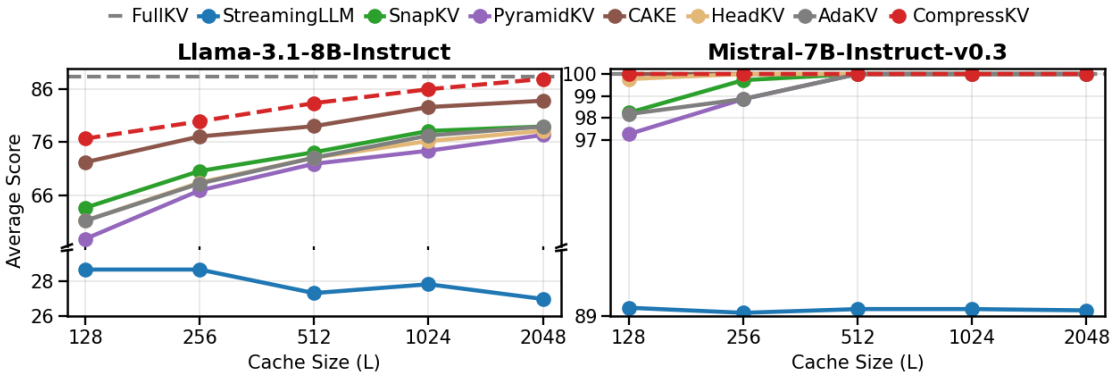

Figure 5: Average performance on the NIAH benchmark under different KV cache budget settings, in comparison with baseline methods.

Table 2
Comparison between traditional Retrieval Heads (TRH) and Semantic Retrieval Heads (SRH). (a) Performance drop after masking top-k heads on NIAH. (b) LongBench average score when CompressKV uses TRH or SRH under the same KV-cache budget. Darker cells in (a) indicate larger drops.

Heads	Top-10	Top-20	Top-30
TRH	1.02	7.67	13.30
SRH	24.55	72.56	73.81

(a) Causal masking on NIAH.

Heads	Avg.	Δ
TRH	44.72	0.00
SRH	**44.96**	+0.24

(b) LongBench average score.

4.3. Semantic Retrieval Heads: Causal Ablation and Head-Agnostic Gains

Following the masking-based causal test of [27], we conduct targeted ablations on Mistral-7B-Instruct-v0.3. Specifically, we mask the top-k heads ($k \in \{10, 20, 30\}$) on the NIAH benchmark. Table 2 (a) reports the resulting performance drop and compares Semantic Retrieval Heads against traditional Retrieval Heads (TRH). While masking TRH leads to only a minor degradation, masking even a small subset of Semantic Retrieval Heads yields a substantial drop in retrieval accuracy and markedly increases hallucinations, highlighting their critical role in faithful retrieval and localization of supporting evidence. CompressKV is compatible with heterogeneous head definitions. Table 2 (b) compares CompressKV using TRH vs. SRH on Mistral-7B-Instruct-v0.3 under a fixed per-layer KV budget of 256 tokens. SRH yields a modest yet consistent average gain over TRH (+0.24). Moreover, even with TRH and without dynamic budget allocation, CompressKV still surpasses most representative baselines (Table 1), evidencing more precise salient-token selection.

4.4. Memory and Latency under Long-Context Scaling

We evaluate end-to-end latency, decoding latency, time to first token, peak GPU memory, and throughput on Llama-3.1-8B-Instruct with FlashAttention-2 [34] on a single NVIDIA A100, sweeping context length from 4K to 128K with a fixed generation length of 1024. We compare CompressKV with a full-cache baseline and six KV-cache eviction methods under a 1024-token KV budget (except full cache). Figure 6 shows that end-to-end latency and time-to-first-token increase with context length for all methods, while eviction-based approaches keep decoding latency nearly constant; in contrast, full-cache decoding latency grows with context length. Under the fixed KV budget, eviction methods have similar peak memory, whereas full cache uses substantially more memory at long contexts.

4.5. Ablation Studies

To evaluate the effectiveness of each part in CompressKV, we conduct a series of ablation studies on the LongBench benchmark using Mistral-7B-Instruct-v0.3 with a fixed KV cache budget of 256.

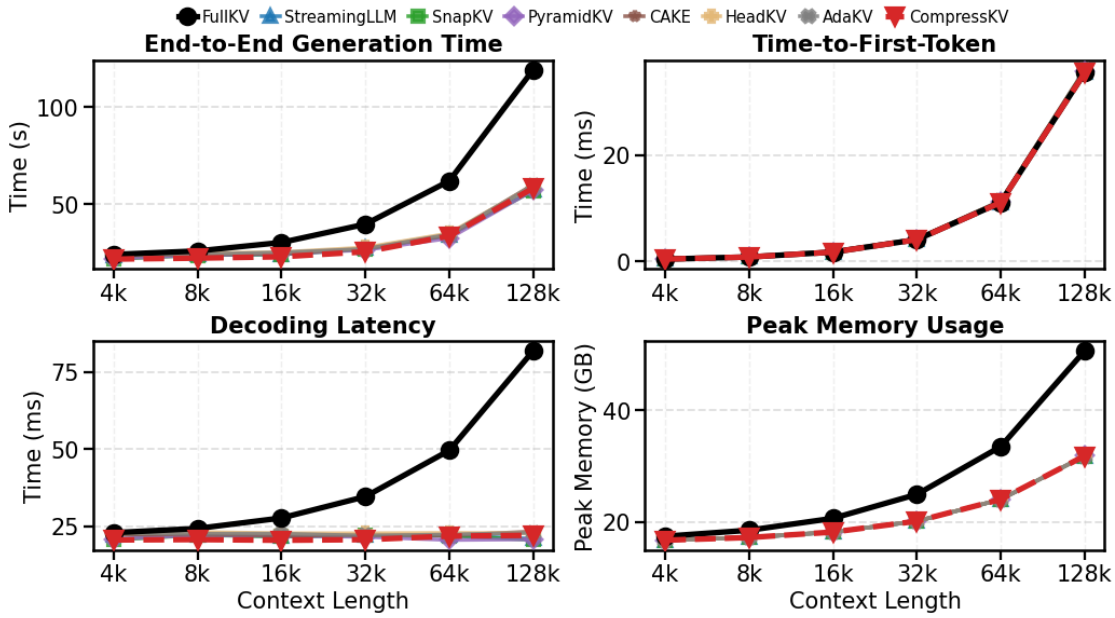

Figure 6: Comprehensive evaluation of inference efficiency on a single NVIDIA A100 GPU.

Table 3
Component analysis on Mistral-7B-Instruct-v0.3 under a fixed KV-cache budget of 256.

Method	Acc. (%)
SnapKV	43.76
+ SRH Selection	44.96
+ SRH + Layer Alloc.	**45.43**

(a) Contribution of each component.

SRHs per Layer	Mean Acc. (%)	Δ vs. Top-4
Top-2	44.33	−0.63
Top-4	**44.96**	0.00
Top-6	44.79	−0.17
Top-12	44.96	0.00
Top-24	44.30	−0.66

(b) Effect of the number of SRHs per layer.

Token selection and layer-wise cache allocation. We ablate SRHs–driven token selection and layer-aware budget allocation on Table 3 (a). Adding our selection to SnapKV improves accuracy; adding layer-aware allocation yields further gains—both components are complementary.

Number of selected heads per layer. We sweep SRH per layer from 2 to 24 (Table 3 (b)). Accuracy peaks at 4 and saturates thereafter (Top-6: −0.17; Top-12: 0.00), with Top-24 slightly worse; thus, 4 heads per layer suffice.

4.6. Orthogonal to Prior Efficiency Methods

With Prefilling Acceleration. CompressKV can be integrated with prefilling-stage accelerators such as MInference [35] and XAttention [36], as they target prefilling cost while CompressKV targets decoding-stage KV-cache memory. We conduct the integration experiments on Mistral-7B-Instruct-v0.3 under a 2048-token per-layer KV-cache budget. As shown in Figure 7, the combined variants maintain accuracy close to the prefilling-only baselines while further reducing decoding memory.

With KV-Cache Quantization. CompressKV also complements KV-cache quantization methods such as KIVI [6]: KIVI reduces KV precision, whereas CompressKV prunes less critical tokens while preserving full-precision KV entries. On Mistral-7B-Instruct-v0.3 with a 2048-token per-layer KV-cache budget, Figure 7 shows that 2-bit KIVI slightly outperforms CompressKV at comparable memory usage, but degrades sharply under 1-bit quantization. In contrast, CompressKV remains robust, and combining

it with 2-bit KIVI further reduces KV memory to about 1.6% of the 16-bit full-cache baseline while maintaining strong accuracy.

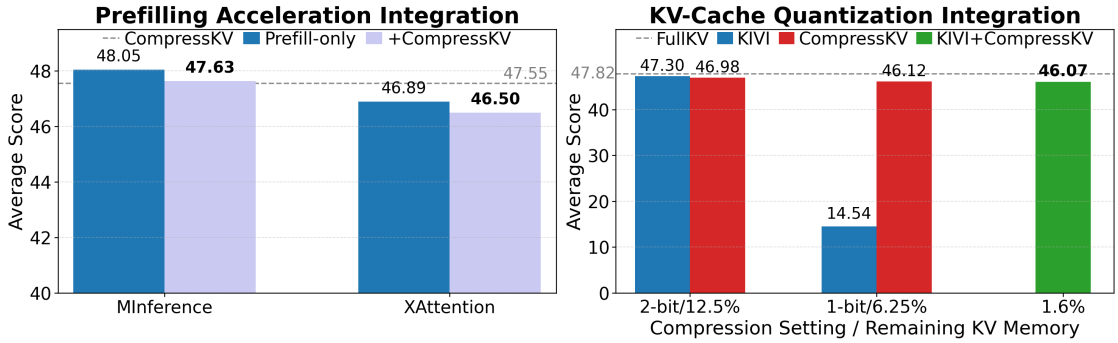

Figure 7: Integration of CompressKV with existing efficiency techniques on Mistral-7B-Instruct-v0.3. Left: integration with prefilling-stage accelerators; the dashed line denotes standalone CompressKV. Right: integration with KV-cache quantization; the dashed line denotes 16-bit FullKV.

With Head-Level Allocation. CompressKV can also be combined with head-level budget allocation methods such as HeadKV [16] and AdaKV [17]. We evaluate these integrations on LLaMA-3.1-8B-Instruct under different KV-cache budgets. Integrating our token selection with HeadKV yields HeadCompressKV, while combining our token selection with error-aware layer-wise allocation on AdaKV yields AdaCompressKV. As shown in Figure 8, both variants consistently improve performance across KV-cache budgets, achieving gains of up to nearly 2 points on LongBench and 11 points on Needle-in-a-Haystack under tight memory.

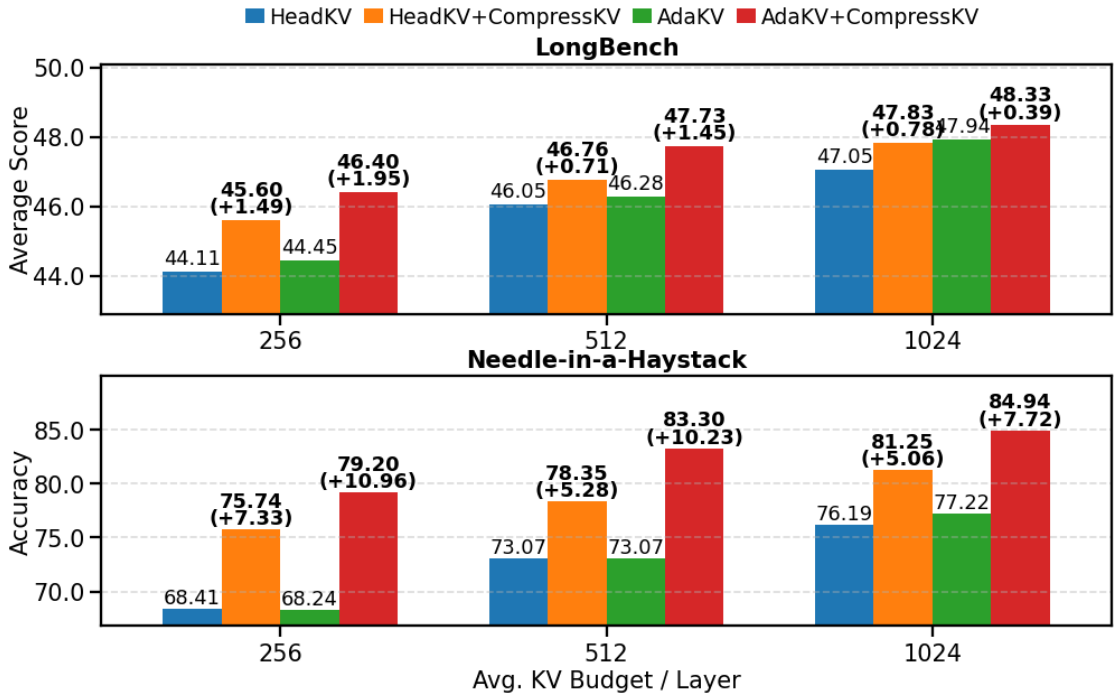

Figure 8: Integration of CompressKV with head-level allocation methods on Llama-3.1-8B-Instruct

4.7. Head Visualization

In Figures 9, we present a comparison between traditional Retrieval Heads and Semantic Retrieval Heads identified using Mistral-7B-Instruct-v0.3. All scores are L1-normalized across the attention

head importance distributions. Unlike traditional methods that require exact top-k attention hits, our approach aggregates scores over entire answer spans, capturing heads that contribute semantically relevant context even when they never achieve top-1 attention for individual tokens. For instance, as shown in Figure 9, layers 0 and 1 of the Mistral model have zero scores for all heads using the traditional method, whereas our approach successfully identifies heads of lower yet meaningful importance.

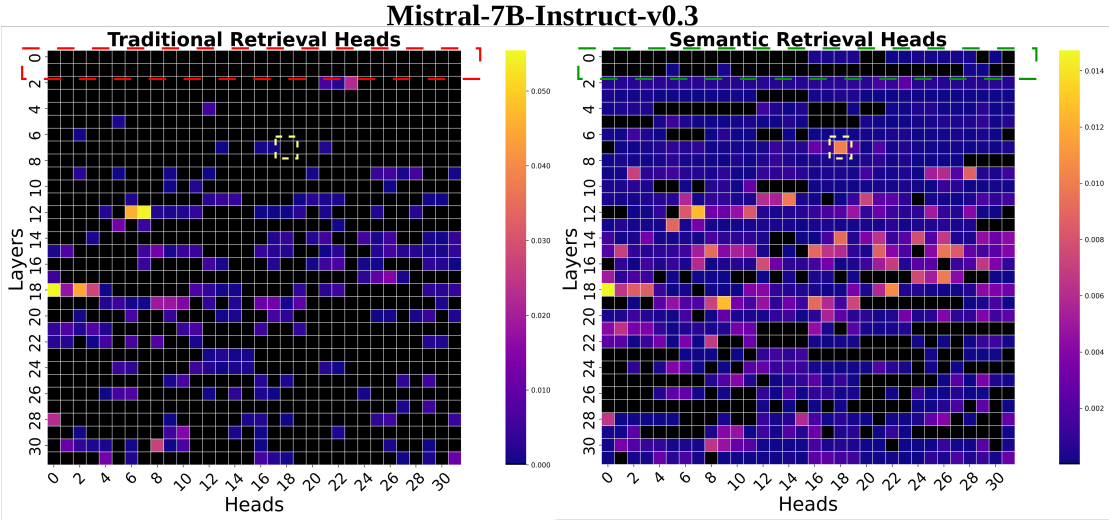

Figure 9: Head visualization for Mistral-7B-Instruct-v0.3. Left: Traditional Retrieval Heads. Right: Semantic Retrieval Heads identified.

5. Conclusion

We presented CompressKV, a KV-cache compression framework for GQA-based LLMs that improves the resource–performance trade-off of long-context inference. CompressKV identifies Semantic Retrieval Heads to avoid streaming-head-dominated token eviction and uses offline layer-wise eviction errors to allocate cache budgets adaptively across layers. Experiments on LongBench and Needle-in-a-Haystack across multiple models and cache budgets show that CompressKV consistently preserves accuracy under tight KV-cache memory constraints. These results demonstrate that retrieval-aware token selection and error-aware allocation provide an effective path toward more memory-efficient and sustainable long-context LLM inference.

Acknowledgement

This work is funded by the European Union - European Research Council (ERC) Starting Grant - Project-ID 101219243. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT (OpenAI) to assist with grammar improvement, language polishing, and manuscript editing. After using this tool, the authors carefully reviewed and revised all content and take full responsibility for the accuracy, originality, and integrity of the publication.

References

[1] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al., Gpt-4 technical report, 2024. URL: https://arxiv.org/abs/2303.08774. arXiv:2303.08774.

[2] Anthropic, The Claude 3 Model Family: Opus, Sonnet, Haiku, Technical Report, Anthropic, 2024. URL: https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf, accessed: 2024-07-09.

[3] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, et al., The llama 3 herd of models, 2024. URL: https://arxiv.org/abs/2407.21783. arXiv:2407.21783.

[4] B. Hui, J. Yang, Z. Cui, J. Yang, D. Liu, L. Zhang, T. Liu, J. Zhang, B. Yu, K. Lu, et al., Qwen2.5 technical report, 2025. URL: https://arxiv.org/abs/2412.15115. arXiv:2412.15115.

[5] J. Wang, Y.-G. Chen, I.-C. Lin, B. Li, G. L. Zhang, Basis sharing: Cross-layer parameter sharing for large language model compression, in: The Thirteenth International Conference on Learning Representations, 2025. URL: https://openreview.net/forum?id=gp32jvUquq.

[6] Z. Liu, J. Yuan, H. Jin, S. Zhong, Z. Xu, V. Braverman, B. Chen, X. Hu, KIVI: A tuning-free asymmetric 2bit quantization for KV cache, in: Forty-first International Conference on Machine Learning, 2024. URL: https://openreview.net/forum?id=L057s2Rq8O.

[7] H. Kang, Q. Zhang, S. Kundu, G. Jeong, Z. Liu, T. Krishna, T. Zhao, Gear: An efficient kv cache compression recipe for near-lossless generative inference of llm, 2024. URL: https://arxiv.org/abs/2403.05527. arXiv:2403.05527.

[8] S. Ge, Y. Zhang, L. Liu, M. Zhang, J. Han, J. Gao, Model tells you what to discard: Adaptive kv cache compression for llms, in: The Thirteenth International Conference on Learning Representations, 2024. URL: https://openreview.net/pdf?id=uNrFpDPMyo.

[9] G. Xiao, Y. Tian, B. Chen, S. Han, M. Lewis, Efficient streaming language models with attention sinks, in: The Twelfth International Conference on Learning Representations, 2024. URL: https://openreview.net/forum?id=NG7sS51zVF.

[10] Y. Li, Y. Huang, B. Yang, B. Venkitesh, A. Locatelli, H. Ye, T. Cai, P. Lewis, D. Chen, SnapKV: LLM knows what you are looking for before generation, in: The Thirty-eighth Annual Conference on Neural Information Processing Systems, 2024. URL: https://openreview.net/forum?id=poE54GOq2l.

[11] Z. Cai, Y. Zhang, B. Gao, Y. Liu, Y. Li, T. Liu, K. Lu, W. Xiong, Y. Dong, J. Hu, W. Xiao, Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling, 2025. URL: https://arxiv.org/abs/2406.02069. arXiv:2406.02069.

[12] D. Yang, X. Han, Y. Gao, Y. Hu, S. Zhang, H. Zhao, Pyramidinfer: Pyramid kv cache compression for high-throughput llm inference, in: Findings of the Association for Computational Linguistics ACL 2024, 2024, pp. 3258–3270.

[13] Z. Qin, Y. Cao, M. Lin, W. Hu, S. Fan, K. Cheng, W. Lin, J. Li, CAKE: Cascading and adaptive KV cache eviction with layer preferences, in: The Thirteenth International Conference on Learning Representations, 2025. URL: https://openreview.net/forum?id=EQgEMAD4kv.

[14] J. Ainslie, J. Lee-Thorp, M. de Jong, Y. Zemlyanskiy, F. Lebron, S. Sanghai, Gqa: Training generalized multi-query transformer models from multi-head checkpoints, in: The 2023 Conference on Empirical Methods in Natural Language Processing, 2023. URL: https://openreview.net/forum?id=hmOwOZWzYE.

[15] G. Xiao, J. Tang, J. Zuo, junxian guo, S. Yang, H. Tang, Y. Fu, S. Han, Duoattention: Efficient long-context LLM inference with retrieval and streaming heads, in: The Thirteenth International Conference on Learning Representations, 2025. URL: https://openreview.net/forum?id=cFu7ze7xUm.

[16] Y. Fu, Z. Cai, A. Asi, W. Xiong, Y. Dong, W. Xiao, Not all heads matter: A head-level kv cache compression method with integrated retrieval and reasoning, in: The Twelfth International Conference on Learning Representations, 2025. URL: https://arxiv.org/abs/2410.19258.

[17] Y. Feng, J. Lv, Y. Cao, X. Xie, S. K. Zhou, Ada-kv: Optimizing kv cache eviction by adaptive budget allocation for efficient llm inference, 2025. URL: https://arxiv.org/abs/2407.11550.

arXiv:2407.11550.

[18] Z. Liu, A. Desai, F. Liao, W. Wang, V. Xie, Z. Xu, A. Kyrillidis, A. Shrivastava, Scissorhands: Exploiting the persistence of importance hypothesis for LLM KV cache compression at test time, in: Thirty-seventh Conference on Neural Information Processing Systems, 2023. URL: https://openreview.net/forum?id=JZfg6wGi6g.

[19] Z. Zhang, Y. Sheng, T. Zhou, T. Chen, L. Zheng, R. Cai, Z. Song, Y. Tian, C. Re, C. Barrett, Z. Wang, B. Chen, H2o: Heavy-hitter oracle for efficient generative inference of large language models, in: Thirty-seventh Conference on Neural Information Processing Systems, 2023. URL: https://openreview.net/forum?id=RkRrPp7GKO.

[20] C. Han, Q. Wang, H. Peng, W. Xiong, Y. Chen, H. Ji, S. Wang, 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), 2024, pp. 3991–4008.

[21] M. Oren, M. Hassid, N. Yarden, Y. Adi, R. Schwartz, Transformers are multi-state rnns, in: Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, 2024, pp. 18724–18741.

[22] Z. Wan, X. Wu, Y. Zhang, Y. Xin, C. Tao, Z. Zhu, X. Wang, S. Luo, J. Xiong, L. Wang, M. Zhang, D2o: Dynamic discriminative operations for efficient long-context inference of large language models, in: The Thirteenth International Conference on Learning Representations, 2025. URL: https://openreview.net/forum?id=HzBfoUdjHt.

[23] C. Olsson, N. Elhage, N. Nanda, N. Joseph, N. DasSarma, T. Henighan, B. Mann, A. Askell, Y. Bai, A. Chen, T. Conerly, D. Drain, D. Ganguli, Z. Hatfield-Dodds, D. Hernandez, S. Johnston, A. Jones, J. Kernion, L. Lovitt, K. Ndousse, D. Amodei, T. Brown, J. Clark, J. Kaplan, S. McCandlish, C. Olah, In-context learning and induction heads, 2022. URL: https://arxiv.org/abs/2209.11895. arXiv:2209.11895.

[24] W. Kwon, S. Kim, M. W. Mahoney, J. Hassoun, K. Keutzer, A. Gholami, A fast post-training pruning framework for transformers, in: A. H. Oh, A. Agarwal, D. Belgrave, K. Cho (Eds.), Advances in Neural Information Processing Systems, 2022. URL: https://openreview.net/forum?id=0GRBKLBjJE.

[25] Z. Zheng, Y. Wang, Y. Huang, S. Song, M. Yang, B. Tang, F. Xiong, Z. Li, Attention heads of large language models: A survey, 2024. URL: https://arxiv.org/abs/2409.03752. arXiv:2409.03752.

[26] J. Ren, Q. Guo, H. Yan, D. Liu, Q. Zhang, X. Qiu, D. Lin, Identifying semantic induction heads to understand in-context learning, in: Findings of the Association for Computational Linguistics: ACL 2024, 2024. URL: https://aclanthology.org/2024.findings-acl.412/.

[27] W. Wu, Y. Wang, G. Xiao, H. Peng, Y. Fu, Retrieval head mechanistically explains long-context factuality, in: The Thirteenth International Conference on Learning Representations, 2025. URL: https://openreview.net/forum?id=EytBpUGB1Z.

[28] E. Todd, M. Li, A. S. Sharma, A. Mueller, B. C. Wallace, D. Bau, Function vectors in large language models, in: The Twelfth International Conference on Learning Representations, 2024. URL: https://openreview.net/forum?id=AwyxtyMwaG.

[29] K. Yin, J. Steinhardt, Which attention heads matter for in-context learning?, 2025. URL: https://arxiv.org/abs/2502.14010. arXiv:2502.14010.

[30] H. Tang, Y. Lin, J. Lin, Q. Han, S. Hong, Y. Yao, G. Wang, Razorattention: Efficient kv cache compression through retrieval heads, in: The Twelfth International Conference on Learning Representations, 2025. URL: https://arxiv.org/abs/2407.15891.

[31] D. Jiang, Y. Liu, S. Liu, J. Zhao, H. Zhang, Z. Gao, X. Zhang, J. Li, H. Xiong, From clip to dino: Visual encoders shout in multi-modal large language models, 2024. URL: https://arxiv.org/abs/2310.08825. arXiv:2310.08825.

[32] Y. Bai, X. Lv, J. Zhang, H. Lyu, J. Tang, Z. Huang, Z. Du, X. Liu, A. Zeng, L. Hou, Y. Dong, J. Tang, J. Li, 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), 2024. URL: https://aclanthology.org/2024.acl-long.172/.

[33] G. Kamradt, Needleinahaystack, https://github.com/gkamradt/LLMTest_NeedleInAHaystack, 2023.

Accessed: 2025-07-13.

[34] T. Dao, Flashattention-2: Faster attention with better parallelism and work partitioning, in: The Twelfth International Conference on Learning Representations, 2024. URL: https://openreview.net/forum?id=mZn2Xyh9Ec.

[35] H. Jiang, Y. LI, C. Zhang, Q. Wu, X. Luo, S. Ahn, Z. Han, A. H. Abdi, D. Li, C.-Y. Lin, Y. Yang, L. Qiu, MInference 1.0: Accelerating pre-filling for long-context LLMs via dynamic sparse attention, in: The Thirty-eighth Annual Conference on Neural Information Processing Systems, 2024. URL: https://openreview.net/forum?id=fPBACAbqSN.

[36] R. Xu, G. Xiao, H. Huang, J. Guo, S. Han, XAttention: Block sparse attention with antidiagonal scoring, in: Forty-second International Conference on Machine Learning, 2025. URL: https://openreview.net/forum?id=KG6aBfGi6e.