LAB-SPARSE: LOCAL ACCUMULATIVE BOOST FOR ADAPTIVE SEMANTIC-BLOCK RETENTION IN KV CACHE

Anonymous authors
Paper under double-blind review

ABSTRACT

The expanding context windows of large language models (LLMs) have brought substantial gains in reasoning and comprehension, yet this expansion comes at the cost of a linearly growing Key-Value (KV) cache that strains GPU memory and increases decoding latency.Existing KV sparsification methods, whether heuristic token-level eviction or fixed-size block retention, share a common limitation: they treat the KV cache as a collection of isolated units. This overlooks a fundamental property of natural language—semantic information is typically organized as continuous spans rather than isolated tokens. Fixed-size blocks are particularly problematic: a coherent semantic unit may be split across multiple blocks, and averaging scores within a block can dilute or filter out critical tokens. Consequently, these methods often suffer from fragmented critical information and degraded continuity in multi-hop reasoning and long-context retrieval.

To address this, we propose LAB-Sparse, a dynamic continuous block retention method powered by Local Accumulative Boost (LAB). Our core insight is to elevate the basic unit of sparsification from individual tokens to **dynamic-length contiguous blocks**, where block lengths are determined by contiguous runs above a quantile-based threshold, which implicitly reflect local importance density. The LAB mechanism applies a neighborhood cumulative gain to smooth token-level importance scores, thereby boosting tokens surrounded by high-importance neighbors and suppressing isolated noise. On this enhanced distribution, a quantile-based interval delineation strategy identifies high-value contiguous regions, followed by a greedy block-wise retention policy that prioritizes complete contiguous blocks under a fixed KV budget.

Experiments on Llama-3.1-8B, Qwen2.5-7B, and Mistral-7B across LongBench and RULER show that LAB-Sparse achieves competitive or superior average accuracy compared with strong baselines such as H2O, SnapKV, StreamingLLM, ChunkKV, and PyramidKV, while remaining close to full attention in the 4K–16K range. Under a fixed KV budget of 512, it matches the decoding throughput and memory savings of block-based baselines at 32K ($3.03\times$ and $\sim$12.6%), while improving average accuracy over token-level methods. Anonymous code will be submitted as supplementary material.

1 INTRODUCTION

As the context windows of large language models (LLMs) expand to hundreds of thousands of tokens (Dubey et al., 2024; Yang et al., 2024; Ding et al., 2024), their capabilities in long-document understanding and complex reasoning have advanced remarkably. However, this progress comes at a significant inference cost: the Key-Value (KV) cache of historical tokens grows as $O(L)$ with sequence length L, rendering memory bandwidth the primary bottleneck (Dao et al., 2022) and causing high decoding latency or even out-of-memory errors. Existing KV sparsification methods can be broadly grouped into three categories. Token-level eviction strategies, such as H2O (Zhang et al., 2023), SnapKV (Li et al., 2024b), and StreamingLLM (Xiao et al., 2024), retain tokens with high attention scores but often disrupt continuity—complete entities or logical clauses may be truncated. Block-based methods, such as ChunkKV (Liu et al., 2025) and PyramidKV (Cai et al., 2025),

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

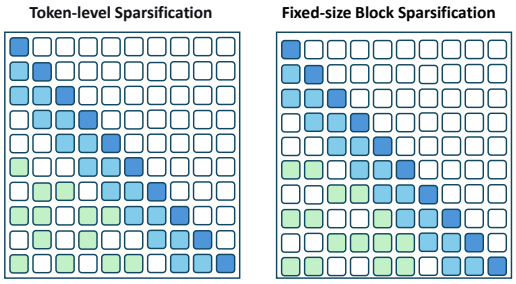


Figure 1: Comparison of token-level sparsification and fixed-size block sparsification. Token-level methods retain isolated tokens, fragmenting continuous semantic units. Fixed-size blocks may split a coherent unit across multiple blocks or dilute critical tokens through intra-block averaging.Keep the most recent window for both.

operate on fixed-size blocks, yet their block sizes cannot adapt to the non-uniform distribution of information density. Learning-driven methods introduce adaptivity through trainable modules, at the cost of additional training. Taken together, existing methods fall short on two coupled decisions: which positions to retain, and how long the retained spans should be.

A key reason for this shortfall is the granularity at which sparsification is performed. As illustrated in Figure 1, token-level methods retain isolated units, fragmenting a continuous semantic span into scattered pieces; fixed-size block methods, on the other hand, may split one coherent semantic unit across multiple blocks, or dilute the importance of critical tokens when averaging scores within a block. Both cases overlook a fundamental property of natural language: semantic information is typically organized as continuous spans rather than isolated tokens. Consequently, the retained KV cache loses local continuity, which is particularly harmful for multi-hop reasoning and long-context retrieval that rely on complete evidence spans.

To address this, we propose **LAB-Sparse**, a dynamic continuous block retention method powered by Local Accumulative Boost (LAB). Our core observation is that semantic information in natural language is often organized as continuous spans rather than isolated tokens. Therefore, we elevate the basic unit of sparsification from individual tokens to **dynamic-length contiguous blocks** that better preserve local continuity. Specifically, the LAB mechanism applies a neighborhood cumulative gain to smooth token-level importance scores, thereby boosting tokens surrounded by high-importance neighbors and suppressing isolated noise. Based on this enhanced distribution, a quantile-based interval delineation strategy identifies high-value contiguous regions, followed by a greedy block-wise retention policy that prioritizes complete contiguous blocks under a fixed KV budget. The decision is executed during the prefill stage, with even layers computing the retention indices and odd layers reusing them, and the resulting indices are shared across all decoding steps. Figure 2 illustrates the overall pipeline.

The main contributions of this paper are summarized as follows:

- **A novel sparsification paradigm.** We elevate the fundamental unit of KV sparsification from token-level eviction to **dynamic-length contiguous block retention**, providing a granularity that better preserves local continuity.
- **The Local Accumulative Boost (LAB) mechanism.** We design a neighborhood cumulative gain-based enhancement that mitigates local fluctuations of raw attention scores and yields a stable importance distribution for interval delineation.
- **Quantile-based interval delineation and greedy block selection.** Our dynamic delineation determines the number and length of retained blocks from contiguous runs above a quantile threshold, which implicitly reflect local importance density, balancing flexibility with contiguity.
- **Comprehensive empirical evaluation.** Experiments on Llama-3.1-8B (Dubey et al., 2024), Qwen2.5-7B (Yang et al., 2024), and Mistral-7B (Jiang et al., 2023) across LongBench (Bai et al., 2024) and RULER (Hsieh et al., 2024) demonstrate that LAB-Sparse achieves competitive or superior average accuracy compared with strong baselines, while

maintaining performance close to full attention on several tasks. Ablation studies further confirm the contribution of the LAB mechanism. Under a fixed KV budget of 512, it delivers up to $3.0\times$ decoding throughput improvement over the chunked-attention full-cache baseline and reduces peak memory by approximately 12.6% at 32K context length.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 details LAB-Sparse. Section 4 reports experiments, and Section 5 concludes.

2 Related Work

KV cache compression is a central challenge for efficient large language model (LLM) inference, attracting substantial attention (Li et al., 2024a). Existing approaches fall into three categories: heuristic token-level eviction, block-based sparsification, and learning-driven sparsification.

Heuristic-Based Token-Level KV Eviction. Early methods use heuristics for token-level retention or eviction. H_2O (Zhang et al., 2023) observes that a few tokens contribute most attention scores ("heavy hitters") and proposes a dynamic eviction strategy retaining recent tokens while prioritizing them. SnapKV (Li et al., 2024b) reveals that each head focuses on specific prompt attention features captured from an observation window at the prompt end. StreamingLLM (Xiao et al., 2024) identifies the "attention sink" phenomenon—retaining initial-token KV largely recovers window attention performance—and retains only initial tokens and a recent KV window, enabling long-sequence modeling. Beyond these, FastGen (Ge et al., 2024a) profiles attention to select per-head compression policies adaptively, Scissorhands (Liu et al., 2023) and TOVA (Barbero et al., 2024) exploit token importance persistence to drop low-value tokens during decoding, and Quest (Tang et al., 2024) performs query-aware page selection at inference.

Block-Based KV Sparsification. To mitigate continuity degradation, recent work explores chunk/block-level sparsification. ChunkKV (Liu et al., 2025) groups consecutive tokens into chunks as the compression unit, retaining informative chunks while discarding unimportant ones, and proposes cross-layer index reuse to reduce overhead. PyramidKV (Cai et al., 2025) observes a "pyramidal information funneling" phenomenon—attention scatters in lower layers and consolidates in higher layers—and dynamically adjusts cache sizes across layers, retaining only 12% of the KV cache while matching full-attention performance.

Learning-Driven KV Sparsification. Another line introduces learnable modules replacing handcrafted heuristics. SP-KV (Szilvasy et al., 2026) introduces a learnable sparse-write mechanism that predicts future KV utility via a lightweight per-head predictor, writing a KV pair into the persistent cache only if its predicted utility exceeds a threshold. DMS (Łańcucki et al., 2025) requires only 1K training steps to achieve $8\times$ compression, delaying token eviction to implicitly merge representations and preserve critical information. Ada-KV (Feng et al., 2025) establishes a theoretical loss bound between pre- and post-eviction attention and proposes the first head-wise adaptive budget allocation, offering plug-and-play benefits for prior eviction methods. STAR-KV (Bhatnagar et al., 2026) learns rank allocation from data via soft thresholding, enabling fine-grained rank control across heads and decoder blocks, reducing the KV cache by up to $20\times$. Orthogonally, quantization methods such as KIVI (Liu et al., 2024b) and KVQuant (Hooper et al., 2024) reduce per-token memory while retaining all tokens, and MInference (Jiang et al., 2024) accelerates prefill via dynamic sparse attention. Prompt-compression approaches, including ICAE (Ge et al., 2024b) and Gist tokens (Mu et al., 2023), encode long context into compact latent representations but require additional training.

Summary and Gap Analysis. A common deficiency is sparsification granularity. Token-level heuristic methods (H_2O (Zhang et al., 2023), SnapKV (Li et al., 2024b), StreamingLLM (Xiao et al., 2024)) operate on individual tokens, often disrupting continuity—complete entities, logical clauses, or math expressions may be truncated, harming multi-hop reasoning and long-context retrieval (Liu et al., 2024a). Block-based methods (ChunkKV (Liu et al., 2025), PyramidKV (Cai et al., 2025)) partially address this by operating on contiguous groups, yet their fixed block sizes cannot adapt to non-uniform information density. Learning-driven methods (SP-KV (Szilvasy et al., 2026), DMS (Łańcucki et al., 2025), Ada-KV (Feng et al., 2025), STAR-KV (Bhatnagar et al., 2026)) introduce adaptivity through trainable modules, yet still operate primarily at the token level and require additional training. In contrast, our method is **training-free** and operates on adaptively-sized contiguous

blocks. We propose **Local Accumulative Boost (LAB)**, which identifies and retains dynamically-sized contiguous blocks without training, elevating the fundamental unit of sparsification from isolated tokens or fixed chunks to contiguous intervals that better align with local importance density.

3 METHODOLOGY

3.1 PRELIMINARIES

KV Cache Bottleneck. For a sequence of length L, autoregressive generation requires caching the key and value vectors of all previous tokens, known as the KV cache. Its size grows linearly with L, making memory bandwidth the primary bottleneck for long-sequence inference and motivating the need for cache compression.

Problem Formulation. Let $\mathcal{C} = \{(\mathbf{k}_i, \mathbf{v}_i)\}_{i=1}^{L}$ denote the full KV cache, where $\mathbf{k}_i, \mathbf{v}_i \in \mathbb{R}^d$ are the key and value vectors of the i-th token. Given a fixed budget M ($M < L$), the goal is to select a subset $\mathcal{S} \subset \{1, \ldots, L\}$ with $|\mathcal{S}| = M$ that minimizes the degradation in generation quality:

$$\mathcal{S}^* = \arg \min_{\mathcal{S}, |\mathcal{S}|=M} \mathcal{J}(\mathcal{C}, \mathcal{C}_{\mathcal{S}}), \tag{1}$$

where $\mathcal{C}_{\mathcal{S}} = \{(\mathbf{k}_i, \mathbf{v}_i) \mid i \in \mathcal{S}\}$ is the retained cache. In practice, we perform a block-wise retention decision per even layer during the prefill stage, approximating this objective by retaining tokens that are most important for future attention.

Token Importance Scores. We define the raw importance score s_i of the i-th token as the attention weight it receives at the last query step, averaged across heads:

$$s_i = \frac{1}{H} \sum_{h=1}^{H} \text{Attn}_{h,\text{last},i}, \tag{2}$$

where H is the number of heads. We normalize s_i to $[0, 1]$ with $\epsilon = 10^{-8}$ for numerical stability. This normalized score serves as the input to LAB enhancement.

3.2 LAB-SPARSE: DYNAMIC BLOCK-WISE RETENTION

We propose **LAB-Sparse**, a method that elevates the fundamental unit of sparsification from individual tokens to *dynamic-length contiguous blocks*. As illustrated in Figure 2, the pipeline consists of three stages: (1) **Local Accumulative Boost (LAB)** to perform context-aware smooth enhancement of token-level importance scores; (2) **adaptive interval delineation** to identify high-value contiguous regions; and (3) **greedy block-wise retention** to prioritize complete contiguous blocks under a fixed KV budget. The decision is executed during the prefill stage, with even layers computing the retention indices and odd layers reusing them; the resulting indices are fixed and reused across all decoding steps.

3.2.1 LOCAL ACCUMULATIVE BOOST (LAB)

Raw importance scores s_i often exhibit high-frequency fluctuations that do not align with semantic boundaries. We introduce **Local Accumulative Boost (LAB)** to enhance tokens whose neighborhoods exhibit high accumulated importance, motivated by our diagnostic observation that high-importance tokens cluster into contiguous spans (Appendix G). For token i, we define a neighborhood of radius r and compute the enhanced score:

$$\tilde{s}_i = s_i + \alpha \sum_{j \in \mathcal{N}(i)} \left(1 - \frac{|i-j|}{r+1}\right) \cdot s_j, \tag{3}$$

where $\mathcal{N}(i) = \{j \mid 0 < |i-j| \leq r\}$ and $\alpha > 0$ controls the contribution of neighbor information. The distance decay weight gives larger contributions to closer neighbors. This mechanism boosts tokens surrounded by high-importance neighbors while suppressing isolated noise, yielding a smoother importance distribution that better reflects local importance density.

In practice, the neighborhood weighted sum is efficiently computed via 1D convolution: a kernel of length $2r + 1$ is constructed with zero weight at the center and weights $1 - \frac{u}{r+1}$ (or 1) at offsets

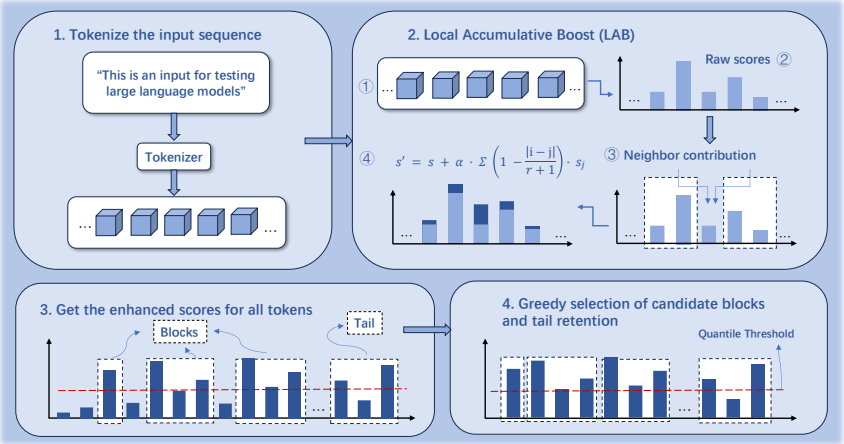


Figure 2: Overview of LAB-Sparse. Given an input sequence, we compute raw token importance scores, apply LAB to obtain enhanced scores, delineate adaptive intervals, and greedily select complete blocks to form the compressed KV cache. The entire process runs once at the end of prefill per even layer.

$u \in [1, r]$; convolving the raw scores with this kernel gives the neighbor sum, which is then added to the raw scores. The complexity is $O(L \cdot r)$ with a small constant r.

Hyperparameter choices. We set the neighborhood radius $r = 3$ and the scaling factor $\alpha = 0.2$ in all experiments. The choice of $r = 3$ balances two considerations: a larger radius would oversmooth the importance distribution and blur semantic boundaries, while a smaller radius would provide insufficient neighborhood context. The value of α controls the relative contribution of neighbor information to the final score; we adopt $\alpha = 0.2$ as a conservative default that preserves the dominant role of each token's own attention score while still incorporating neighborhood context. We leave a systematic sensitivity analysis to future work.

3.2.2 Adaptive Interval Delineation

Given the enhanced distribution $\{\tilde{s}_i\}_{i=1}^{L}$, we identify contiguous intervals whose raw normalized scores exceed a quantile-based threshold. With a quantile ratio q (corresponding to `lab_threshold_ratio`, e.g., $q = 0.7$), the threshold is:

$$\tau = \text{Quantile}\left(\{\tilde{s}_i\}_{i=1}^{L}, q\right). \tag{4}$$

We then mark positions where the **raw normalized score** $s_i \geq \tau$. Note that τ is computed from the enhanced distribution, while the comparison uses raw normalized scores. The enhanced scores are not re-normalized before quantile computation. This design uses the enhancement only to determine the threshold level, and the final retention decision is still based on the original attention contribution.This cross-distribution comparison is empirically stable: sweeping the quantile ratio $q \in [0.5, 0.8]$ (Table 5, Appendix A) varies PG19 perplexity within 2% of the default, indicating that τ is insensitive to the choice of q and to the raw-vs-enhanced distribution shift.

Contiguous marked positions are merged into candidate blocks. Specifically, we scan the boolean array, extract all runs of True, and filter out fragments shorter than a minimum block length `min_block_len` (set to 2 in our implementation). The importance of a candidate block B is defined as the average enhanced score of its tokens:

$$R_B = \frac{1}{|B|} \sum_{i \in B} \tilde{s}_i. \tag{5}$$

Using the average rather than the sum prevents long blocks from dominating solely by length, making block importance reflect average importance. All candidate blocks are sorted in descending order of R_B for the subsequent greedy selection.When the budget is sufficiently large, the number and length of retained blocks are primarily determined by the quantile threshold; when the budget becomes the binding constraint, the greedy policy retains tokens up to the budget limit.

Algorithm 1 LAB-Sparse: Dynamic Block-wise KV Retention (executed per even layer during prefill)

Input: Full KV cache $\mathcal{C}$, budget M, tail length T, window radius r, scaling α, quantile q, min block length m.
Output: Retained KV cache $\mathcal{C}_{\mathcal{S}}$.
Compute raw importance scores s_i and normalize to $[0, 1]$.
Apply LAB: $\tilde{s}_i = s_i + \alpha \sum_{j \in \mathcal{N}(i)} (1 - \frac{|i-j|}{r+1}) s_j$.
Compute threshold $\tau = \text{Quantile}(\{\tilde{s}_i\}, q)$.
Mark positions with $s_i \geq \tau$, merge contiguous runs into candidate blocks $\mathcal{B}$, filter blocks with length $< m$.
For each block $B \in \mathcal{B}$, compute $R_B = \frac{1}{|B|} \sum_{i \in B} \tilde{s}_i$.
Sort $\mathcal{B}$ by R_B descending.
Always retain the last T tokens as $\mathcal{T}$; if $|\mathcal{T}| > M$, keep top-M by score and return.
$B_{\text{rem}} = M - |\mathcal{T}|$.
for each block $B \in \mathcal{B}$ **do**
  if $|B| \leq B_{\text{rem}}$ **then**
    $\mathcal{S} \leftarrow \mathcal{S} \cup B$; $B_{\text{rem}} \leftarrow B_{\text{rem}} - |B|$.
  else
    Find a contiguous sub-segment of length B_{rem} in B maximizing the sum of $\tilde{s}$.
    Add the sub-segment to $\mathcal{S}$; $B_{\text{rem}} \leftarrow 0$; **break**.
  end if
end for
$\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{T}$; sort in ascending order.
return $\mathcal{C}_{\mathcal{S}} = \{(\mathbf{k}_i, \mathbf{v}_i) \mid i \in \mathcal{S}\}$.

3.2.3 Greedy Block-wise Retention

Given a fixed KV budget M (corresponding to `max_capacity`), we select a subset of candidate blocks such that the total number of retained tokens does not exceed M while preserving complete contiguous blocks. Additionally, to preserve local continuity, we always retain the last `keep_tail` tokens (corresponding to `window_size`), which directly form part of the retained set.

The procedure is as follows.

Tail retention. Take the last `keep_tail` tokens as the forced set $\mathcal{T}$. If `keep_tail` exceeds M, retain only the top-M tokens by importance and return.

Remaining budget. $B_{\text{rem}} = M - |\mathcal{T}|$.

Greedy block selection. Iterate over candidate blocks sorted by R_B descending:

- If the full block length $\leq B_{\text{rem}}$, add the entire block to $\mathcal{S}$ and update B_{rem}.
- Otherwise, enter **sub-segment selection**: within the current block, find a contiguous sub-segment of length B_{rem} that maximizes the sum of enhanced scores. This is efficiently solved via a sliding window using cumulative sums in $O(|B|)$ time. Add the selected sub-segment to $\mathcal{S}$ and terminate.
- If $B_{\text{rem}} = 0$, terminate.

Merge and sort. Merge the tail indices with the greedily selected indices, sort in ascending order to obtain the final index set $\mathcal{S}$. If the total exceeds M (edge case), truncate to M.

This greedy strategy prioritizes complete contiguous blocks and falls back to contiguous sub-segment filling when the remaining budget is insufficient, ensuring that semantic units are preserved as a whole while fully utilizing the budget. The complete algorithm is summarized in Algorithm 1.

3.2.4 Implementation Details and Complexity

Cross-layer decision sharing. To reduce computation and maintain consistency across layers, only even layers (`layer_idx % 2 == 0`) compute importance scores and perform LAB-Sparse se-

lection, while odd layers reuse the retention indices from the most recent even layer. This design is motivated by our diagnostic observation that adjacent layers exhibit moderate consistency in their judgments of low-importance tokens (Appendix G). Since adjacent layers exhibit correlated attention patterns, this sharing does not significantly affect performance while reducing the decision overhead.

Decision at prefill. The compression decision of our method is executed once per even layer during the prefill stage. Specifically, after the layer processes the input sequence and generates the full KV cache, we compute importance scores based on the attention distribution of the last query position, perform LAB enhancement, interval delineation, and greedy block selection to obtain the retention index set $\mathcal{S}$. Even layers compute the indices, and odd layers reuse the most recent even-layer indices. During autoregressive decoding, the model only uses the pruned KV cache for attention computation and does not trigger any new compression. This design ensures: (1) compression overhead occurs only during prefill, with no impact on decoding latency; (2) all decoding steps share the same compressed cache, ensuring generation stability; (3) the implementation is simple and avoids extra branching in the decoding loop.

Chunked attention. In the prefill stage, to reduce peak memory usage, we compute attention in chunks: the query sequence is split into fixed-size blocks (`chunked_attn_block_size`, e.g., 512), and attention outputs are computed block by block and written to the result tensor. Importance scores are computed only from the last query block's attention weights, ensuring alignment with the final decoding step's focus.

Complexity analysis. LAB enhancement uses 1D convolution with complexity $O(L \cdot r)$, where r is a small constant (typically 3), hence linear in L. Quantile threshold computation requires sorting or selection, taking $O(L \log L)$ or $O(L)$ depending on implementation. Continuous block extraction is a single linear scan, $O(L)$. Greedy selection involves sorting at most m candidate blocks, $O(m \log m)$ with $m \ll L$. Sub-segment selection uses a sliding window, $O(|B|)$, worst-case $O(L)$. Overall, the additional computation is negligible compared to the quadratic cost of full attention, and the method involves no trainable parameters, making it a plug-and-play module for existing pretrained models. A detailed complexity breakdown is provided in Appendix E.

4 Experiments

4.1 Experimental Setup

Models and Datasets. We evaluate LAB-Sparse on Llama-3.1-8B (Dubey et al., 2024), Qwen2.5-7B (Yang et al., 2024), and Mistral-7B (Jiang et al., 2023) using LongBench (Bai et al., 2024) and RULER (Hsieh et al., 2024). LongBench covers single-document QA, multi-document QA, summarization, few-shot learning, synthetic tasks, and code completion; following the official protocol, inputs exceeding 8K tokens are truncated from both ends to 8K. RULER evaluates retrieval and multi-hop reasoning over long sequences.

Baselines. We compare against Full (full attention without KV compression, as an upper bound), H2O (Zhang et al., 2023), SnapKV (Li et al., 2024b), StreamingLLM (Xiao et al., 2024), ChunkKV (Liu et al., 2025), and PyramidKV (Cai et al., 2025). All baselines are re-run under identical hardware, model weights, and prompt templates.

Implementation Details. LongBench experiments are conducted on a vGPU-48GB under the official protocol (inputs beyond 8K truncated from both ends, no chunked attention). Efficiency experiments are conducted separately on a vGPU-32GB using chunked attention to reduce peak memory during prefill on long sequences (4K–32K); the reference baseline is the chunked-attention full-cache implementation (`original`). All experiments use bfloat16 precision with greedy decoding (temperature=0) and a fixed KV budget of 512. Task descriptions and full efficiency measurements are provided in Appendix A and Appendix D. Our design choices are motivated by three diagnostic observations on Qwen2.5-7B (Appendix G): cross-layer consistency of low-importance tokens, partial correlation among attention heads, and spatial clustering of high-importance tokens.

4.2 Results

LongBench. Table 1 reports results on 16 LongBench tasks. Ours (LAB) achieves the highest average among sparsification methods on Llama-3.1-8B (36.76) and Qwen2.5-7B (34.19), and re-

Table 1: LongBench results with a fixed KV budget of 512, following the official protocol (inputs beyond 8K truncated from both ends, no chunked attention). Best and second-best results among sparsification methods are **bolded** and underlined.

Method	2WQ	DUR	GOV	HPQ	LCC	LSH	MNW	MFE	MFZ	NQA	QAS	QMS	RBP	SAM	TRE	VCS	Avg.
Llama-3.1-8B																	
Full	34.34	27.88	29.34	42.65	71.89	38.75	19.98	47.26	57.29	22.95	29.37	24.36	67.33	47.17	70.50	9.02	40.01
H2O	33.07	18.15	17.08	41.18	69.55	**37.75**	12.10	43.56	52.40	23.49	26.37	22.92	61.91	**45.54**	64.50	7.55	36.07
SnapKV	33.35	19.08	17.18	42.27	**70.56**	32.75	12.95	47.05	**56.14**	**24.27**	26.43	**23.64**	62.84	44.90	64.50	7.40	36.58
ChunkKV	**34.23**	19.23	**18.45**	42.21	67.92	**37.75**	**13.78**	45.32	53.40	20.90	26.80	22.76	62.19	44.91	**68.00**	**7.97**	36.61
StreamingLLM	26.17	14.99	15.66	33.94	67.85	20.50	12.82	28.00	31.27	13.75	17.39	20.98	59.74	44.65	54.50	4.10	29.14
PyramidKV	31.90	17.77	16.04	42.19	70.29	32.00	12.05	44.97	52.51	23.39	25.77	23.57	59.76	44.01	51.00	7.24	34.65
Ours (LAB)	33.04	**20.56**	18.42	**42.73**	69.65	33.50	12.03	**47.74**	54.90	20.85	**28.11**	23.16	**63.27**	44.62	**68.00**	7.64	**36.76**
Qwen2.5-7B																	
Full	35.93	40.29	32.69	12.09	57.96	33.22	23.55	39.17	56.20	7.02	40.29	17.70	63.21	43.20	66.50	15.16	36.51
H2O	34.15	31.34	23.17	**13.16**	54.41	29.61	19.93	34.16	48.54	6.64	31.34	16.88	55.69	**43.42**	61.00	13.06	32.28
SnapKV	33.40	33.94	24.14	10.06	53.74	28.12	20.50	35.03	52.01	4.55	33.94	16.10	59.76	42.75	56.50	12.61	32.32
ChunkKV	34.71	29.01	24.57	12.82	**55.65**	**33.67**	**20.67**	37.37	56.31	5.41	37.60	16.06	**60.19**	42.57	65.00	12.65	34.02
StreamingLLM	29.84	37.60	21.10	7.72	50.12	20.50	18.23	24.22	29.74	2.93	29.01	13.67	56.03	41.99	56.50	12.34	28.22
PyramidKV	36.06	34.11	22.37	12.69	55.33	23.89	19.74	**38.29**	51.95	**6.74**	34.11	**17.15**	59.76	42.26	54.50	**13.12**	32.63
Ours (LAB)	**36.14**	**38.83**	**24.97**	11.74	51.59	29.02	20.00	37.80	**56.45**	5.45	**38.83**	16.15	57.88	42.53	**67.00**	12.68	**34.19**
Mistral-7B																	
Full	40.76	33.90	34.26	43.23	60.65	31.75	27.91	55.23	57.05	24.91	40.32	24.35	59.61	47.29	74.50	16.05	41.99
H2O	38.97	21.43	24.48	44.44	58.69	31.00	24.34	53.50	49.62	24.39	33.94	21.71	56.42	45.52	72.50	13.42	38.40
SnapKV	34.75	21.91	**26.20**	42.49	57.24	24.25	24.33	48.49	36.00	22.57	35.28	21.19	55.57	46.03	**73.50**	**13.74**	36.47
ChunkKV	**39.46**	24.56	25.04	45.31	59.83	**31.75**	**24.46**	**54.42**	52.98	**25.17**	36.46	22.16	56.77	**46.24**	70.50	12.49	**39.23**
StreamingLLM	36.19	18.10	21.27	41.35	58.25	18.00	21.75	35.33	32.01	22.29	26.71	20.16	54.76	44.63	69.00	13.09	33.31
PyramidKV	39.23	22.89	24.44	44.95	**60.48**	21.50	24.23	53.10	48.21	24.31	33.57	**23.14**	**57.97**	44.94	59.00	12.93	37.18
Ours (LAB)	38.91	**26.87**	25.02	**45.35**	57.06	30.25	23.40	53.57	**54.18**	25.00	**36.47**	22.19	54.45	45.82	72.50	13.36	39.03

mains comparable to the strongest baseline on Mistral-7B (39.03 vs. 39.23 for ChunkKV). It obtains the best sparsification results on multiple tasks across all three models, including `dureader` and `qasper`, while performing comparably to or slightly below some baselines on `multi_news` and certain single-document QA tasks. Task abbreviations are defined in Appendix A.

RULER. Figure 3 reports accuracy from 4K to 32K on Qwen2.5-7B and Mistral-7B. Due to computational constraints, we focus on these two models, which represent different pretraining context length distributions; extending to Llama-3.1-8B is left for future work. In the 4K–16K range, Ours (LAB) remains close to Full and consistently exceeds the strongest sparse baseline. At 32K, all sparsification methods degrade substantially, yet Ours (LAB) still outperforms PyramidKV on both models. A detailed length-dependent analysis is provided in Appendix C, and per-task results are reported in Appendix B.

Inference Efficiency. Table 2 reports the decoding throughput speedup over the chunked-attention full-cache baseline (`original`). Ours (LAB) achieves $1.18\times$ at 4K, $1.51\times$ at 8K, $1.91\times$ at 16K, and $3.03\times$ at 32K, on par with ChunkKV ($3.21\times$) and PyramidKV ($3.00\times$) at long sequences. At 32K, it also reduces peak memory from 30.28 GB to 26.47 GB ($\sim$12.6%). Prefill latency remains nearly identical across methods (36.37 s for `original` vs. 36.54 s for Ours (LAB) at 32K, a 0.5% overhead), indicating that the overhead of LAB enhancement and interval delineation is negligible relative to the quadratic attention cost. Full efficiency measurements are provided in Appendix D.

Ablation. Table 3 reports ablation results on Llama-3.1-8B using HotpotQA (HPQ) and Qasper (QAS). Removing LAB causes the largest drop on Qasper (28.11 $\rightarrow$ 25.34), suggesting that neighborhood-based importance enhancement is critical for tasks requiring fine-grained context aggregation. Disabling distance decay reduces HotpotQA from 42.73 to 42.15, indicating that the distance-weighted neighborhood contributes to boundary detection. Removing tail retention yields 42.67 on HotpotQA but 26.78 on Qasper, suggesting that tail retention primarily benefits tasks relying on recently generated context. A limitation is that this ablation covers only two tasks; extending it to a broader set of task types is left for future work.

5 Conclusion

We introduced LAB-Sparse, a training-free KV cache sparsification method that retains dynamic-length contiguous blocks instead of isolated tokens. By smoothing token-level importance scores

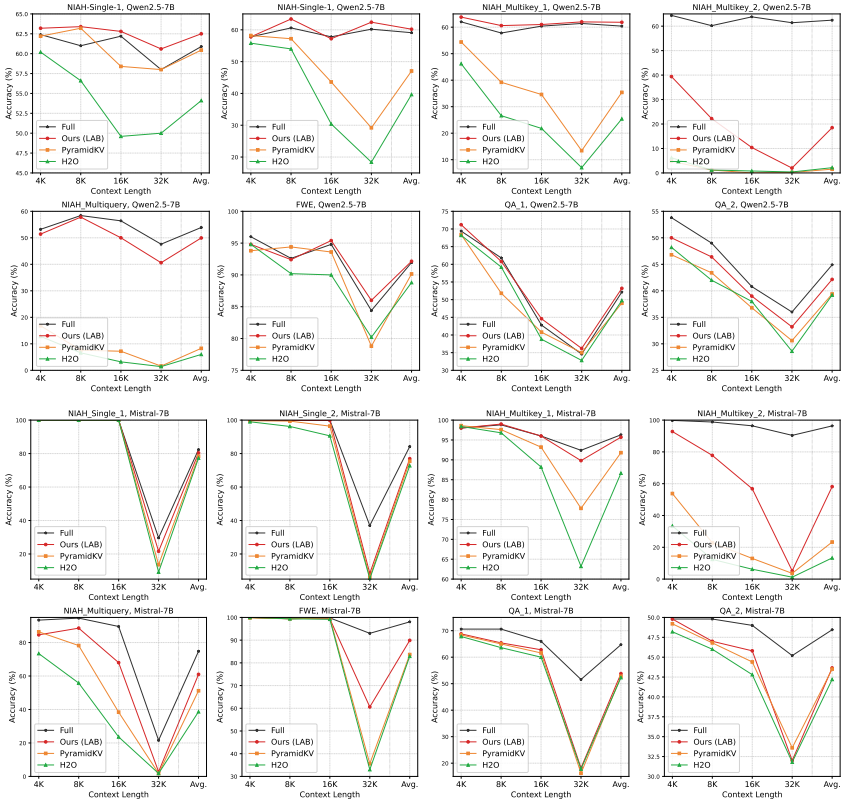


Figure 3: RULER results on Qwen2.5-7B (top) and Mistral-7B (bottom). Each panel reports the eight-task average accuracy versus sequence length. In the 4K–16K range, Ours (LAB) improves over the strongest sparse baseline (PyramidKV) by 21.3–33.5% on Qwen2.5-7B and 5.7–14.9% on Mistral-7B. At 32K, all sparse methods degrade substantially, but Ours (LAB) still exceeds PyramidKV by 55.0% and 26.3%, respectively.

Table 2: Decoding throughput speedup (×) over the chunked-attention full-cache baseline.

Method	**4K**	**8K**	**16K**	**32K**
H2O	0.99	1.28	1.52	2.47
SnapKV	1.16	1.49	1.93	2.98
ChunkKV	1.53	2.02	2.06	3.21
PyramidKV	1.17	1.51	1.91	3.00
Ours (LAB)	**1.18**	**1.51**	**1.91**	**3.03**

Table 3: Ablation on Llama-3.1-8B (F1). Dist. Decay = distance decay; Tail Ret. = tail retention.

Config.	**HPQ**	**QAS**
LAB-Sparse (full)	42.73	28.11
w/o LAB	42.49	25.34
w/o Dist. Decay	42.15	26.72
w/o Tail Ret.	42.67	26.78

with Local Accumulative Boost (LAB), it identifies contiguous high-importance regions via a quantile threshold and greedily preserves complete blocks under a fixed KV budget. This design improves local continuity while reducing memory and decoding cost.

Experiments on Llama-3.1-8B, Qwen2.5-7B, and Mistral-7B across LongBench and RULER show that LAB-Sparse matches or surpasses strong sparsification baselines and remains close to full attention. Under a 512-token KV budget, it improves decoding throughput by up to 3.03× and reduces peak memory by approximately 12.6% at 32K context length. Ablations confirm LAB as the key component, with distance decay and tail retention providing complementary gains.

The retention decision is fixed after prefill and shared across all decoding steps; adaptive refinement during decoding is a promising direction for tasks with evolving information needs. Our evaluation primarily covers question answering, summarization, code completion, and few-shot learning, while code generation, multi-turn dialogue, and sequences beyond 32K remain uncharacterized and are left for future work.

AI USE STATEMENT

In this work, we used generative AI tools for the following tasks:

- **Code modification and debugging.** We used AI tools to assist in modifying and debugging the implementation of LAB-Sparse, including attention computation, KV cache compression, and contiguous block selection. All AI-assisted code modifications were manually reviewed, tested, and verified by the authors. The authors take full responsibility for the correctness and final version of the code.
- **Literature survey.** We used AI tools to assist in retrieving and organizing relevant literature on KV cache compression, helping to identify research gaps and baseline methods. All cited references were personally read and verified by the authors. AI tools were not used to generate fabricated citations or unverified content.
- **Manuscript review and revision.** We used AI tools for language polishing, logical review, and expression refinement of the manuscript draft. All AI-assisted revisions were carefully reviewed and confirmed by the authors. The authors take full responsibility for the final content, claims, and conclusions of the paper.

We did not use generative AI tools to generate research ideas, formulate theoretical proofs, generate experimental data, or interpret experimental results. We adhere to the ICLR AI use policy and take responsibility for all content in this work.

ETHICS STATEMENT

This work adheres to the ICLR Code of Ethics. Our study focuses on improving the inference efficiency of large language models and does not involve human subjects, sensitive personal data, or animal experiments. The LongBench (Bai et al., 2024) and RULER (Hsieh et al., 2024) benchmarks used in our experiments are publicly available and do not contain private or sensitive information. The proposed method aims to reduce the memory and computational overhead of KV cache and does not involve techniques that could be misused for malicious purposes. We do not identify any ethical risks or potential harmful applications of this work.

REPRODUCIBILITY STATEMENT

We are committed to ensuring the reproducibility of this work. The methodological details of LAB-Sparse are fully described in the main text and appendix, including the LAB enhancement formula, quantile-based threshold selection, contiguous block extraction, and greedy block retention algorithm (see Algorithm 1). All experimental hyperparameters (e.g., `lab_window=3`, `lab_alpha=0.2`, `lab_threshold_ratio=0.7`, `max_capacity=512`, `window_size=64`), and hardware configurations are specified in the paper or supplementary material.

We use publicly available pretrained models (Llama-3.1-8B (Dubey et al., 2024), Qwen2.5-7B (Yang et al., 2024), Mistral-7B (Jiang et al., 2023)) and public benchmark datasets (LongBench (Bai et al., 2024), RULER (Hsieh et al., 2024)). All baselines are re-run under identical hardware, model weights, and prompt templates to ensure fair comparison.

Anonymous code will be submitted as supplementary material, including the complete model implementation, evaluation scripts, and running instructions. Due to the anonymous review requirement, the code link is not publicly available at this time and will be released upon acceptance.

REFERENCES

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchen Lyu, Jiankai Tang, Zhidian Huang, et al. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

Federico Barbero, Andrea Banino, Steven Kapturowski, D. Kumaran, João G. M. Araújo, et al. Transformers need glasses! Information over-squashing in language tasks. In *Advances in Neural Information Processing Systems 37*, 2024.

Priyansh Bhatnagar, Ashkan Moradifirouzabadi, Se-Hyun Yang, SeungJae Lee, Jungwook Choi, and Mingu Kang. STAR-KV: Low-rank KV cache compression via soft thresholding for adaptive rank control. *arXiv preprint arXiv:2606.08382*, 2026.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems 35*, 2022.

Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, et al. LongRoPE: Extending LLM context window beyond 2 million tokens. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, et al. Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S. Kevin Zhou. Ada-KV: Optimizing KV cache eviction by adaptive budget allocation for efficient LLM inference. In *Advances in Neural Information Processing Systems 38*, 2025.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive KV cache compression for LLMs. In *International Conference on Learning Representations*, 2024a.

Tao Ge, Jing Hu, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model. In *International Conference on Learning Representations*, 2024b.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, et al. KVQuant: Towards 10 million context length LLM inference with KV cache quantization. In *Advances in Neural Information Processing Systems 37*, 2024.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, et al. RULER: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, et al. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, et al. MInference 1.0: Accelerating pre-filling for long-context LLMs via dynamic sparse attention. In *Advances in Neural Information Processing Systems 37*, 2024.

Adrian Łańcucki, Konrad Staniszewski, Piotr Nawrot, and Edoardo M. Ponti. Inference-time hyper-scaling with KV cache compression. In *Advances in Neural Information Processing Systems 38*, 2025.

Haoyang Li, Yiming Li, Anxin Tian, Tianhao Tang, Zhanchao Xu, Xuejia Chen, et al. A survey on large language model acceleration based on KV cache management. *arXiv preprint arXiv:2412.19442*, 2024a.

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

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024a.

Xiang Liu, Zhenheng Tang, Peijie Dong, Zeyu Li, Yue Liu, Bo Li, Xuming Hu, and Xiaowen Chu. ChunkKV: Semantic-preserving KV cache compression for efficient long-context LLM inference. In *Advances in Neural Information Processing Systems 38*, 2025.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, et al. Scissorhands: Exploiting the persistence of importance hypothesis for LLM KV cache compression. In *Advances in Neural Information Processing Systems 36*, 2023.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, et al. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *Proceedings of the 41st International Conference on Machine Learning*, 2024b.

Jesse Mu, Xiang Lisa Li, and Noah D. Goodman. Learning to compress prompts with Gist tokens. In *Advances in Neural Information Processing Systems 36*, 2023.

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

Gergely Szilvasy, Manuel Faysse, Maria Lomeli, Matthijs Douze, Pierre-Emmanuel Mazaré, Loïc Cabannes, Wen-tau Yih, and Hervé Jégou. Self-pruned key-value attention: Learning when to write by predicting future utility. *arXiv preprint arXiv:2605.14037*, 2026.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context LLM inference. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

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

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bo-Wen Yu, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems 36*, 2023.

A Task Descriptions and Abbreviations

We evaluate on 16 tasks from LongBench (Bai et al., 2024). Table 4 lists the abbreviation, full name, and category of each task.

Discussion. The neighborhood radius $r = 3$ balances two competing considerations: a larger r over-smooths the importance distribution and blurs semantic boundaries, while a smaller r provides insufficient neighborhood context for detecting rising-edge signals. We adopt $r = 3$ as a stable default in all experiments. The scaling factor α controls the relative contribution of neighbor information to the final importance score; we set $\alpha = 0.2$ across all experiments, which provides a conservative enhancement that preserves the dominant role of each token's own attention score while still incorporating neighborhood context. The quantile ratio $q = 0.7$ provides a data-driven threshold that adapts to the score distribution of each sequence, avoiding fixed absolute thresholds that would vary across models and layers; we use the 70th percentile rather than the median ($q = 0.5$) because the latter retains too many marginal tokens under a fixed budget. The tail retention length $T = 64$ is small relative to the budget ($K = 512$) and serves only to preserve local continuity near the generation point. Cross-layer decision sharing uses a stride of 2; we did not observe a meaningful accuracy difference relative to per-layer decisions in our runs, while the decision overhead

Table 4: LongBench (Bai et al., 2024) task abbreviations, full names, and categories.

Abbr.	Full Name	Category
2WQ	2WikiMultihopQA	Multi-doc QA
DUR	DuReader	Multi-doc QA
GOV	GovReport	Summarization
HPQ	HotpotQA	Multi-doc QA
LCC	LCC	Code Completion
LSH	LSHT	Few-shot Learning
MNW	MultiNews	Summarization
MFE	MultiFieldQA-en	Single-doc QA
MFZ	MultiFieldQA-zh	Single-doc QA
NQA	NarrativeQA	Single-doc QA
QAS	Qasper	Single-doc QA
QMS	QMSum	Summarization
RBP	RepoBench-P	Code Completion
SAM	SAMSum	Few-shot Learning
TRE	TREC	Few-shot Learning
VCS	VCSUM	Summarization

is halved. To verify the robustness of these choices, we sweep each parameter individually while fixing the others at their defaults and measure PG19 (Rae et al., 2019) 16K perplexity on Mistral-7B (Jiang et al., 2023) (8 samples). Figure 4 and Table 5 report the results: PPL varies only between 9.38 and 9.61 across all sweeps (full-cache reference 8.32), remaining within $\sim$2% of the default configuration ($\alpha = 0.2$, $r = 3$, $q = 0.7$), which confirms that LAB-Sparse is insensitive to these hyperparameters.

Table 5: Hyperparameter sensitivity of LAB-Sparse on Mistral-7B (Jiang et al., 2023) (PG19 (Rae et al., 2019) 16K perplexity, 8 samples; full-cache reference 8.32). Defaults are **bolded**.

Parameter	Values	PPL
α (`lab_alpha`)	0.0	9.377
	0.1	9.432
	0.2	**9.414**
	0.4	9.581
r (`lab_window`)	1	9.451
	3	**9.414**
	5	9.553
q (`lab_threshold_ratio`)	0.5	9.396
	0.6	9.460
	0.7	**9.414**
	0.8	9.609

B Additional RULER Results

Due to computational constraints, we evaluate RULER (Hsieh et al., 2024) on a subset of baselines (original, LAB-Sparse, H2O (Zhang et al., 2023), and PyramidKV (Cai et al., 2025)) and two models (Qwen2.5-7B (Yang et al., 2024) and Mistral-7B (Jiang et al., 2023)) at four sequence lengths (4K, 8K, 16K, and 32K). Tables 6 and 7 report the results on eight RULER (Hsieh et al., 2024) tasks. The best result among the four evaluated methods is **bolded**, with ties bolded jointly. Missing entries indicate that the corresponding method was not evaluated under that setting; we leave a full evaluation across all models and baselines to future work.

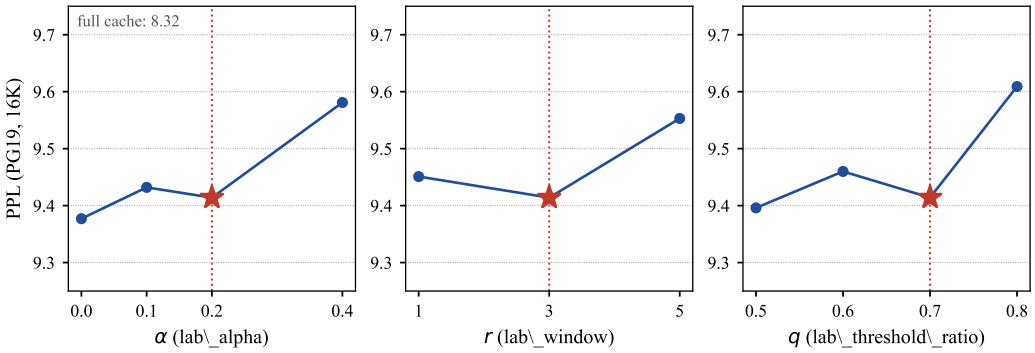


Figure 4: Hyperparameter sensitivity of LAB-Sparse on Mistral-7B (Jiang et al., 2023) (PG19 (Rae et al., 2019) 16K perplexity, 8 samples). Each panel sweeps one parameter while fixing the others at their defaults ($\alpha = 0.2$, $r = 3$, $q = 0.7$, marked with a star).

Table 6: RULER (Hsieh et al., 2024) results on Qwen2.5-7B (Yang et al., 2024) across four sequence lengths.

Task	**4K**				**8K**				**16K**				**32K**			
	orig	lab	h2o	pyr	orig	lab	h2o	pyr	orig	lab	h2o	pyr	orig	lab	h2o	pyr
niah_single_1	62.4	**63.2**	60.2	62.2	61.0	**63.4**	56.6	63.2	62.2	**62.8**	49.6	58.4	58.0	**60.6**	50.0	58.0
niah_single_2	57.8	57.8	55.8	**58.2**	60.6	**63.4**	54.0	57.2	**57.8**	57.2	30.4	43.6	60.2	**62.4**	18.4	29.2
niah_multikey_1	62.0	**63.8**	46.2	54.4	57.8	**60.6**	26.6	39.2	60.4	**61.0**	21.8	34.6	61.4	**62.0**	7.0	13.4
niah_multikey_2	**64.4**	39.4	6.2	4.8	**60.2**	22.2	1.2	1.2	**63.8**	10.4	0.8	0.0	**61.4**	2.0	0.4	0.2
niah_multiquery	**53.2**	51.4	12.8	16.8	**58.4**	57.8	6.6	7.6	**56.4**	50.0	3.2	7.2	**47.6**	40.6	1.4	1.6
fwe	**96.0**	94.8	94.8	93.8	92.6	92.4	90.2	**94.4**	94.8	**95.4**	90.0	93.6	84.4	**86.0**	80.2	78.8
qa_1	69.4	**71.2**	68.2	68.4	**61.8**	60.8	59.2	51.8	42.8	**44.6**	38.8	40.8	34.6	**36.2**	32.8	35.0
qa_2	**53.8**	50.0	48.2	46.8	**49.0**	46.4	42.0	43.4	**40.8**	39.0	38.0	36.8	**36.0**	33.2	28.6	30.6

C Length-Dependent Performance Analysis

Table 8 summarizes the eight-task average RULER (Hsieh et al., 2024) accuracy of Qwen2.5-7B (Yang et al., 2024) and Mistral-7B (Jiang et al., 2023) at four sequence lengths. Both models exhibit a consistent degradation trend: as the sequence length grows from 4K to 32K, the average accuracy of all methods declines to varying degrees.

On Qwen2.5-7B (Yang et al., 2024), the full-cache baseline (`orig`) drops from 64.9 at 4K to 55.5 at 32K (a 9.4-point decline). LAB-Sparse drops from 61.5 to 47.9, a 13.6-point decline, positioned between the full-cache baseline and H2O (Zhang et al., 2023) (-21.7) and PyramidKV (Cai et al., 2025) (-19.8). On multi-key and multi-query tasks such as `niah_multikey_2` and `niah_multiquery`, H2O (Zhang et al., 2023) and PyramidKV (Cai et al., 2025) deteriorate to single-digit accuracy beyond 16K, whereas LAB-Sparse remains at 40–63.

On Mistral-7B (Jiang et al., 2023), all methods remain at a high level from 4K to 16K (`orig` from 88.9 to 87.1; LAB from 86.7 to 78.6), but a sharp decline occurs at 32K. Here `orig` falls to 57.6 (-31.3 points), while the three sparsification methods decline by a similar margin of 56–58 points: LAB-Sparse to 29.8, H2O (Zhang et al., 2023) to 20.4, and PyramidKV (Cai et al., 2025) to 23.6. Tasks such as `niah_single_1` and `niah_single_2` plunge from near 100 to 20–37, and `niah_multiquery` falls from 89.6 to 21.6.

We attribute this degradation to three compounding factors. **First**, the KV budget is fixed at 512 while the sequence length grows eightfold, reducing the average budget available per token and diluting critical information. **Second**, the multi-key and multi-query tasks in RULER (Hsieh et al., 2024) require retrieving multiple independent facts from a long sequence; under higher compression, these scattered pieces of evidence are more likely to be truncated or discarded. **Third**, the

Table 7: RULER (Hsieh et al., 2024) results on Mistral-7B (Jiang et al., 2023) across four sequence lengths.

Task	4K				8K				16K				32K			
	orig	lab	h2o	pyr	orig	lab	h2o	pyr	orig	lab	h2o	pyr	orig	lab	h2o	pyr
niah_single_1	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**29.6**	21.6	9.2	13.6
niah_single_2	**99.8**	**99.8**	99.0	99.6	**100.0**	99.8	96.2	99.4	**100.0**	99.8	90.6	96.4	**37.0**	8.4	5.2	6.2
niah_multikey_1	98.0	98.0	98.4	**98.6**	98.8	**99.0**	96.8	97.6	**96.0**	**96.0**	88.2	93.2	**92.4**	89.8	63.2	77.8
niah_multikey_2	**99.8**	92.8	33.4	53.8	**98.8**	77.8	12.4	22.8	**96.4**	56.8	6.2	13.0	**90.4**	5.2	1.2	3.6
niah_multiquery	**93.4**	84.6	73.4	86.4	**94.6**	88.6	55.8	78.2	**89.6**	68.0	23.6	38.4	**21.6**	2.8	1.8	1.8
fwe	**100.0**	**100.0**	**100.0**	99.8	99.4	**99.6**	99.4	99.4	**99.8**	99.4	99.2	**99.8**	**93.0**	60.6	33.0	35.6
qa_1	**70.6**	68.8	67.8	68.4	**70.6**	65.4	63.6	65.0	**66.0**	62.8	60.0	61.6	**51.6**	18.2	17.8	16.2
qa_2	**49.8**	**49.8**	48.2	49.2	**49.8**	47.0	46.0	46.8	**49.0**	45.8	42.8	44.4	**45.2**	32.0	31.8	33.6

Table 8: Eight-task average RULER (Hsieh et al., 2024) accuracy at four sequence lengths. Δ denotes the drop from 4K to 32K.

Model	Method	4K	8K	16K	32K	Δ
Qwen2.5-7B	orig	64.9	62.7	59.9	55.5	-9.4
	lab	61.5	58.4	52.6	47.9	-13.6
	h2o	49.1	42.1	34.1	27.4	-21.7
	pyr	50.7	44.8	39.4	30.9	-19.8
Mistral-7B	orig	88.9	89.0	87.1	57.6	-31.3
	lab	86.7	84.7	78.6	29.8	-56.9
	h2o	77.5	71.3	63.8	20.4	-57.1
	pyr	82.0	76.2	68.4	23.6	-58.4

degradation on Mistral-7B (Jiang et al., 2023) at 32K is substantially larger than on Qwen2.5-7B (Yang et al., 2024), likely related to their pretraining context length distributions.

Notably, in the 4K–16K range, LAB-Sparse degrades less than H2O (Zhang et al., 2023) and PyramidKV (Cai et al., 2025), indicating that it preserves information integrity well at moderate lengths. At 32K under extreme compression, all sparsification methods degrade substantially, reflecting a common bottleneck of KV sparsification on very long sequences rather than a deficiency specific to any single method.

D Efficiency Details

Table 9 reports the full efficiency measurements across four sequence lengths (4K, 8K, 16K, 32K) for all methods. The reference baseline is the chunked-attention full-cache implementation (`original`). The original full-attention implementation (`base`) is only available at 4K and 8K, as it exceeds the 32GB memory limit at longer sequences. This directly motivates the need for KV compression in long-context inference.

Observations. (1) Prefill time is nearly constant across methods at each sequence length: 0.89–1.09 s at 4K, 2.90–3.38 s at 8K, 9.84–10.04 s at 16K, and 36.32–36.94 s at 32K; Ours (LAB) adds only 0.17 s (0.5%) over the `original` baseline at 32K, indicating that the overhead of LAB enhancement and interval delineation is negligible relative to the quadratic attention cost. (2) At 32K, Ours(LAB) achieves 57.62 tps decoding throughput, corresponding to $3.03\times$ speedup over the chunked-attention baseline (19.04 tps), and reduces peak memory from 30.28 GB to 26.47 GB ($\sim$12.6%). (3) ChunkKV (Liu et al., 2025) and StreamingLLM (Xiao et al., 2024) report lower peak memory at 4K and 8K; this is attributable to their implementation directly operating on chunked tensors without materializing the full KV tensor, rather than to the compression method itself. (4) The kept ratio of Ours(LAB) at 4K exhibits a range of $[0.129, 0.152]$ across layers, reflecting the adaptive nature of interval delineation; at longer sequences the ratio becomes uniform as the budget becomes the binding constraint.

Table 9: Full efficiency measurements. Decode: decoding throughput (tokens/s). E2E: end-to-end latency (s). Peak: peak memory (GB).

Method	Length	Decode (tps)	E2E (s)	Peak (GB)	Kept Ratio
base	4K	42.46	4.08	33.23	1.00
	8K	31.87	7.38	36.01	1.00
original	4K	42.35	4.07	28.89	1.00
	8K	31.89	7.33	30.83	1.00
	16K	29.74	14.14	21.40	1.00
	32K	19.04	43.10	30.28	1.00
H2O	4K	41.71	4.13	28.43	0.123
	8K	40.79	6.47	29.91	0.063
	16K	45.22	12.68	19.55	0.032
	32K	47.04	39.13	26.59	0.016
SnapKV	4K	49.21	3.67	28.47	0.150
	8K	47.51	6.03	29.92	0.076
	16K	57.33	12.09	19.52	0.038
	32K	56.82	38.61	26.46	0.019
ChunkKV	4K	64.92	2.89	14.96	0.150
	8K	64.55	4.93	16.42	0.076
	16K	61.32	12.13	19.51	0.038
	32K	61.06	39.03	26.46	0.019
StreamingLLM	4K	44.02	3.80	14.95	0.121
	8K	43.71	5.83	16.39	0.062
	16K	42.12	12.88	19.45	0.031
	32K	39.25	39.58	26.34	0.016
PyramidKV	4K	49.59	3.64	28.47	0.151
	8K	48.16	5.98	29.95	0.077
	16K	56.73	12.10	19.52	0.039
	32K	57.19	38.58	26.47	0.019
Ours(LAB)	4K	50.00	3.65	28.47	0.150
	8K	48.11	6.04	29.92	0.077
	16K	56.76	12.16	19.52	0.039
	32K	57.62	38.77	26.47	0.020

E Complexity Analysis

Let L denote the sequence length, K the KV budget, r the neighborhood radius, m the number of candidate blocks, and d the hidden dimension.

Time complexity.

- Raw importance score computation: $O(L \cdot d)$ for the last-query attention over all keys.
- LAB enhancement via 1D convolution: $O(L \cdot r)$, with $r = 3$ constant.
- Quantile threshold: $O(L \log L)$ for sorting, or $O(L)$ with selection algorithms.
- Block extraction: $O(L)$ single linear scan.
- Block sorting: $O(m \log m)$ with $m \ll L$.
- Greedy selection with sub-segment search: $O(L)$ worst case.

The total overhead is $O(L \log L + L \cdot r)$, negligible compared with the $O(L^2 \cdot d)$ cost of full attention. Since the decision is executed once at prefill, the amortized cost over T decoding steps is $O((L \log L + L \cdot r)/T)$.

Memory complexity. The additional memory required is $O(L)$ for the importance score vector and $O(L)$ for the boolean mask, both released immediately after the decision. The retained KV cache occupies $O(K \cdot d)$ memory, compared with $O(L \cdot d)$ for full attention.

Training-free property. LAB-Sparse introduces no trainable parameters. All operations are deterministic given the input sequence and hyperparameters, making the method immediately applicable to any pretrained model without fine-tuning.

F Reproducibility Details

Hardware. LongBench (Bai et al., 2024) experiments are conducted on a vGPU-48GB. Efficiency experiments are conducted on a vGPU-32GB using chunked attention to reduce peak memory during prefill. All experiments use bfloat16 precision.

Software. We use PyTorch 2.x and HuggingFace Transformers. All baselines are re-implemented or adapted from official repositories and run under identical conditions.

Decoding. All experiments use greedy decoding (temperature=0) with a maximum generation length determined by the official LongBench (Bai et al., 2024) protocol. Greedy decoding eliminates sampling randomness; the only source of nondeterminism is GPU kernel scheduling, which we did not observe to affect reported metrics.

Data preprocessing. Following the official LongBench (Bai et al., 2024) protocol, inputs exceeding 8K tokens are truncated from both ends to 8K. No additional preprocessing is applied.

Baselines. H2O (Zhang et al., 2023), SnapKV (Li et al., 2024b), StreamingLLM (Xiao et al., 2024), ChunkKV (Liu et al., 2025), and PyramidKV (Cai et al., 2025) are run with their official recommended configurations. The KV budget is set to 512 for all methods to ensure fair comparison.

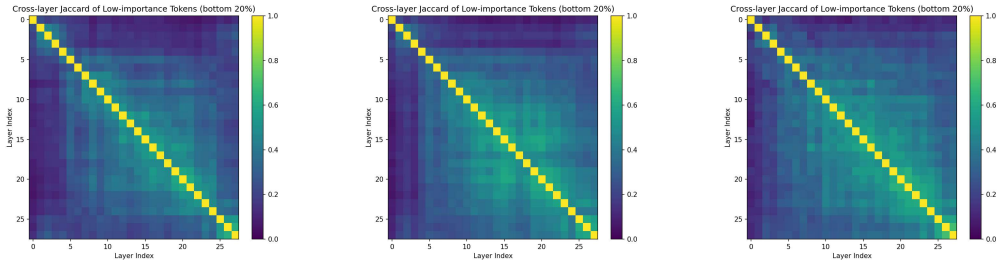
Anonymous code. An anonymized implementation of LAB-Sparse, including the attention replacement module and the block selection algorithm, will be submitted as supplementary material.

G Attention Structure Diagnostics

To validate the core assumptions underlying the design choices of LAB-Sparse, we conduct a set of diagnostic analyses on Qwen2.5-7B (Yang et al., 2024), examining three structural properties of attention scores: cross-layer consistency, head-level similarity, and spatial clustering of high-importance tokens. We note that these analyses are performed on a single model and are based on attention patterns from individual forward passes; their purpose is to *qualitatively illustrate design motivations* rather than to provide comprehensive cross-model statistical evidence.

G.1 Cross-Layer Consistency of Low-Importance Tokens

Figure 5 shows the Jaccard similarity of low-importance tokens (bottom 20%) across layers for three representative cases. The diagonal is 1 (self-similarity), and adjacent layers exhibit a similarity of approximately 0.4–0.5, which decays gradually as the layer distance increases. This indicates that different layers have moderately consistent judgments about "which tokens are unimportant," especially between adjacent layers. This supports our **cross-layer decision sharing** scheme (stride 2): sharing retention indices across adjacent layers does not introduce significant semantic deviation while halving the decision overhead. At the same time, the similarity is not close to 1, indicating that layers still differ, and fully sharing decisions across all layers would be inappropriate.

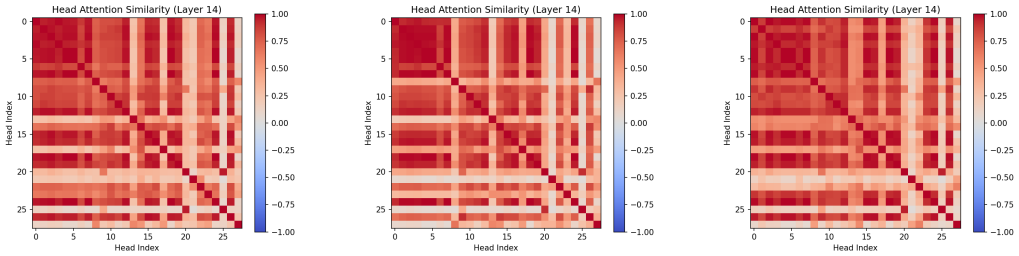


(a) Case 1 (b) Case 2 (c) Case 3

Figure 5: Cross-layer Jaccard similarity of low-importance tokens (bottom 20%) on Qwen2.5-7B (Yang et al., 2024) across three representative cases.

G.2 Head-Level Attention Similarity

Figure 6 shows the similarity of attention patterns across attention heads at Layer 14 for three representative cases. Some heads are highly correlated (dark red blocks), while others are weakly correlated or nearly independent. This indicates **functional differentiation** among heads, but not complete independence. Our choice to average importance scores across heads is based on this observation: since heads are partially correlated, averaging suppresses per-head noise while preserving the consensus signal. Other layers may exhibit different head-level structures, which warrants systematic analysis in future work.



(a) Case 1 (b) Case 2 (c) Case 3

Figure 6: Head attention similarity at Layer 14 on Qwen2.5-7B (Yang et al., 2024) across three representative cases.

G.3 Spatial Clustering of High-Importance Tokens

Figure 7 presents box plots of the distances between consecutive high-importance tokens (top 50%) for two representative samples. In both cases, the median gap is **1**, and the mean is close to **2**, indicating that high-importance tokens typically appear *consecutively* rather than being uniformly scattered across the sequence. This provides the most direct evidence for our core design—**continuous block retention**: if high-importance tokens naturally cluster into contiguous spans, retaining them as blocks better preserves semantic integrity than retaining isolated tokens. Only a small fraction of gaps exceed 5, suggesting the existence of a few isolated important tokens, which motivates our sub-segment fallback mechanism in the greedy selection.

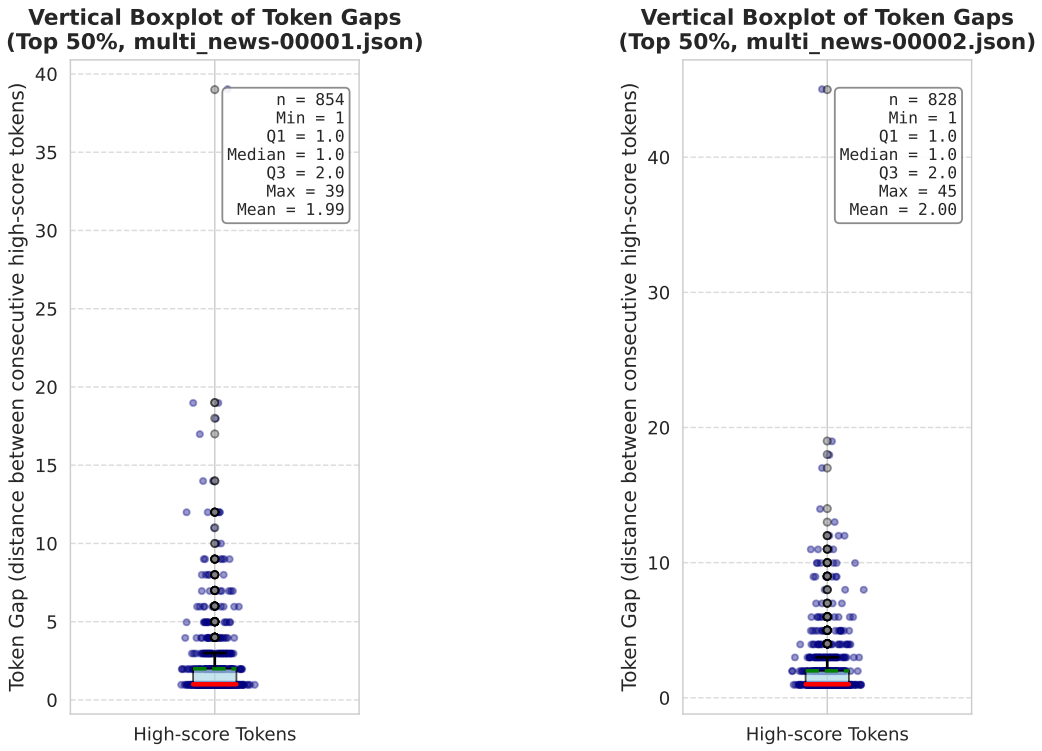
G.4 Summary

The three diagnostic analyses jointly support three key design choices of LAB-Sparse: cross-layer decision sharing (F.1), head-wise averaging (F.2), and continuous block retention (F.3). We emphasize that these analyses are performed on a single model and serve to illustrate design motivations rather than establish general conclusions. Systematic validation of these structural properties across multiple models is left for future work.

H Limitations and Broader Impact

Limitations.

- The retention decision is fixed after prefill. Adaptive refinement during decoding could further improve performance, especially for tasks with evolving information needs.
- Multi-key and multi-query retrieval under extreme compression (e.g., RULER at 32K) remains challenging for all sparsification methods, including ours. Preserving many independent short spans within a small KV budget is not well aligned with the continuity-focused design of LAB-Sparse.
- Our evaluation focuses on question answering and summarization. Code generation, multi-turn dialogue, and sequences beyond 32K remain uncharacterized.



(a) Sample 1

(b) Sample 2

Figure 7: Distribution of gaps between consecutive high-importance tokens (top 50%) on two representative samples from Qwen2.5-7B (Yang et al., 2024).

- Ablation studies cover two tasks on one model; a broader study across all models and tasks would strengthen the empirical evidence.
- The attention structure diagnostics (Appendix G) are performed on a single model (Qwen2.5-7B (Yang et al., 2024)) and individual forward passes; they illustrate design motivations rather than establish general conclusions.
- A direct comparison with learning-driven methods (SP-KV (Szilvasy et al., 2026), DMS (Łańcucki et al., 2025), Ada-KV (Feng et al., 2025)) under matched compression ratios is not included, as these methods require additional training that is orthogonal to our training-free setting.

Broader impact. LAB-Sparse reduces the memory and latency cost of long-context LLM inference, contributing positively to the sustainability of AI deployment. The method introduces no trainable parameters and no additional data collection, and therefore raises no new privacy or fairness concerns beyond those inherent to the underlying pretrained models. We do not foresee any negative societal impacts specific to this work.