LIFETIME: KV CACHE EVICTION FROM ATTENTION TRAJECTORIES FOR LONG-CONTEXT LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-context language models store a key–value (KV) cache whose size grows linearly with the sequence length and quickly dominates inference memory and bandwidth. Training-free prefill eviction reduces this cost, but observation-window scorers typically collapse prompt-tail attention into a single static statistic, discarding its temporal order and conflating tokens whose attention is fading with tokens whose attention is recurring or rising. We introduce Lifetime, a query-aware, prefill-only method that keeps a temporally binned, ordered attention trajectory for every KV head. Using only evidence available once prefill is complete, Lifetime estimates a proxy for the proximity of each token's next important use, weights its peak attention rank by this proxy, stabilizes the result with mean-attention rank, and adds a lightweight value-magnitude cue. Every layer–head pair then retains the same number of independently selected entries, which are physically gathered into a regular shortened cache and decoded without rescoring or further eviction. On 16 LongBench tasks, Lifetime attains the highest average among compressed methods at both 10% and 5% retention on Qwen3-4B (47.90 and 45.13) and Llama-3.1-8B-Instruct (48.60 and 46.45). On RULER-16K with 256 retained entries per layer–KV-head pair, it exceeds the strongest alternative by 9.97 and 5.16 points on the two models, while adding less than 1% prefill latency over SnapKV. These results show that prompt-tail temporal structure is a practical, training-free signal for aggressive one-shot KV-cache compression.

1 INTRODUCTION

Long-context language models have broadened the scope of document understanding, retrieval-augmented generation, code analysis, and multi-step reasoning, but this capability carries a substantial inference-time memory cost (Vaswani et al., 2017; Dai et al., 2019; Rae et al., 2020; Beltagy et al., 2020; Zaheer et al., 2020). During autoregressive inference, every attention layer stores a key and a value vector for each prompt token, so both the KV-cache footprint and its memory traffic grow linearly with the context length (Dai et al., 2019; Rae et al., 2020). Multi-query and grouped-query attention reduce the cached heads (Shazeer, 2019; Ainslie et al., 2023), yet they leave this sequence-length growth untouched: for sufficiently long inputs, the cache becomes the dominant memory consumer and makes decoding bandwidth-bound even when the model weights fit comfortably in device memory (Dao et al., 2022; Kwon et al., 2023). Training-free KV-cache eviction offers a practical remedy for frozen models, requiring no task-specific optimization, calibration data, or architectural change (Xiao et al., 2024; Zhang et al., 2023; Li et al., 2024). It is especially attractive in the query-aware setting, where the user request is available at the end of prefill and can guide which context remains accessible during generation (Li et al., 2024; Tang et al., 2024).

Existing eviction methods approach this decision from several complementary directions. Structural policies preserve attention sinks and recent tokens (Xiao et al., 2024; Zhang et al., 2023); observation-window methods rank older entries by the attention they receive near the end of the prompt (Li et al., 2024); adaptive methods redistribute a fixed budget across heads or layers (Feng et al., 2024; Cai et al., 2024); and value-, output-, prediction-, or capacity-aware methods refine importance with richer notions of consequence or future relevance (Feng et al., 2025; Devoto et al., 2025; Yang et al., 2026). Despite this diversity, many scoring-based prefill methods eventually

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.

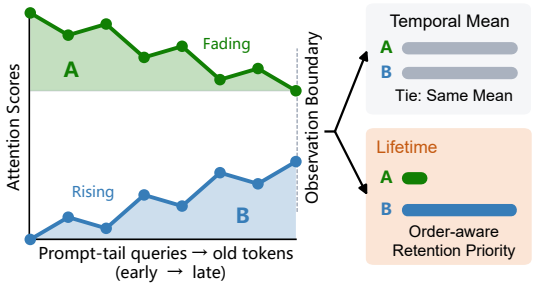


Figure 1: Temporal-order ambiguity. Token A (fading) and token B (rising) receive equal mean attention from prompt-tail queries, so an order-invariant temporal mean ties them. Lifetime reads the ordered trajectory and gives B, whose attention still rises at the observation boundary, higher priority than A.

reduce each KV entry to a single static statistic before pruning, which conceals the temporal organization of attention. Two tokens with the same average attention can follow fundamentally different trajectories: one may be steadily fading, another may receive sparse but recurring activations, and a third may have only recently begun to rise. The distinction is especially consequential under one-shot prefill pruning, because an entry removed at this point cannot be recovered if it becomes relevant later in generation. A static summary therefore conflates accumulated past attention with plausible near-term reuse, as Figure 1 illustrates for a fading and a rising token with equal mean.

Classical cache replacement offers a useful lens on this problem. Belady's MIN policy evicts the entry whose next use lies farthest in the future (Belady, 1966), and practical hardware policies approximate it by predicting reuse distances from observed access histories (Shah et al., 2023). Motivated by this view, we introduce *Lifetime*, a training-free, query-aware, prefill-only eviction method that constructs a temporally binned attention trajectory independently for each KV head and derives from it a proxy for the proximity of a token's next important use. Recurring activations yield a phase-based estimate, whereas a rising trend yields a slope-based estimate. Lifetime weights each token's peak observed attention rank by this proximity proxy, stabilizes the result with mean-attention rank, and adds a lightweight value-magnitude cue instead of reducing the observation window to a single average. Unlike Belady's oracle, it uses only the information available at the end of prefill and never observes future decoding states. Given a fixed budget, each KV head selects its retained positions independently, after which the selected keys and values are physically gathered into a shorter cache and decoding proceeds without further eviction. Our contributions are summarized as follows:

- We identify temporal-order ambiguity as a blind spot of observation-window eviction and propose Lifetime, which estimates a next-use proximity proxy from the ordered trajectory of every KV head instead of a static average over the observation window.
- We fuse this proxy with peak rank, mean-attention rank, and a value-rank cue, and gather the top entries of every head into a regular shortened cache that needs no further eviction.
- On Qwen3-4B and Llama-3.1-8B-Instruct, Lifetime attains the best compressed average on LongBench at both budgets and leads under the tightest RULER budget.

2 Related Work

Online and streaming eviction. Early KV-cache eviction focused on streaming decoding, where cache contents are revised as generation proceeds. StreamingLLM preserves attention sinks and recent entries, while H_2O augments recency with heavy hitters identified by cumulative attention (Xiao et al., 2024; Zhang et al., 2023). Scissorhands exploits the persistence of token importance, and TOVA updates the cache from the attention of the current decoding step (Liu et al., 2023; Oren et al., 2024). These policies address evolving relevance, but they either require decode-time state changes or collapse prior observations into scalar histories maintained online.

Attention-based prefill eviction. Prefill eviction instead selects a compact cache once, after the complete prompt is available. SnapKV, the closest baseline to Lifetime, scores earlier tokens from a prompt-tail observation window, aggregates the evidence across queries, and applies local pooling before per-head selection (Li et al., 2024). This aggregation captures recent relevance but removes the order in which attention changes within the window. DefensiveKV replaces mean aggregation with a worst-case-oriented defensive aggregation, and its Layer-DefensiveKV variant adds layer-

wise budget allocation (Feng et al., 2026). AdaKV redistributes a global budget across heads, PyramidKV assigns nonuniform capacity across layers (Feng et al., 2024; Cai et al., 2024), and chunk-level and query-aware sparsity methods provide complementary mechanisms (Liu et al., 2025; Tang et al., 2024). Because allocation and scoring are orthogonal choices, Lifetime keeps equal per-head budgets so that its contribution isolates temporal scoring rather than adaptive allocation.

Consequence-aware and predictive eviction. Recent work enriches eviction criteria beyond observed attention magnitude. CriticalKV analyzes attention-output perturbation and accounts for value states and projection matrices when estimating deletion consequences (Feng et al., 2025), while ReST-KV minimizes a layer-wise output-reconstruction error and smooths the resulting scores over space and time (An et al., 2026). Expected Attention scores entries under a modeled distribution of future queries, and CapKV maximizes an information-capacity objective through a log-determinant leverage approximation (Devoto et al., 2025; Yang et al., 2026). KVzip selects entries through context reconstruction (Kim et al., 2025), and orthogonal directions use quantization, channel pruning, layer similarity, or a split between retrieval and streaming heads (Liu et al., 2024; Xu et al., 2025; Zhang et al., 2024; Xiao et al., 2025). Unlike methods that model unknown future queries, Lifetime derives its proximity proxy directly from the observed ordering of prompt-tail attention, without access to future tokens or decode-time attention. It is thus closer to reuse-distance prediction for hardware caches (Belady, 1966; Shah et al., 2023), applied per KV head.

3 Method

3.1 Problem Setting and Overview

Let a prompt of length N contain both the long context and the user query available before generation. After full prefill, an autoregressive Transformer with L layers, H_q query heads, and H_{kv} KV heads stores key and value tensors $K_{l,h}, V_{l,h} \in \mathbb{R}^{N \times d_h}$ for every layer–head pair (l, h). Lifetime operates once at this boundary and assigns a retention score to every prompt position in each KV head; it is query-aware, prefill-only, and training-free, requiring neither calibration data nor updates to model parameters. Every KV head in every layer receives the same token budget B. Under a relative retention ratio $\rho \in (0, 1]$ we set $B = \lfloor \rho N \rfloor$, whereas under an absolute target (as in our RULER experiments) B is given directly. The decision is therefore one-shot and irreversible, since an entry removed after prefill can no longer be recovered at any later decoding step.

A common query-aware strategy averages the attention received from a prompt-tail observation window, reducing an ordered sequence to a single scalar that cannot distinguish a fading trajectory such as $(0.4, 0.2, 0.0)$ from a rising one such as $(0.0, 0.2, 0.4)$. Lifetime instead preserves the ordered trajectory of each KV head. As summarized in Figure 2, it converts prompt-tail attention into a binned trajectory (Section 3.2), estimates a next-use proximity proxy from that trajectory (Section 3.3), fuses the proxy with rank and value cues into a protected retention score (Section 3.4), and finally gathers the top-B entries of every head into a shortened cache (Section 3.5).

3.2 Ordered Attention Trajectories

We describe the scorer for one layer and omit the layer index for clarity. Let the prompt contain $N > W$ positions, with the last W queries forming the observation window and the first $M = N - W$ positions forming the old prefix to be scored. Let $T \geq 2$ divide W, so that the window is partitioned into T contiguous bins $\{\mathcal{Q}_t\}_{t=0}^{T-1}$ of equal size. For query head g, observation position q, and old-prefix position i, let $a_{g,q,i}$ denote the causal softmax attention weight. Writing $h(g)$ for the KV head shared by query head g, this weight is

$$a_{g,q,i} = \frac{\exp\big(\langle q_{g,q}, k_{h(g),i} \rangle / \sqrt{d_h}\big)}{\sum_{j \leq q} \exp\big(\langle q_{g,q}, k_{h(g),j} \rangle / \sqrt{d_h}\big)}. \tag{1}$$

Under grouped-query attention (Ainslie et al., 2023), each KV head h is shared by a group $\mathcal{G}(h)$ of query heads. Lifetime averages over the query heads of this group and over the successive queries within each bin:

$$A_{h,t,i} = \frac{1}{|\mathcal{G}(h)|\,|\mathcal{Q}_t|} \sum_{g \in \mathcal{G}(h)} \sum_{q \in \mathcal{Q}_t} a_{g,q,i}. \tag{2}$$

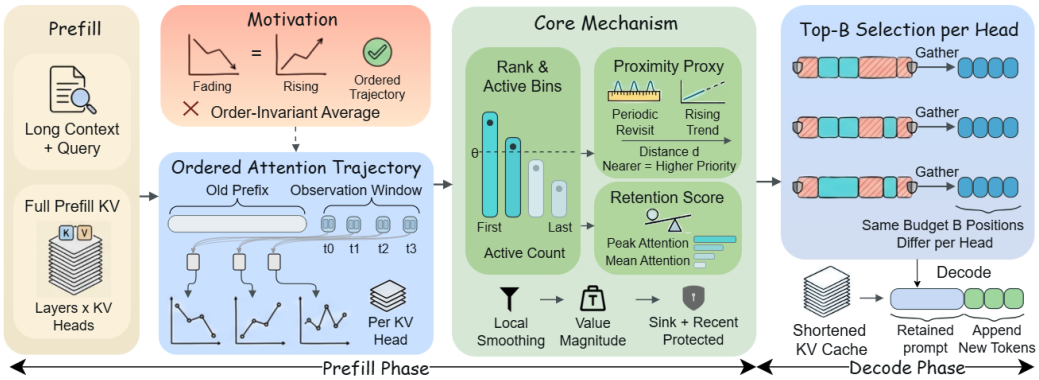


Figure 2: Overview of Lifetime. After full prefill, ordered prompt-tail attention trajectories are converted into a proximity proxy and blended with mean-attention rank and a value-rank cue to produce retention scores. Each KV head then independently selects the same fixed top-B budget, gathers the retained K/V entries into a shorter cache, and decodes without further eviction.

This aggregation suppresses position-level noise while keeping a coarse temporal order in $A_{h,:,i}$.

Lifetime next converts each bin into head-local ordinal evidence. Raw attention scales vary across heads and layers, so rank normalization provides a stable basis for comparing tokens within each bin. For any length-M vector x, let $\mathcal{R}(x)_i$ be the ordinal rank of x_i, increasing from 0 for the smallest entry to $M-1$ for the largest, divided by $\max(M-1,1)$. The normalized trajectory is $R_{h,t,i} = \mathcal{R}(A_{h,t,:})_i$. Given an active-rank threshold $\theta \in [0,1]$ and the indicator function $\mathbb{I}$, we mark an active event by $Z_{h,t,i} = \mathbb{I}[R_{h,t,i} \geq \theta]$. Let $c_{h,i} = \sum_t Z_{h,t,i}$ be the active count, and, when $c_{h,i} \geq 2$, let $f_{h,i}$ and $\ell_{h,i}$ denote the first and last active bins of token i in head h.

3.3 Next-Use Proximity Proxy

Repeated active bins support a spacing-based estimate of when a token will next be important, whereas trajectories with fewer than two active bins require a trend-based fallback; these two cases define the periodic and the rising branch. For a token active in at least two bins, the average observed spacing is $p_{h,i} = (\ell_{h,i} - f_{h,i})/(c_{h,i} - 1)$. The number of bins elapsed since its last active event, $\delta_{h,i}$, and the resulting periodic distance proxy, $d^{\text{per}}_{h,i}$, are then given by

$$\delta_{h,i} = T - 1 - \ell_{h,i}, \qquad d^{\text{per}}_{h,i} = (p_{h,i} - \delta_{h,i}) \bmod \max(p_{h,i}, 1). \tag{3}$$

When fewer than two active bins are available, the scorer instead examines the raw change between the last two bins. Let $v_{h,i} = A_{h,T-1,i} - A_{h,T-2,i}$, let $A^{\max}_{h,i} = \max_t A_{h,t,i}$, and let $\varepsilon > 0$ be a small numerical constant. The rising proxy extrapolates how many bins the current slope needs to recover the observed peak,

$$d^{\text{rise}}_{h,i} = \begin{cases} \dfrac{A^{\max}_{h,i} - A_{h,T-1,i}}{\max(v_{h,i}, \varepsilon)}, & v_{h,i} > 0, \\ T, & v_{h,i} \leq 0. \end{cases} \tag{4}$$

Lifetime selects the applicable branch and bounds its output by the observed temporal range:

$$d_{h,i} = \text{clip}_{[0,T]}\left(\begin{cases} d^{\text{per}}_{h,i}, & c_{h,i} \geq 2, \\ d^{\text{rise}}_{h,i}, & c_{h,i} < 2. \end{cases} \right) \tag{5}$$

A small $d_{h,i}$ therefore marks a token whose next important use appears imminent, and a large $d_{h,i}$ marks a token whose attention is fading or absent at the observation boundary.

3.4 Retention Score

Phase and slope estimates from a short observation window can be noisy, so Lifetime uses mean-attention rank as a stabilizer rather than relying on the proximity branch alone. Let $P_{h,i} =$

Algorithm 1 Lifetime KV-cache compression, executed once between prefill and decoding.

Require: Prompt length N; budget B; hyperparameters $W, T, \theta, m, \beta, k, S, R$
Ensure: Shortened per-head caches $\{K'_{l,h}, V'_{l,h}\}$
1: **for** each layer l and each KV head h **do**
2: Attend from the last W queries to the $M = N - W$ old positions; bin into A, rank into R
3: Mark active bins at threshold θ; estimate d via the periodic or rising branch
4: $u \leftarrow m\,\mathcal{R}(\overline{A}) + (1-m)\,P/(1+d)$, $s \leftarrow \mathrm{AvgPool}_k(u) + \beta Q$; protect $\mathcal{P}$
5: $\mathcal{I}_{l,h} \leftarrow \mathrm{TopB}(s; B)$; gather $K'_{l,h}, V'_{l,h}$ with $\mathcal{I}_{l,h}$
6: **end for**
7: **return** $\{K'_{l,h}, V'_{l,h}\}$ for append-only decoding without rescoring

$\max_t R_{h,t,i}$ denote the peak normalized attention rank, let $\overline{A}_{h,i} = T^{-1}\sum_t A_{h,t,i}$ denote the mean attention, and let $m \in [0,1]$ denote the mean-rank weight. The blended temporal score is

$$u_{h,i} = m\,\mathcal{R}(\overline{A}_{h,:})_i + (1-m)\,\frac{P_{h,i}}{1+d_{h,i}}. \tag{6}$$

Lifetime applies local average pooling to $u_{h,:}$ and then adds a lightweight, head-local value-magnitude rank. Because an attention output is a weighted sum of value vectors, the value norm provides an inexpensive cue for the possible output consequence of retaining an attended entry. Specifically, we define $Q_{h,i} = \mathcal{R}\big((\|V_{h,j}\|_2)_{j=0}^{M-1}\big)_i$ with weight $\beta \in [0,1]$, and for an odd pooling width k we form the provisional full-length score

$$\bar{s}_{h,i} = \begin{cases} \mathrm{AvgPool}_k(u_{h,:})_i + \beta\, Q_{h,i}, & 0 \le i < M, \\ 0, & M \le i < N. \end{cases} \tag{7}$$

Let S and R be the numbers of protected sink and recent positions, and define the protected set

$$\mathcal{P} = \underbrace{\{0, \ldots, \min(S,N)-1\}}_{\mathcal{P}_{\mathrm{sink}}} \cup \underbrace{\{\max(N-R,0), \ldots, N-1\}}_{\mathcal{P}_{\mathrm{recent}}}. \tag{8}$$

Lifetime assigns every $i \in \mathcal{P}$ a common protected score C_h that is strictly larger than $\max_i \bar{s}_{h,i}$, and sets $s_{h,i} = \bar{s}_{h,i}$ elsewhere. Protected positions count toward B; all of them are retained when they fit, and otherwise the backend's tie handling selects among their equal scores. The proximity term is what injects temporal order into this score: with $m = 1$ and $\beta = 0$, $u_{h,i}$ reduces to the rank of the window-averaged attention, an order-invariant statistic in the spirit of SnapKV, so the Shuffled-Time and Reversed-Time ablations of Section 4.5 directly probe this contribution.

Throughout, Lifetime uses one fixed configuration: $W = 16$, $T = 8$, $\theta = 0.95$, $m = 0.35$, $\beta = 0.20$, $k = 5$, $S = 4$, and $R = 64$, with mean GQA-group aggregation. The 16-position observation window supplies scoring evidence, whereas the 64-position recent region controls retention priority and therefore also protects the positions just before the observation window.

3.5 Physical Cache Construction and Complexity

Restoring the layer index, Lifetime independently selects the B highest-scoring positions for each layer and KV head, and materializes the cache by gathering along the sequence dimension:

$$\mathcal{I}_{l,h} = \mathrm{TopB}\big((s_{l,h,i})_{i=0}^{N-1}; B\big), \qquad (K'_{l,h}, V'_{l,h}) = \mathrm{Gather}((K_{l,h}, V_{l,h}), \mathcal{I}_{l,h}). \tag{9}$$

Every layer–head pair receives the same number of entries, but its positions are selected independently and may therefore differ across heads and layers. Equal per-head lengths preserve a regular cache shape, while gathering physically shortens the sequence dimension of the prompt cache instead of retaining a dense eviction mask. Algorithm 1 summarizes the complete procedure.

When the attention backend exposes prefill attention weights, Lifetime reuses the required prompt-tail slice. Otherwise, as with the SDPA path in our implementation, it recomputes attention only for the last W queries rather than materializing an $N \times N$ matrix. Across all layers, this recomputation requires $O(LH_qWNd_h)$ one-time arithmetic, temporal rank normalization requires $O(LH_{\mathrm{kv}}TM \log M)$, and top-$B$ selection requires at most $O(LH_{\mathrm{kv}}N \log B)$ comparisons. As layers are processed sequentially, the transient tensors of one layer scale as $O(H_qWN + H_{\mathrm{kv}}TM)$, excluding existing K/V tensors. After gathering, prompt K/V storage decreases from $O(LH_{\mathrm{kv}}Nd_h)$ to $O(LH_{\mathrm{kv}}Bd_h)$, and decoding proceeds as with a regular cache of length B.

Table 1: LongBench results on 16 tasks at 10% and 5% KV keep ratios under the query-aware, untruncated-prompt protocol. Full KV does not depend on the budget and is listed once per model; **bold** and underlined values mark the best and second-best compressed methods in each block.

Method	Single-Doc QA			Multi-Doc QA			Summarization			Few-Shot			Synthetic		Code		Avg.
	NQA	Qasp.	MFQA	HQA	2Wiki	Mus.	Gov.	QMS	MNews	TREC	Triv.	SAM	Count	Retr.	LCC	Repo	
Qwen3-4B (10% keep ratio)																	
Full KV	31.71	41.22	55.12	66.47	54.32	33.72	30.48	25.14	26.07	75.00	84.35	43.24	4.00	100.00	56.71	59.40	49.18
SnapKV	31.79	34.46	50.92	65.93	48.83	33.05	24.49	24.35	17.59	55.00	86.36	41.14	**7.00**	**100.00**	58.62	**58.79**	46.15
CriticalKV+SnapKV	32.17	36.79	54.03	66.56	48.13	32.62	25.39	24.29	18.47	63.00	86.36	42.12	5.00	**100.00**	58.35	57.27	46.91
Layer-DefensiveKV	29.10	37.65	52.06	64.52	47.77	30.23	24.71	24.09	**19.65**	**68.00**	85.92	42.86	**7.00**	**100.00**	59.02	58.09	46.92
ReST-KV	**32.31**	**38.51**	**54.74**	65.10	47.21	34.07	26.05	**24.45**	19.06	62.00	85.45	**43.26**	6.00	**100.00**	61.99	57.58	47.36
Lifetime	32.07	38.31	52.49	**67.44**	**49.31**	**34.65**	**26.43**	23.77	19.00	**68.00**	**87.11**	42.90	5.00	**100.00**	**62.45**	57.48	**47.90**
Qwen3-4B (5% keep ratio)																	
SnapKV	28.57	33.09	49.43	65.47	45.32	29.36	20.23	23.25	15.20	35.00	**87.79**	40.08	**6.00**	99.50	57.57	58.11	43.37
CriticalKV+SnapKV	29.68	31.68	48.24	63.96	43.21	30.88	20.92	23.46	15.30	44.00	85.46	40.93	**6.00**	**100.00**	57.47	55.96	43.57
Layer-DefensiveKV	31.09	35.18	**50.43**	65.63	**44.96**	**30.91**	22.28	24.04	**16.34**	55.00	84.92	41.05	5.00	**100.00**	55.40	57.29	44.97
ReST-KV	30.07	**36.53**	50.20	63.24	42.40	30.46	**24.01**	**24.60**	15.53	55.00	86.45	41.61	5.00	**100.00**	58.29	57.63	45.06
Lifetime	30.30	33.33	50.15	**65.86**	43.36	28.29	22.51	24.27	16.04	**58.00**	86.50	**41.83**	5.00	**100.00**	58.09	**58.58**	**45.13**
Llama-3.1-8B-Instruct (10% keep ratio)																	
Full KV	35.34	48.66	55.13	60.09	51.67	33.53	34.34	25.13	27.54	74.00	73.57	45.32	9.10	100.00	61.61	58.19	49.58
SnapKV	33.69	43.13	53.67	**58.87**	48.55	30.74	25.67	23.87	20.47	53.00	75.06	43.53	10.08	**100.00**	64.94	60.62	46.62
CriticalKV+SnapKV	32.94	42.40	55.68	58.34	48.32	30.29	26.87	23.91	**22.08**	55.00	77.03	43.38	9.12	**100.00**	65.47	**62.70**	47.10
Layer-DefensiveKV	34.47	**46.49**	53.69	57.11	**52.17**	29.30	**28.71**	24.54	20.13	**72.00**	77.06	**45.53**	11.09	99.00	62.61	62.27	48.51
ReST-KV	33.51	44.17	54.58	58.07	48.55	30.02	27.22	**25.16**	21.34	62.00	77.61	44.07	10.10	**100.00**	**65.90**	58.57	47.55
Lifetime	**36.63**	42.71	**56.08**	57.56	49.54	31.63	27.53	24.67	21.78	70.00	**79.73**	44.23	**12.10**	99.00	63.21	61.13	**48.60**
Llama-3.1-8B-Instruct (5% keep ratio)																	
SnapKV	33.80	36.29	53.07	54.62	44.95	26.25	22.71	23.72	17.93	43.00	76.04	41.12	12.06	99.00	62.22	61.52	44.27
CriticalKV+SnapKV	**35.94**	39.27	53.63	55.58	44.24	28.38	23.72	24.13	18.60	46.00	**79.17**	42.75	10.12	**100.00**	62.68	63.01	45.45
Layer-DefensiveKV	33.90	40.43	52.61	56.50	**49.36**	29.66	**25.57**	24.30	10.74	57.00	74.36	**44.26**	10.00	99.00	48.60	**63.97**	45.02
ReST-KV	33.14	**42.38**	**53.85**	56.69	47.14	29.62	24.87	**24.52**	18.57	54.00	74.89	43.76	9.12	**100.00**	**65.32**	59.63	46.09
Lifetime	34.31	39.40	51.61	57.13	45.05	28.94	24.30	24.27	18.26	**64.00**	77.19	43.22	11.09	99.00	63.11	62.38	**46.45**

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Benchmarks and protocol. We evaluate on all 16 tasks of LongBench (Bai et al., 2024) and on the 16K setting of RULER (Hsieh et al., 2024). For LongBench, we sample 100 instances per task with seed 42 and reuse exactly the same rows for every method. The user query is appended to the context before compression, so every method is query-aware; prompts are kept untruncated (`max_context_length=None`), and the official task-specific prompt formats, generation limits, and scorers are preserved. We evaluate KV keep ratios of 10% and 5%. For RULER, we use model-specific tokenizers, 100 seed-42 instances per task, and official string-match scoring on seven tasks: common- and frequent-word extraction (CWE, FWE), two single-needle retrieval tasks (S1, S2), two question-answering tasks (QA1, QA2), and variable tracking (VT). We use three globally budget-equivalent targets of 256, 512, and 1024 retained entries per layer–KV-head pair: fixed-budget methods, including Lifetime, retain exactly the target number in every KV head, whereas adaptive baselines redistribute the same total budget across heads or layers.

Models, baselines, and implementation. We evaluate Qwen3-4B (Yang et al., 2025) and Llama-3.1-8B-Instruct (Grattafiori et al., 2024) with greedy decoding and batch size one. The main comparison includes Full KV, SnapKV (Li et al., 2024), CriticalKV+SnapKV (Feng et al., 2025), Layer-DefensiveKV (Feng et al., 2026), and ReST-KV (An et al., 2026); AdaKV+SnapKV (Feng et al., 2024), Expected Attention (Devoto et al., 2025), and CapKV (Yang et al., 2026) are additionally reported on LongBench averages and, for AdaKV+SnapKV, on RULER. All compressed methods prune once after prefill with their official or repository-recommended settings (Appendix A.2), and within each setting they receive the same prompts, examples, generation limits, and cache budgets. All methods are implemented in KVPress (Devoto et al., 2025) on Hugging Face Transformers

Table 2: LongBench averages of the additional baselines under the protocol of Table 1, with Full KV and Lifetime listed as anchors.

Method	Qwen3-4B		Llama-3.1-8B	
	10%	5%	10%	5%
Full KV	49.18	49.18	49.58	49.58
Lifetime	**47.90**	**45.13**	**48.60**	**46.45**
AdaKV+SnapKV	46.72	43.81	47.01	45.38
Expected Attn.	33.29	27.27	34.35	30.22
CapKV	34.38	27.60	33.82	28.52

Table 3: RULER-16K averages under globally budget-equivalent targets of 256, 512, and 1024 retained entries per layer–KV-head pair.

Method	Qwen3-4B			Llama-3.1-8B		
	256	512	1024	256	512	1024
Full KV	86.05			89.74		
SnapKV	47.60	66.98	74.97	66.09	73.91	78.70
AdaKV+Snap	43.70	63.89	73.39	63.27	73.25	79.35
CriticalKV	55.60	69.89	**76.53**	67.00	74.31	78.99
Layer-Def.	49.16	67.37	75.08	69.59	78.55	**82.98**
ReST-KV	60.59	70.76	72.13	69.70	77.01	79.66
Lifetime	**70.56**	**71.54**	74.08	**74.87**	**78.86**	82.02

(Wolf et al., 2020) with the SDPA attention backend on one NVIDIA H20 GPU (Appendix A.3), and Lifetime uses the single configuration of Section 3.4 for every model, task, and budget.

4.2 Main Results on LongBench

Table 1 reports task-level LongBench results for both models and retention budgets. Across both model families and budgets, Lifetime achieves the highest 16-task average among compressed methods: 47.90 and 45.13 on Qwen3-4B at 10% and 5% retention, and 48.60 and 46.45 on Llama-3.1-8B-Instruct. The clearest gains appear on Qwen at 10% and on Llama at 5%, where Lifetime leads the next-best method by 0.54 and 0.36 points. Relative to the Full-KV averages of 49.18 and 49.58, Lifetime at 10% retention preserves 97.4% of the Qwen performance and 98.0% of the Llama performance. The advantage is not universal at the task level, where competing methods remain stronger on several tasks and the Qwen 5% aggregate is particularly close. Lifetime ranks first or second on TREC and TriviaQA in all four settings; we conjecture that their few-shot demonstrations are revisited repeatedly, producing the recurring trajectories that the periodic branch targets.

Table 2 completes the comparison with the three baselines omitted from the main table. AdaKV+SnapKV, which adds adaptive head-level allocation to SnapKV, trails Lifetime by 1.07 to 1.59 points in every setting, although Lifetime uses equal per-head budgets. Expected Attention and CapKV, which model future queries or cache capacity instead of observed attention, fall below 35 points at both budgets in this aggressive regime. Overall, Lifetime gives the most consistent aggregate performance across architectures and budgets without changing its configuration.

4.3 Main Results on RULER

Table 3 reports RULER-16K averages under the three cache targets, where CriticalKV and Layer-Def. abbreviate CriticalKV+SnapKV and Layer-DefensiveKV. At the most restrictive target of 256 entries, Lifetime ranks first on both models, exceeding the strongest alternative by 9.97 points on Qwen3-4B and 5.16 points on Llama-3.1-8B-Instruct. At target 512 it again ranks first on both models, with a cross-model mean of 75.20. At target 1024 the differences among the strongest methods narrow, and Lifetime stays within 2.5 points of the best result on each model. Together with LongBench, these results show that one fixed configuration remains effective under both relative and absolute cache constraints, without any per-benchmark or per-model retuning.

Table 4 shows where this margin comes from. On single-needle retrieval and question answering, Lifetime performs on par with the strongest baselines. The decisive difference lies in variable tracking, where Lifetime reaches 97.20 on Qwen3-4B and 88.60 on Llama-3.1-8B-Instruct, compared with at most 43.20 and 62.40 for any baseline. Variable tracking requires following a chain of assignments whose links are revisited at irregular intervals, plausibly the kind of recurring, order-dependent evidence that a mean over the observation window blurs. Common-word extraction, which depends on global frequency statistics rather than on individual entries, remains near failure for every compressed method at this budget, leaving little room to separate methods, whereas Lifetime also attains the best FWE score on both models (67.00 and 91.67).

Table 4: Task-level RULER-16K scores at target 256, where compressed methods differ most; S1 and S2 denote `niah_single_1` and `niah_single_2`, and the Avg. columns match the 256 column of Table 3. The margin of Lifetime comes mainly from variable tracking (VT).

Method	Qwen3-4B								Llama-3.1-8B-Instruct							
	CWE	FWE	S1	S2	QA1	QA2	VT	Avg.	CWE	FWE	S1	S2	QA1	QA2	VT	Avg.
Full KV	81.00	91.33	100.00	100.00	77.00	53.00	100.00	86.05	86.50	91.67	100.00	100.00	86.00	64.00	100.00	89.74
SnapKV	8.10	62.67	76.00	48.00	65.00	48.00	25.40	47.60	1.50	76.33	100.00	95.00	82.00	61.00	46.80	66.09
AdaKV+SnapKV	6.50	57.00	88.00	17.00	59.00	49.00	29.40	43.70	0.90	64.00	100.00	98.00	81.00	60.00	39.00	63.27
CriticalKV+SnapKV	8.70	66.33	99.00	55.00	70.00	47.00	43.20	55.60	2.50	71.33	100.00	98.00	83.00	65.00	49.20	67.00
Layer-DefensiveKV	8.10	51.00	95.00	31.00	71.00	52.00	36.00	49.16	9.70	72.00	100.00	98.00	85.00	60.00	62.40	69.59
ReST-KV	6.60	64.33	100.00	100.00	74.00	49.00	30.20	60.59	0.80	91.33	100.00	100.00	85.00	59.00	51.80	69.70
Lifetime	6.70	67.00	100.00	100.00	72.00	51.00	97.20	70.56	2.80	91.67	100.00	97.00	83.00	61.00	88.60	74.87

Table 5: Efficiency on Qwen3-4B at 32K and 10% retention; median prompt lengths are 29.3K (BS=1) and 30.6K (BS=4) tokens; values are run medians.

Method	BS	Prefill (s)	Peak (GiB)	Decode (tok/s)
Full KV	1	4.780	13.77	46.34
	4	20.353	33.35	67.94
SnapKV	1	4.890	10.15	53.54
	4	20.556	18.25	215.08
Lifetime	1	4.917	10.15	53.47
	4	20.702	18.25	214.90

Table 6: Component ablations on LongBench with Qwen3-4B at 5% retention; Δ is the change relative to Lifetime.

Variant	Score	Δ
Lifetime	**45.13**	–
Shuffled-Time	42.84	-2.29
Reversed-Time	39.27	-5.86
w/o Periodic	37.28	-7.85
w/o Rising	43.37	-1.76
w/o Value	44.02	-1.11
w/o Smoothing	43.07	-2.06
Short Recent	44.91	-0.22

4.4 Efficiency Analysis

We measure runtime efficiency on Qwen3-4B using the real, untruncated, query-aware LongBench prompts nearest the 32K setting, and Table 5 reports medians from repeated paired runs with homogeneous batching. Physical compression substantially lowers peak memory and raises decode throughput relative to Full KV, especially at BS=4, where peak usage falls from 33.35 to 18.25 GiB and throughput rises from 67.94 to about 215 tokens/s, a 45% memory reduction and a $3.2\times$ speedup; at BS=1 the corresponding gains are 26% and 15%. Under the same cache budget, Lifetime adds only 47.3 ms (0.90%) of paired prefill time over SnapKV at BS=1 and 140.1 ms (0.68%) at BS=4, with no measured difference in peak memory. Because fixed per-head gathering gives SnapKV and Lifetime the same final cache shape, their decode costs match, as expected. The two scorers use different temporary workspaces, yet their measured peak memory is identical.

4.5 Ablation and Parameter Sensitivity

We analyze Lifetime on Qwen3-4B at 5% retention on LongBench, using the same paired examples and the official prompt and scoring protocol for all variants; all values are task-level macro averages.

Component analysis. Table 6 evaluates controlled variants around the final configuration, changing one component at a time without retuning the others. Shuffled-Time applies the same per-token permutation to the raw and rank-normalized histories, preserving every order-invariant statistic while changing temporal placement; averaged over two fixed permutation seeds (43.04 and 42.65), it loses 2.29 points. Reversing every trajectory costs 5.86 points, showing that the direction of a trajectory carries information beyond its unordered magnitudes. The branch controls dissect the proximity estimator: w/o Periodic forces every token through the rising branch and loses 7.85 points, whereas w/o Rising assigns the maximum distance to tokens without repeated active bins and loses 1.76 points. Removing the value-rank cue or local smoothing costs 1.11 and 2.06 points, while shortening recent protection from 64 to 16 positions has a smaller effect of 0.22 points. Every controlled variant remains below the final configuration, confirming that each component contributes.

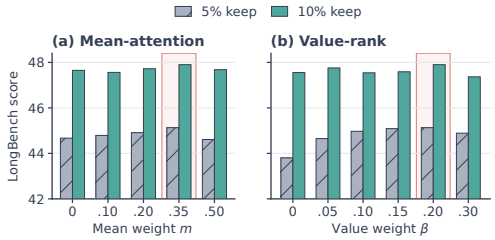


Figure 3: LongBench sensitivity to (a) the mean-rank weight m at $\beta = 0.20$ and (b) the value weight β at $m = 0.35$, at 5% and 10% retention.

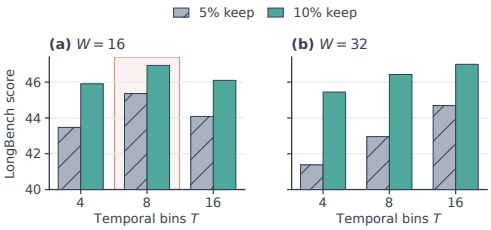


Figure 4: Structural sensitivity to the number of temporal bins T with (a) $W = 16$ and (b) $W = 32$; shading marks the selected $W = 16$, $T = 8$.

Mean and value components. The mean-attention rank stabilizes the trajectory-derived proxy against phase and slope noise from the short window. With $\beta = 0.20$ fixed, Figure 3(a) evaluates $m \in \{0, 0.10, 0.20, 0.35, 0.50\}$: at 5% retention the five settings score 44.67, 44.79, 44.91, 45.13, and 44.61, and at 10% they score 47.65, 47.56, 47.72, 47.90, and 47.68. Thus $m = 0.35$ is best at both budgets, improving over $m = 0$ by 0.46 and 0.25 points. The head-local value rank supplies a magnitude cue that is absent from the attention trajectory; Figure 3(b) reports normalized value-weight sensitivity at $m = 0.35$, derived from auxiliary runs rather than a direct fixed-m rerun, and places $\beta = 0.20$ highest at both budgets, consistent with the selected value weight.

Observation window and temporal bins. We vary the observation length W and the temporal resolution T in an auxiliary structural scan with $m = 0.35$, $\beta = 0.20$, and all other components, including the 64 protected recent positions, held fixed. With $W = 16$, $T = 8$ scores 45.36 and 46.93 at 5% and 10% retention; because this scan was run separately from the final evaluation, we compare its settings only with one another. At 5%, $T = 8$ exceeds $T = 4$ and $T = 16$ by 1.89 and 1.28 points, and at 10% the margins are 1.02 and 0.84 points. This is plausibly a resolution trade-off: $T = 4$ leaves too few bins to resolve periodic spacing, whereas $T = 16$ reduces each bin to a single query and makes the active-bin statistics noisier. The $W = 32$, $T = 16$ setting reaches 44.69 and 47.00, trailing the selected setting by 0.67 points at 5% and leading by only 0.06 points at 10% while using twice as many observation queries. We therefore retain $W = 16$, $T = 8$, with two queries per bin, as a compact operating point; the scans serve configuration selection only.

5 Conclusion

We presented Lifetime, a training-free, query-aware, prefill-only method for KV-cache compression. Rather than collapsing the prompt-tail observation window into a single static statistic, Lifetime preserves the ordered attention trajectory of each KV head, estimates a next-use proximity proxy through complementary periodic and rising branches, and fuses it with mean-attention rank and a lightweight value-rank cue before one-shot selection. Across Qwen3-4B and Llama-3.1-8B-Instruct, a single fixed configuration attains the highest compressed-method average on LongBench at both retention budgets and leads all baselines under the tightest RULER budget, while adding less than 1% prefill latency over SnapKV and matching its decode cost. These results indicate that the temporal structure of prompt-tail attention is a practical signal for aggressive one-shot compression, and that eviction scores should consider when attention occurs, not only how much.

Limitations and future work. Lifetime currently assigns equal per-head budgets and compresses once after prefill, so it cannot revise its decision if generation drifts toward context that the prompt tail never emphasized. Its proximity proxy is a heuristic estimated from a short window rather than a calibrated forecast of future attention, and our evaluation covers two models of up to 8B parameters at contexts of up to about 32K tokens. Moreover, our latency measurements use the SDPA backend; under fused kernels that never expose attention weights, the prompt-tail slice must likewise be recomputed, and the relative overhead may differ. Combining temporal scoring with adaptive budget allocation, extending it to decode-time cache revision, and validating it on larger models and on longer contexts are natural and promising directions for future work.

Reproducibility Statement

Section 3 and Algorithm 1 specify every step of Lifetime, and Section 3.4 lists its single configuration. Section 4.1 and Appendix A describe the benchmarks, sampled instances, prompts, budgets, and baseline settings, and Appendix A.3 documents the hardware and software stack; anonymized code will be provided as supplementary material to reproduce every reported table.

AI use statement

In this work, we used generative AI tools solely to assist in searching for relevant baseline methods. We did not use generative AI tools for any other tasks. We manually reviewed the search results and checked them against the original sources. We take responsibility for the final content of this work, including any content informed by AI-assisted searches.

Ethics Statement

This work proposes a training-free method to reduce the memory of long-context inference by evicting entries from the KV cache. It involves no human subjects, crowdsourcing, or collection of personal data. All experiments use publicly available benchmarks, LongBench and RULER, and publicly released models, Llama-3.1-8B-Instruct and Qwen3-4B, each used under its original license. Lifetimedoes not train or fine-tune any model and does not add new capabilities; it only changes which cached positions are kept, so the outputs inherit the biases and limitations of the underlying models. Because any compression of the cache can alter or degrade responses, users should validate compressed inference before deploying it in high-stakes applications. By lowering the memory and cost of long-context inference, this work may make long-document and retrieval-augmented applications more accessible and energy-efficient. We are not aware of other ethical concerns raised by this work.

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

Yongqi An, Chang Lu, Kuan Zhu, Tao Yu, Chaoyang Zhao, Hong Wu, Ming Tang, and Jinqiao Wang. ReST-KV: Robust KV cache eviction with layer-wise output reconstruction and spatial-temporal smoothing. In *International Conference on Learning Representations*, 2026.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 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)*, pp. 3119–3137, 2024.

L. A. Belady. A study of replacement algorithms for a virtual-storage computer. *IBM Systems Journal*, 5(2):78–101, 1966.

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

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

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

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*, volume 35, 2022.

Alessio Devoto, Maximilian Jeblick, and Simon Jégou. Expected attention: KV cache compression by estimating attention from future queries distribution. *arXiv preprint arXiv:2510.00636*, 2025.

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. *arXiv preprint arXiv:2407.11550*, 2024.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S. Kevin Zhou. Identify critical KV cache in LLM inference from an output perturbation perspective. *arXiv preprint arXiv:2502.03805*, 2025.

Yuan Feng, Haoyu Guo, Junlin Lv, S. Kevin Zhou, and Xike Xie. Taming the fragility of KV cache eviction in LLM inference. In *International Conference on Learning Representations*, 2026.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024.

Jang-Hyun Kim, Jinuk Kim, Sangwoo Kwon, Jae W. Lee, Sangdoo Yun, and Hyun Oh Song. KVzip: Query-agnostic KV cache compression with context reconstruction. *arXiv preprint arXiv:2505.23416*, 2025.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, 2023.

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*, volume 37, 2024.

Xiang Liu, Zhenheng Tang, Peijie Dong, Zeyu Li, Bo Li, Xuming Hu, and Xiaowen Chu. ChunkKV: Semantic-preserving KV cache compression for efficient long-context LLM inference. *arXiv preprint arXiv:2502.00299*, 2025.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for LLM KV cache compression at test time. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

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

Matanel Oren, Michael Hassid, Yossi Adi, and Roy Schwartz. Transformers are multi-state RNNs. *arXiv preprint arXiv:2401.06104*, 2024.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*, 2020.

Ishan Shah, Akanksha Jain, and Calvin Lin. Effective mimicry of Bélády's MIN policy. *IEEE Micro*, 43(4):45–52, 2023.

Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

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.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pp. 5998–6008, 2017.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, 2020.

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

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. DuoAttention: Efficient long-context LLM inference with retrieval and streaming heads. In *International Conference on Learning Representations*, 2025.

Yuhui Xu, Zhanming Jie, Hanze Dong, Lei Wang, Xudong Lu, Aojun Zhou, Amrita Saha, Caiming Xiong, and Doyen Sahoo. ThinK: Thinner key cache by query-driven pruning. In *International Conference on Learning Representations*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Jiaming Yang, Chenwei Tang, Liangli Zhen, and Jiancheng Lv. Rethinking KV cache eviction via a unified information-theoretic objective. *arXiv preprint arXiv:2604.25975*, 2026.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Xuan Zhang, Cunxiao Du, Chao Du, Tianyu Pang, Wei Gao, and Min Lin. SimLayerKV: A simple framework for layer-level KV cache reduction. *arXiv preprint arXiv:2410.13846*, 2024.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark W. Barrett, Zhangyang Wang, and Beidi Chen. H_2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

A Evaluation Protocol and Baseline Configurations

A.1 Datasets and Protocol

LongBench. We evaluate all 16 LongBench tasks: NarrativeQA, Qasper, MultiFieldQA-en, HotpotQA, 2WikiMultihopQA, MuSiQue, GovReport, QMSum, MultiNews, TREC, TriviaQA, SAMSum, PassageCount, PassageRetrieval-en, LCC, and RepoBench-P. For every method, we sample 100 instances per task with seed 42 and reuse the same rows across methods. The user query is appended before prefill compression; prompts are not manually truncated, task-specific prompt formats and generation limits are preserved, and the official LongBench scorer is used. Model chat templates are applied under the task protocol, except that TREC, TriviaQA, SAMSum, LCC, and RepoBench-P retain the official plain-prompt formats of the original benchmark.

RULER. We use model-specific tokenizers at a 16K context length and evaluate 100 seed-42 instances per task with official string-match scoring. The evaluated suite comprises `cwe`, `fwe`, `niah_single_1`, `niah_single_2`, `qa_1`, `qa_2`, and `vt`. The query is included before compression and prompts are not manually truncated. Fixed-budget methods retain the specified number of entries per layer–KV-head pair, whereas adaptive methods receive the same global token budget.

A.2 Baseline Configurations

Unless otherwise stated, all compressed methods are query-aware, prune once after prefill, use greedy decoding, and follow the settings recommended by their papers or repositories. SnapKV uses a 32-query observation window and average-pooling kernel 5. AdaKV+SnapKV uses the same window, max-pooling kernel 7, and safeguard ratio 0.5. CriticalKV+SnapKV uses window 32, max-pooling kernel 7, and first-stage ratio 0.5. Layer-DefensiveKV uses window 32, kernel 5, and critical mass 0.9. ReST-KV uses window 32, EMA coefficient 0.3, adaptive spatial-pooling scale 2000, and output-projected reconstruction; its kernel is 5 on LongBench and 21 on RULER. Expected Attention uses covariance and value-norm terms, $\epsilon = 2 \times 10^{-2}$, 512 future RoPE positions, and AdaKV allocation. CapKV uses $\tau = 5$, 512 future positions, and four sink tokens. Lifetime uses $W = 16$, $T = 8$, $\theta = 0.95$, $m = 0.35$, $\beta = 0.20$, pooling kernel 5, four sink tokens, 64 protected recent tokens, mean GQA-group aggregation, fixed per-head budgets, and physical K/V gathering.

A.3 Computing Infrastructure

All experiments run on a single NVIDIA H20 GPU (97 GiB) under Debian GNU/Linux 12 with CUDA 12.8 and NVIDIA driver 550.127.08. The software stack uses Python 3.10, PyTorch 2.9.1, Hugging Face Transformers 4.57, and Accelerate 1.14, with all methods implemented in KVPress. Attention is computed through the scaled dot-product attention (SDPA) backend, so Lifetime recomputes the prompt-tail attention slice rather than reading exposed prefill weights.