# WRITEKV: QUERY-AGNOSTIC KV CACHE EVICTION VIA NATIVE WRITES IN HYBRID ATTENTION MODELS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

During autoregressive inference, transformer-based large language models (LLMs) cache the key and value (KV) representations of previously processed tokens to avoid recomputing historical representations. In hybrid-attention models, linear-attention layers maintain a fixed-size recurrent state, whereas full-attention layers retain an ever-growing KV cache whose memory and per-step attention costs increase with the history length. This coexistence suggests that the update dynamics of linear-attention memory may provide a query-independent signal for compressing the full-attention KV cache. We observe that the native writes generated when linear-attention layers update their fixed-size states are conditioned on the current memory contents: writes aligned with existing memory directions are more likely to be redundant, whereas writes that introduce distinct directions are more likely to encode new memory content. Thus, the novelty of a write direction relative to the existing memory can serve as a retention signal for full-attention KV entries before future queries arrive. Based on this observation, we propose WRITEKV, a query-agnostic KV eviction method that quantifies the novelty of write directions using blockwise Ridge statistics and physically compresses the full-attention KV cache through global layer-token selection. WRITEKV requires neither additional context reconstruction nor a trained scoring model, and the resulting cache can be reused by multiple queries over the same history. Across three multi-query long-context benchmarks, WRITEKV outperforms all evaluated KV-cache eviction baselines under a 20% KV budget. In system measurements with an 800K-token prefix at the same budget, it reduces decode-time attention latency to 26.8% of the full-KV attention latency, while increasing cache-construction time by only 4.20% relative to directly pre-filling the full KV cache.

## 1 INTRODUCTION

Long-context capabilities allow large language models (LLMs) to exploit increasingly long conversation histories, external knowledge, and interaction traces from agentic systems (Bai et al., 2024; Wu et al., 2024; Yao et al., 2022). During autoregressive inference, full-attention layers cache the key and value (KV) representations of previously processed tokens, causing both the memory footprint of the KV cache and the per-step attention cost to grow with the context length (Dao et al., 2022). Hybrid-attention architectures reduce this length-dependent state by replacing some full-attention layers with linear-attention layers that maintain fixed-size states, but the remaining full-attention layers still retain growing KV caches (Kwon et al., 2023). KV-cache eviction can complement this architectural strategy by removing redundant historical positions (Liu et al., 2023; Li et al., 2024). This is particularly useful in multi-turn dialogue, retrieval-augmented generation (RAG), and agentic systems, where the same history often serves as a shared prefix for multiple downstream queries (Wu et al., 2024; Lewis et al., 2020; Yao et al., 2022). Reusing the inference state of this prefix avoids repeatedly processing the same history; therefore, eviction should be performed before query-specific branches are formed so that all queries can reuse the same compact cache (Datta et al., 2026).

Most KV-cache eviction methods estimate token importance from current-query attention (Zhang et al., 2023; Liu et al., 2023; Tang et al., 2024; 2025). Decode-time methods update the cache using generated-token attention statistics, whereas prefill-stage methods use queries or attention patterns from an observation window to select historical positions (Li et al., 2024; Tang et al., 2025; Xu et al., 2025). Although these query-dependent methods provide task-relevant signals, their retained entries



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.

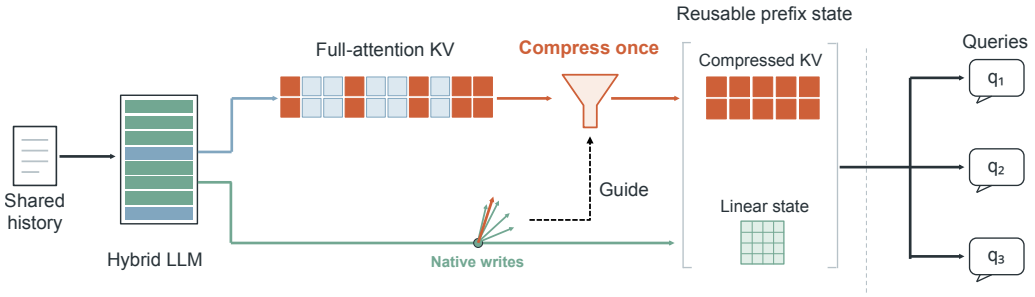



Figure 1: **Query-agnostic KV compression from native writes.** For each shared history, WRITEKV uses native writes from linear-attention layers to select and physically compact the full-attention KV cache once, while leaving the linear recurrent state unchanged. The resulting prefix state is reused across future queries.

are tied to a particular query or generation trajectory. Reconstructing a cache separately for each query forfeits the benefit of history reuse, while reusing one query's cache may omit information required by other queries. Recent work has begun to explore reusable KV compression without relying on future queries (Kim et al., 2026; Bui et al., 2026; Datta et al., 2026; Yang et al., 2026). Existing approaches obtain retention signals through different mechanisms: some reconstruct the context or rely on independently trained scoring models, incurring additional inference or training overhead, whereas others use lightweight static K/V geometry that does not explicitly capture the model's evolving, history-conditioned memory state (Park et al., 2025; Datta et al., 2026; Sengupta et al., 2025; Yang et al., 2026). This motivates a central question: *Can the model's own history-processing dynamics provide an effective and reusable KV-retention signal for full-attention KV before future queries arrive, without requiring additional context reconstruction or separate training?*

The hybrid memory architecture (Katharopoulos et al., 2020; Lieber et al., 2024; Yang et al., 2025) in Figure 1 provides a natural opportunity to address this question: full-attention and linear-attention layers process the same history but use different memory mechanisms. Full-attention layers preserve per-token KV representations, whereas linear-attention layers continuously write historical information into a fixed-size state. As each token arrives, a linear-attention layer produces a native write conditioned on its existing state, forming a complete write trajectory before any future query arrives. Updates aligned with directions repeatedly represented in the past are more likely to reproduce existing content, whereas updates toward directions that are not yet well covered may contribute new memory components. We thus transform the write trajectory of linear memory into a retention signal for the full-attention KV cache, allowing one memory module to guide cache selection in the other.

Building on this observation, we propose WRITEKV, a query-agnostic KV-cache eviction method for hybrid-attention models. WRITEKV estimates the memory novelty contributed by each historical position from native writes, calibrates and aggregates scores across heads and associated linear-attention layers, and uses a unified layer-token budget to construct a retention mask shared across all KV heads within each full-attention layer. This mask physically compacts the full-attention KV cache. The procedure uses only information produced during the history forward pass and requires neither access to future queries, additional context reconstruction, nor an independently trained scoring model. The resulting cache can serve multiple downstream queries sharing the same history.

Across three multi-query long-context benchmarks, WRITEKV outperforms all evaluated KV-cache eviction baselines under a 20% KV budget. In system measurements with an 800K-token prefix at this budget, it reduces decode-time attention latency to 26.8% of the full-KV attention latency, while increasing cache-construction time by only 4.20% relative to directly pre-filling the full KV cache. This shows that native writes provide effective, low-overhead, and reusable retention signals for query-agnostic KV compression.

Our main contributions are threefold. First, we introduce a cross-memory perspective on KV compression that converts history-conditioned writes from linear-attention layers into query-agnostic retention signals for full-attention KV. Second, we propose WRITEKV, which combines block-wise Ridge novelty scoring, aggregation across heads and associated linear-attention layers, and unified layer-token selection to construct reusable physical KV caches. Third, we systematically evaluate

WRITEKV on multi-query long-context tasks, demonstrating near-Full-KV performance under tight budgets, low cache-construction overhead, and decoding speedups from physical cache compaction.

# 2 PROBLEM SETTING AND MODEL BACKGROUND

## 2.1 QUERY-AGNOSTIC KV-CACHE COMPRESSION

We study KV-cache compression for the full-attention layers of hybrid-attention models. Let $\mathcal{F}$ denote the set of full-attention layers. Given a historical context $c$ of length $T$, the model produces a cache $\mathrm{KV}_c$ after the forward pass, containing the key and value representations for historical positions in each full-attention layer (Vaswani et al., 2017; Shazeer, 2019; Oren et al., 2024). Our goal is to compress this cache into a smaller cache $\widehat{\mathrm{KV}}_c$ while preserving downstream task performance as much as possible (Nawrot et al., 2024).

Compression is performed after processing the historical context and before any downstream query arrives, so the retention algorithm never observes the queries to be answered (Kim et al., 2025; Bui et al., 2026). For multiple queries sharing the same history, each query independently starts inference from the same compressed KV cache and underlying linear recurrent state; no state is accumulated across queries (Maharana et al., 2024; Wu et al., 2024). Because different queries may access different historical positions, the compression decision cannot depend on any particular query. We refer to this setting as *query-agnostic* compression.

WRITEKV uses a *layer-token* unit as its selection unit. Each unit consists of the key and value vectors for one historical position in a given full-attention layer across all KV heads. The KV heads within the same layer share a token mask, whereas different layers may retain different positions and token counts. Given a retention ratio $\rho$, all compressible units share a global budget. The budget definition, protected positions, and physical compression procedure are described in Section 3.3.

## 2.2 MEMORY STRUCTURE IN HYBRID ATTENTION

A hybrid-attention model also contains linear-attention layers; let $\mathcal{L}$ denote their set. Full-attention layers store per-token KV representations for the history, whereas linear-attention layers aggregate historical information into a recurrent state $\mathcal{S}_c$ whose size does not grow with the context length (Sun et al., 2023; Dao & Gu, 2024). WRITEKV compresses only the full-attention KV cache and leaves the linear recurrent state unchanged.

During the normal forward pass of a linear-attention layer, each position produces a native write conditioned on the accumulated state, without requiring an additional forward pass over the context. Section 3 describes how to construct novelty scores from these writes and convert them into retention decisions for the full-attention KV cache.

# 3 WRITEKV

WRITEKV converts native writes from linear-attention layers into query-agnostic retention scores for full-attention KV. As illustrated in Figure 2, it computes block-wise write novelty, aggregates the resulting evidence across heads and observer layers, and globally selects layer–token units to physically compact the KV cache. Because all signals are collected during the history forward pass, the resulting cache can be reused across future queries.

## 3.1 NATIVE WRITES AS RETENTION SIGNALS

Full-attention and linear-attention layers process the same history but store it differently: the former retain per-token KV, whereas the latter recurrently write information into a fixed-size state. Each position produces a native write conditioned on the accumulated state, yielding a complete write trajectory before any future query arrives.

**Key insight.** Because a native write is the residual produced after reading the existing state, its direction reflects what the current token attempts to add beyond the memory already represented. We assess this update relative to the write trajectory: well-covered directions are more likely to reproduce existing memory, whereas weakly covered directions are more likely to introduce distinct memory components. The resulting directional novelty provides a history-conditioned, query-independent signal for retaining the corresponding full-attention KV.

To formalize this signal, consider a linear layer $\ell$ and head $h$, and omit these subscripts for clarity. Let $\mathbf{S}_{t-1} \in \mathbb{R}^{d_k \times d_v}$ be its recurrent content state before position $t$, and let $\mathbf{k}_t$ and $\mathbf{v}_t$ be the current key and value projections. The native Gated DeltaNet update is

$$\begin{aligned} \bar{\mathbf{S}}_{t-1} &= \alpha_t \mathbf{S}_{t-1}, & \mathbf{w}_t &= \beta_t \left(\mathbf{v}_t - \bar{\mathbf{S}}_{t-1}^\top \mathbf{k}_t\right), \\ \mathbf{S}_t &= \bar{\mathbf{S}}_{t-1} + \mathbf{k}_t \mathbf{w}_t^\top, & \mathbf{u}_t &= \frac{\mathbf{w}_t}{\|\mathbf{w}_t\|_2}. \end{aligned} \tag{1}$$

Here, $\alpha_t$ and $\beta_t$ control state decay and write strength. The value-side write $\mathbf{w}_t$ corresponds to the native intermediate $\mathbf{v}_t^{\text{new}}$, while $\mathbf{k}_t \mathbf{w}_t^\top$ is the full state increment. To isolate directional novelty from write magnitude, WRITEKV normalizes $\mathbf{w}_t$ and scores its unit direction $\mathbf{u}_t$.

Figure 3 illustrates this observation on four queries sharing a LongMemEval-20K history. Full-KV attention from the queries concentrates on different context regions during question processing and answer generation, illustrating that future access is query-specific. Nevertheless, after attention and write-novelty scores are converted to percentile ranks, positions receiving strong future attention tend to rank higher under query-independent write novelty. This shared tendency suggests that native-write novelty can serve as a shared KV-retention signal across distinct future queries.

To operationalize this insight, we assign each write a Ridge novelty score based on how weakly its direction is covered by reference writes. Section 3.2 defines the block-wise form used by WRITEKV.

### 3.2 BLOCK-WISE RIDGE SCORING

WRITEKV measures how weakly each write direction is covered using a Ridge score based on cumulative second-order statistics. To make this computation efficient for long histories, it partitions the context into ordered blocks $\mathcal{B}_1, \ldots, \mathcal{B}_m$ and updates the statistics incrementally. For linear layer $\ell$ and head $h$, block $b$ is scored in parallel as

$$\begin{aligned} \mathbf{G}_{\ell,h}^{(b)} &= \mathbf{G}_{\ell,h}^{(b-1)} + \sum_{j \in \mathcal{B}_b} \mathbf{u}_{\ell,h,j} \mathbf{u}_{\ell,h,j}^\top, \quad \mathbf{G}_{\ell,h}^{(0)} = \mathbf{0}, \\ \nu_{\ell,h,t} &= \mathbf{u}_{\ell,h,t}^\top \left(\mathbf{G}_{\ell,h}^{(b)} + \lambda_{\ell,h}^{(b)} \mathbf{I}\right)^{-1} \mathbf{u}_{\ell,h,t}, \quad t \in \mathcal{B}_b. \end{aligned} \tag{2}$$

The Gram matrix encodes directional coverage from all preceding blocks and the complete current block, but no future blocks; positions within $\mathcal{B}_b$ therefore share the same reference statistics. The regularizer $\lambda_{\ell,h}^{(b)} > 0$ stabilizes the inverse; Appendix A specifies its adaptive selection and numerical implementation. Because the inverse Gram matrix downweights well-covered directions, a larger $\nu_{\ell,h,t}$ identifies a less-covered, more novel direction; it does not guarantee relevance to any future query.

After scoring, the block directions are released while the Gram matrix is retained. Larger blocks require fewer solves and improve parallelism, but increase temporary memory and alter the reference scope; Section 4.4 evaluates this trade-off.

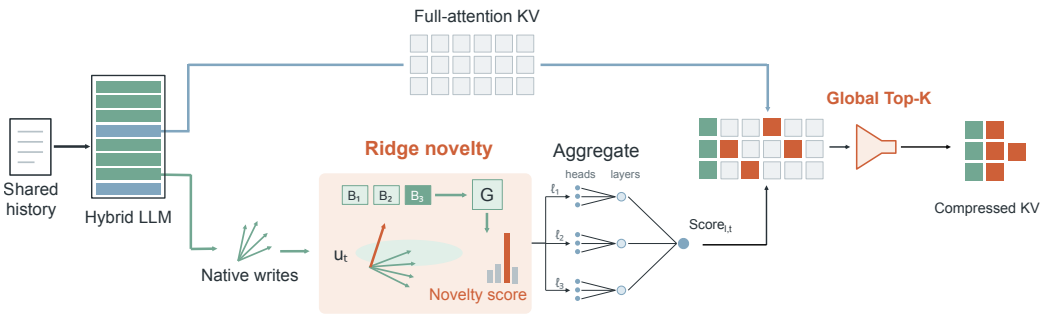



Figure 2: **Overview of WRITEKV.** Native writes produced while encoding the history are scored for block-wise novelty, aggregated into full-attention layer–token scores, and globally selected to construct a compact, reusable KV cache.

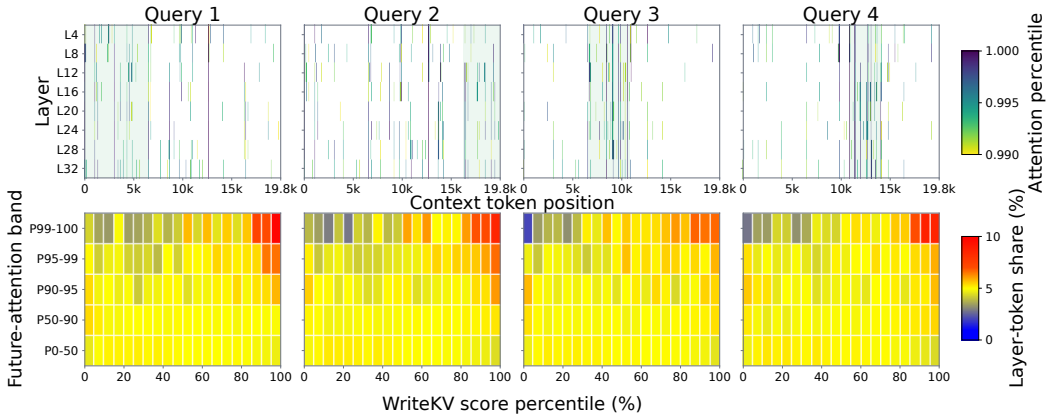



Figure 3: **WriteKV scores align with query-specific future access.** Top: Full-KV attention percentiles over context positions and full-attention layers for four queries sharing one LongMemEval-20K history. Bottom: the distribution of layer–token units over WriteKV score percentiles, stratified by future-attention percentile. Higher-attention bands place more mass on higher WriteKV ranks across all four queries.

### 3.3 Score Aggregation and Physical KV Retention

**Score aggregation.** Raw Ridge scores are not directly comparable across heads, blocks, and linear layers. WriteKV converts this heterogeneous evidence into one score for each full-attention layer–token pair. Let $\mathcal{P}$ and $\mathcal{C}$ denote the protected system-token positions and the remaining compressible positions, respectively, $\mathcal{B}(t)$ the block containing $t$, and $\mathcal{S}(L)$ the linear-layer observer set for full-attention layer $L$. We compute

$$\begin{aligned} a_{\ell,h,t} &= \mathrm{PctRank}_{\mathcal{B}(t)}(\nu_{\ell,h,t}), \quad & m_{\ell,t} &= \max_h a_{\ell,h,t}, \\ z_{\ell,t} &= \mathrm{PctRank}_{\mathcal{C}}(m_{\ell,t}), & s_{L,t} &= \max_{\ell \in \mathcal{S}(L)} z_{\ell,t}. \end{aligned} \tag{3}$$

The first percentile rank removes head- and block-specific scale differences, while the second calibrates the head-aggregated scores within each linear layer. Max aggregation preserves strong evidence from any head or observer layer. For an interleaved hybrid stack, we construct $\mathcal{S}(L)$ by assigning each linear-attention layer to the next full-attention layer in model depth; this architecture-defined mapping is fixed across inputs. The resulting $s_{L,t}$ is shared by all KV heads in layer $L$, requiring no head-wise correspondence between the two memory modules.

**Global selection and compaction.** For retention rate $\rho$ over $\mathcal{C}$, WriteKV globally selects layer–token candidates across all full-attention layers $\mathcal{F}$:

$$\begin{aligned} K_\rho &= \lceil \rho\, |\mathcal{F}|\, |\mathcal{C}| \rceil\,, \\ \mathcal{A}_\rho &= \mathrm{TopK}_{(L,t) \in \mathcal{F} \times \mathcal{C}}(s_{L,t}, K_\rho), \\ I_L &= \mathcal{P} \cup \{ t \in \mathcal{C} : (L,t) \in \mathcal{A}_\rho \}\,. \end{aligned} \tag{4}$$

Selection is global over $\mathcal{F} \times \mathcal{C}$ rather than constrained by equal per-layer quotas, allowing novelty evidence to allocate the compressible budget non-uniformly across layers; $K_\rho$ excludes protected positions. The system then gathers each layer's selected K/V in their original order, preserves position metadata, and releases the remainder, yielding a physically compact cache reusable across queries.

## 4 Experiments

We evaluate WriteKV from four perspectives: task quality, robustness across context lengths and model scales, the scoring mechanism, and system efficiency. We first examine its task performance and applicability under a query-agnostic protocol. We then use ablation studies to analyze the roles of native write signals and Ridge-based novelty scoring. Finally, we evaluate the one-time preparation cost and runtime benefits along the cache preparation and usage pipeline. Together, these experiments

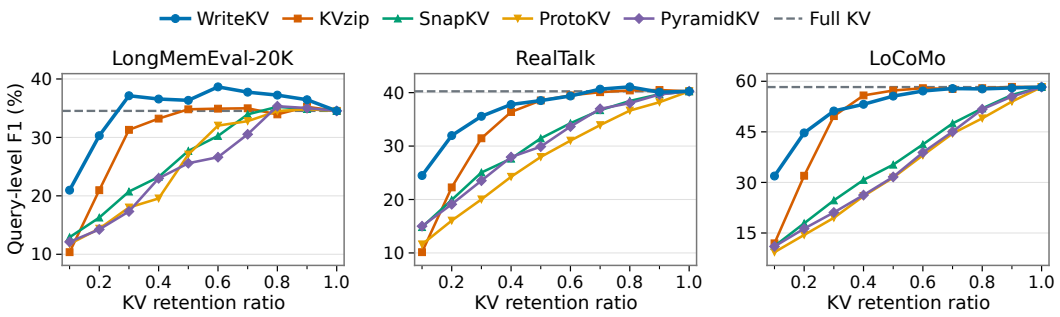



Figure 4: Quality–retention curves for Qwen3.5-9B on LongMemEval-20K, RealTalk, and LoCoMo. Dashed lines indicate Full-KV, and each panel uses an independent vertical axis.

investigate whether WRITEKV can use native writes from a hybrid-attention model to build a reusable compact KV cache without access to future queries.

### 4.1 EXPERIMENTAL SETUP AND EVALUATION PROTOCOL

**Datasets and models.** We evaluate WRITEKV on three long-conversation memory datasets: LoCoMo (Maharana et al., 2024), RealTalk (Lee et al., 2025), and LongMemEval (Wu et al., 2024). LoCoMo contains persona-grounded, multi-session conversations generated through a human–machine pipeline; RealTalk contains real-world, multi-day human–human messaging dialogues; and LongMemEval contains length-configurable, multi-session user–assistant conversations. LongMemEval covers five context-length settings from 20K to 100K. LoCoMo and LongMemEval use normalized token-overlap F1, while RealTalk uses its official lexical F1. We report the query-level mean F1 over all queries. The main experiments use Qwen3.5-9B, which supports the 20K–100K evaluation, while the scaling experiments additionally use the 4B and 27B models from the same family (Qwen Team, 2026). These models permit context-length and scale evaluation under a consistent hybrid architecture and single-A100 BF16 setup. Other hybrid checkpoints do not provide the same controlled setting: Olmo Hybrid 7B (Merrill et al., 2026) has a 32K native context, while Qwen3-Next 80B (Qwen Team, 2025) and Kimi Linear 48B (Team et al., 2025) are MoE models that would introduce architectural differences alongside model scale.

**Evaluation protocol and budget.** Following the query-agnostic setting of KVzip (Kim et al., 2025), each history is scored and compressed independently before its queries arrive. All subsequent queries for that history start independently from the same compressed KV cache and base linear recurrent state, without recomputing the retention mask or accumulating state across queries. At each retention rate, all methods use exactly the same KV budget, including observation windows and positions that are always retained. The requested retention rate $r \in \{0.1, \ldots, 0.9\}$ applies to compressible positions, while $r = 1.0$ retains all candidate KV entries.

**Baselines.** To cover complementary retention signals, we compare WRITEKV with KVzip (Kim et al., 2025), SnapKV (Li et al., 2024), PyramidKV (Cai et al., 2024), ProtoKV (Yu et al., 2026), and Full-KV. KVzip obtains importance scores through an additional context self-reconstruction pass. SnapKV scores tokens using attention probabilities from the last 32 historical tokens, followed by max pooling with a window size of 7. PyramidKV reuses these scores and allocates layer-wise budgets with $\beta = 20$. ProtoKV constructs key-space geometric prototypes and scores them using the post-RoPE queries of the last 32 historical tokens.

**Implementation.** All experiments run on a single NVIDIA A100-SXM4-80GB GPU using BF16 precision, batch size 1, greedy decoding, and a maximum generation length of 96 tokens. All methods use chunked prefill with a chunk size of 1024 tokens. The scoring block length of WRITEKV is 4096.

### 4.2 OVERALL RESULTS AND ROBUSTNESS

**Overall results.** As shown in Figure 4, at $r = 0.1$, WRITEKV achieves F1 scores of 20.97%, 24.49%, and 31.90% on LongMemEval-20K, RealTalk, and LoCoMo, respectively. These results exceed the strongest compression baseline by 8.05, 9.48, and 19.96 percentage points. As the budget increases, WRITEKV gradually approaches Full-KV on RealTalk and LoCoMo. On LongMemEval-20K, its F1 score exceeds Full-KV by 1.81–4.11 percentage points over $r = 0.3$–$0.9$.

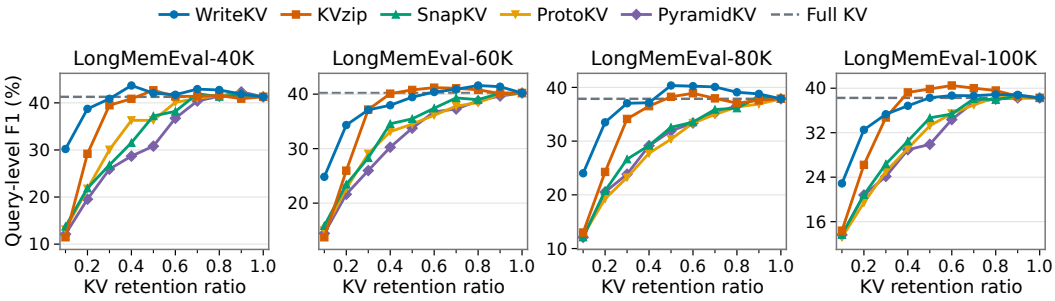



Figure 5: Quality–retention curves for Qwen3.5-9B on the 40K, 60K, 80K, and 100K LongMemEval settings. Dashed lines indicate Full-KV for each setting.

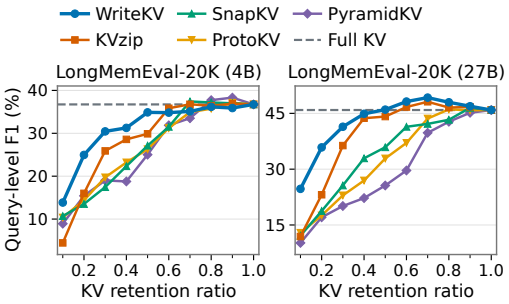



Figure 6: Model scaling on LongMemEval-20K. Left: Qwen3.5-4B; right: Qwen3.5-27B.

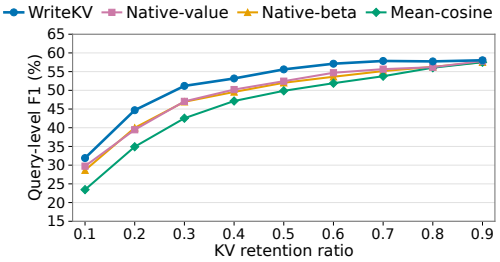



Figure 7: Ablation of the scoring mechanism with Qwen3.5-9B on LoCoMo.

**Context length.** At $r = 0.1$, Figure 5 shows that WRITEKV performs best at all four context lengths and exceeds the strongest compression baseline by 8.49–16.45 percentage points. The gaps between methods become smaller at medium and high retention rates. Because the query sets differ across length settings, we compare methods only within the same setting. Qwen3.5-4B shows the same trend across context lengths (Appendix B.3).

**Model scaling.** Figure 6 reports results for Qwen3.5-4B and Qwen3.5-27B on LongMemEval-20K. At $r = 0.1$, WRITEKV achieves the highest F1 for both models, showing that its tight-budget advantage extends across model sizes. Results on RealTalk and LoCoMo follow the same overall trend and are reported in Appendix B.2.

### 4.3 Ablation of the Scoring Mechanism

**Ablation design.** We study the write signal, novelty measure, and subsequent evidence aggregation using Qwen3.5-9B on LoCoMo. WRITEKV applies block-causal Ridge scoring to the normalized native-write direction $\mathbf{v}_t^{\text{new}}$. Mean-cosine uses the same signal but replaces Ridge scoring with cosine distance from the cumulative mean direction. Native-value and Native-beta keep the rest of the pipeline unchanged but replace the scoring signal with the value projection $\mathbf{v}_t$ and the write gate $\beta_t$, respectively. To examine the aggregation design, we additionally replace the two max operations in Equation 3 with arithmetic means, yielding the Mean/Mean variant.

**Results.** As shown in Figure 7, at $r = 0.1$, WRITEKV achieves an F1 score of 31.90%, while Native-value, Native-beta, and Mean-cosine achieve 29.74%, 28.65%, and 23.45%, respectively. With the same write direction, Ridge scoring improves over Mean-cosine by 8.44 percentage points, and WRITEKV remains the best method at every retention rate from $r = 0.1$ to 0.9. These results support both the normalized native-write signal and the Ridge novelty measure. The default max aggregation also outperforms Mean/Mean under tight budgets, supporting its use to preserve strong evidence from any head or observer layer; see Appendix B.1.

### 4.4 Block-Wise Trade-offs and System Efficiency

The preceding experiments focus on task quality and the scoring mechanism. The practical value of a reusable cache also depends on its preparation cost and usage efficiency. We therefore first analyze

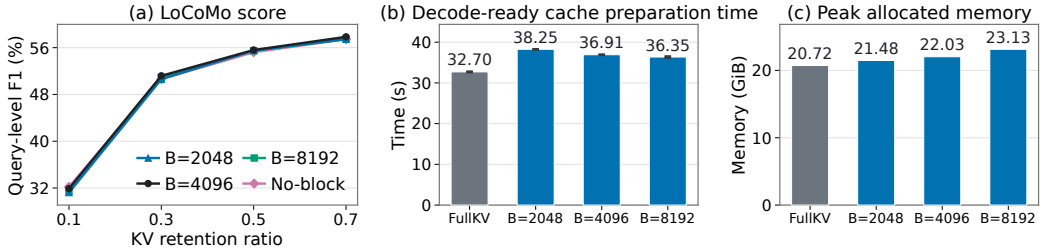



Figure 8: Quality and system trade-offs across block settings. (a) Quality on LoCoMo; (b) cache preparation time; and (c) peak memory during cache preparation. No-block scores the complete context retrospectively, and Full-KV is the no-scoring reference. Time error bars show the sample standard deviation.

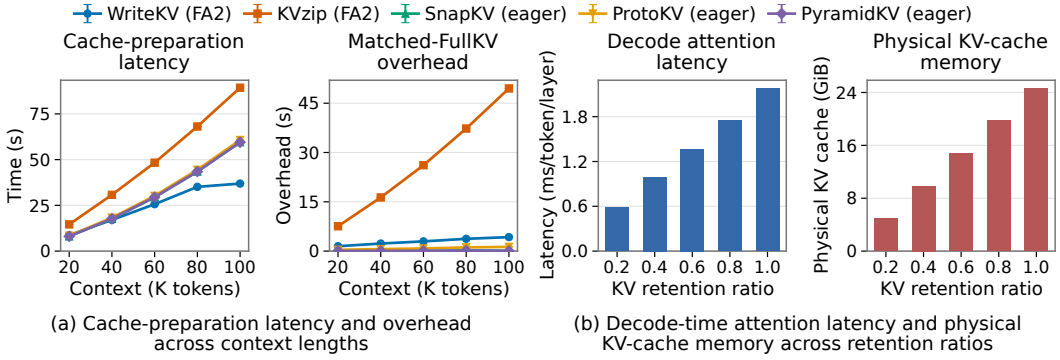



Figure 9: Cache preparation and physical compaction efficiency. (a) Mixed-backend cache preparation time and additional preparation overhead relative to the matched Full-KV implementation; error bars show the sample standard deviation. (b) Packed-attention latency and physical cache occupancy at different KV retention rates in the 800K microbenchmark.

the quality–resource trade-off of block size and select the default setting, and then evaluate cache preparation overhead as well as the decoding and storage benefits of physical compaction.

**Block-wise trade-offs.** Figure 8(a) shows that the quality curves for $B = 2048$, $4096$, $8192$, and No-block are nearly identical on LoCoMo. Figures 8(b)–(c) further report a LongMemEval-100K microbenchmark at a retention rate of 0.5. Increasing the block size from 2048 to 8192 reduces cache preparation time from 38.25 s to 36.35 s, but increases peak memory during preparation from 21.48 GiB to 23.13 GiB. We therefore use $B = 4096$ as the default because it provides a balanced trade-off among quality, time, and memory. The reported peak memory is the preparation working set, rather than the size of the physically compacted cache.

**Cache preparation overhead.** Figure 9(a) reports total cache preparation time from prefill to a decode-ready cache at a retention rate of 0.5, together with the additional overhead relative to matched Full-KV. WRITEKV and KVzip are compatible with FlashAttention 2 (FA2), whereas the other compression baselines use the eager backend. Total time reflects end-to-end implementation efficiency; additional preparation overhead is the time difference from matched Full-KV under the same backend. At 100K, WRITEKV adds 4.26 s and takes 36.91 s in total. SnapKV adds only 0.11 s, but its eager path takes 59.55 s overall, showing the practical benefit of WRITEKV's FA2 compatibility. KVzip takes 89.33 s in total, with 49.49 s of additional overhead.

**Physical compaction and runtime benefits.** Figure 9(b) reports decoding and storage efficiency after cache preparation. With FlashAttention, reducing the retention rate from 1.0 to 0.2 lowers the per-token, per-layer packed-attention latency of the eight full-attention layers from 2.179 ms to 0.583 ms, a $3.74\times$ speedup, and reduces physical cache occupancy by 79.9%. Because 800K exceeds the model's native context length, this experiment serves only as a system microbenchmark. The latency measurement excludes linear-attention layers, projections, and the generation loop. Here, $r = 1.0$ denotes packed full retention rather than native Full-KV. In addition, we measure the one-time cache

preparation cost on the same 800K prefix. WRITEKV ($B = 4096$) takes 775.50 s, compared with 744.23 s for Full-KV, an increase of 4.20%. This cost can be amortized across queries.

## 5 RELATED WORK

**Query-dependent KV-cache eviction.** Most existing methods estimate token importance from attention induced by the current query. H2O (Zhang et al., 2023) and Scissorhands (Liu et al., 2023) dynamically update the cache using decode-time attention, whereas SnapKV (Li et al., 2024) and PyramidKV (Cai et al., 2024) select tokens from attention observed in an end-of-prefill window, with PyramidKV additionally allocating layerwise budgets. CriticalKV(Feng et al., 2025), ReST-KV(An et al., 2026), and LookaheadKV(Ahn et al., 2026) further refine these signals through output perturbation, spatiotemporal smoothing, or future-importance prediction. ProtoKV(Yu et al., 2026) groups tokens according to key-space structure and selects relevant semantic clusters using observation-window query attention. Although these methods provide task-relevant retention signals, their caches are generally tied to a particular query or generation trajectory and cannot be directly reused across different downstream queries.

**Query-agnostic KV-cache compression.** To enable cache reuse, KVzip (Kim et al., 2025) reconstructs the original context to obtain importance scores, Fast KVzip (Kim et al., 2026) distills this signal into a lightweight gating network, and TRIM-KV (Bui et al., 2026) predicts an intrinsic retention score at token creation time. Other methods rely on static K/V geometry: KeyDiff (Park et al., 2025) and ManifoldKV (Datta et al., 2026) use key differences or outlierness, CurDKV(Sengupta et al., 2025) approximates leverage scores from random-projection row norms and combines key- and value-side scores, and CapKV(Yang et al., 2026) uses a weighted value-side leverage-like score to characterize information capacity. The main CapKV(Yang et al., 2026) variant incorporates a prior derived from historical queries, while a query-agnostic variant removes this prior. In contrast, WRITEKV requires neither context reconstruction nor an independently trained scoring model and does not directly rely on static K/V geometry. WRITEKV extracts history-conditioned residual directions from native writes in the linear-attention layers and applies block-wise Ridge novelty scoring to construct a reusable compressed cache.

**Cache representations and memory management in hybrid attention.** Orthogonal to token eviction, GQA (Ainslie et al., 2023), MLA (Liu et al., 2024a), and Palu (Chang et al., 2024) reduce KV storage through cross-head sharing, latent representations, and low-rank projections, respectively, while KIVI (Liu et al., 2024b) applies low-bit quantization. DuoAttention (Xiao et al., 2025) assigns different attention heads either the full history or a local window. Hybrid architectures such as Jamba (Lieber et al., 2024), Gated DeltaNet (Yang et al., 2025), and Kimi Linear (Team et al., 2025) combine full attention with fixed-size recurrent states. WRITEKV leaves the architecture and recurrent state unchanged, instead using existing memory-state write dynamics to guide full-attention KV retention. It is therefore complementary to quantization and representation-compression techniques.

## 6 CONCLUSION

We presented WRITEKV, a query-agnostic KV-cache eviction method for hybrid-attention models. WRITEKV uses history-conditioned native writes from linear-attention layers as retention signals for the full-attention KV cache, applies block-wise Ridge novelty scoring, and performs unified-budget layer-token selection to construct a reusable physical cache without access to future queries, context reconstruction, or independently trained scorers. Across LoCoMo, RealTalk, and LongMemEval, WRITEKV outperforms all evaluated KV-cache eviction baselines under a 20% KV budget. On an 800K-token prefix at this budget, it reduces decode-time attention latency to 26.8% of the full-KV attention latency while increasing cache-construction time by only 4.20% relative to full-KV prefill. These results show that native writes provide an effective, low-overhead, and reusable signal for query-agnostic KV retention.

## AI USE STATEMENT

Generative AI tools were used to assist with drafting and revising portions of this manuscript, including translation, language polishing, and improvements to organization and readability. They were also used to support literature retrieval and discovery. The authors independently reviewed and edited all AI-assisted text, verified suggested references and citations against the original sources, and checked the technical content and experimental claims. Generative AI tools were not used for research ideation or execution, generating synthetic datasets, or proving mathematical claims. All AI-assisted content was reviewed and approved by the authors.

## ETHICS STATEMENT

This work presents an inference-time KV-cache eviction method for hybrid-attention language models. We evaluate it on the publicly available LoCoMo, RealTalk, and LongMemEval benchmarks, following their respective data-use conditions and licenses. Some benchmarks contain human conversation data; we do not attempt to identify individuals or redistribute raw conversation content. No new personal data are collected in this study. The authors follow the ICLR Code of Ethics.

## REPRODUCIBILITY STATEMENT

The WRITEKV and its implementation details are described in Section 3 and Appendix A. The datasets, models, baselines, evaluation protocol, and main experimental settings are specified in Section 4.1, with additional results in Appendix B. An anonymized implementation and configuration files will be released upon acceptance.

## REFERENCES

Jinwoo Ahn, Ingyu Seong, Akhil Kedia, Junhan Kim, Hyemi Jang, Kangwook Lee, and Yongkweon Jeon. Lookaheadkv: Fast and accurate kv cache eviction by glimpsing into the future without generation. *arXiv preprint arXiv:2603.10899*, 2026.

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*, pp. 4895–4901, 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. *arXiv preprint arXiv:2605.08840*, 2026.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. 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.

Ngoc Bui, Shubham Sharma, Simran Lamba, Saumitra Mishra, and Rex Ying. Cache what lasts: Token retention for memory-bounded kv cache in llms. *International Conference on Learning Representations*, 2026:120316–120342, 2026.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, et al. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2024.

Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Chong-Yan Chen, Yu-Fang Hu, Pei-Shuo Wang, Ning-Chi Huang, Luis Ceze, Mohamed S Abdelfattah, and Kai-Chiang Wu. Palu: Compressing kv-cache with low-rank projection. *arXiv preprint arXiv:2407.21118*, 2024.

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

Tri Dao, Daniel Y Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022.

Debajyoti Datta, Trishala Neeraj, Bibek Paudel, Vyom Sharma, and Subhabrata Mukherjee. Manifoldkv: Training-free kv cache compression via euclidean outlier detection. *arXiv preprint arXiv:2602.08343*, 2026.

Yuan Feng, Junlin Lv, Haoyu Guo, Yukun Cao, S Kevin Zhou, and Xike Xie. Criticalkv: Optimizing kv cache eviction from an output perturbation perspective. *arXiv preprint arXiv:2502.03805*, 2025.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pp. 5156–5165. PMLR, 2020.

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. In *Advances in Neural Information Processing Systems*, volume 38, pp. 167563–167591, 2025.

Jang-Hyun Kim, Dongyoon Han, and Sangdoo Yun. Fast kvzip: Efficient and accurate llm inference with gated kv eviction. *arXiv preprint arXiv:2601.17668*, 2026.

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*, pp. 611–626, 2023.

Dong-Ho Lee, Adyasha Maharana, Jay Pujara, Xiang Ren, and Francesco Barbieri. Realtalk: A 21-day real-world dataset for long-term conversation. *arXiv preprint arXiv:2502.13270*, 2025.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

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. *Advances in Neural Information Processing Systems*, 37:22947–22970, 2024.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, et al. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*, 2024.

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024a.

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. *Advances in Neural Information Processing Systems*, 36:52342–52364, 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. *arXiv preprint arXiv:2402.02750*, 2024b.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13851–13870, 2024.

William Merrill, Yanhong Li, Tyler Romero, Anej Svete, Caia Costello, Pradeep Dasigi, Dirk Groeneveld, David Heineman, Bailey Kuehl, Nathan Lambert, Chuan Li, Kyle Lo, Saumya Malik, DJ Matusz, Benjamin Minixhofer, Jacob Morrison, Luca Soldaini, Finbarr Timbers, Pete Walsh, Noah A. Smith, Hannaneh Hajishirzi, and Ashish Sabharwal. Olmo hybrid: From theory to practice and back, 2026. URL https://arxiv.org/abs/2604.03444.

Piotr Nawrot, Adrian Łańcucki, Marcin Chochowski, David Tarjan, and Edoardo M Ponti. Dynamic memory compression: Retrofitting llms for accelerated inference. *arXiv preprint arXiv:2403.09636*, 2024.

Matanel Oren, Michael Hassid, Nir Yarden, Yossi Adi, and Roy Schwartz. Transformers are multi-state rnns. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 18724–18741, 2024.

Junyoung Park, Dalton Jones, Matthew J. Morse, Raghavv Goel, Mingu Lee, and Christopher Lott. KeyDiff: Key similarity-based KV cache eviction for long-context LLM inference in resource-constrained environments. In *Advances in Neural Information Processing Systems*, volume 38, pp. 6771–6807, 2025. doi: 10.52202/085713-0210.

Qwen Team. Qwen3-Next: Towards ultimate training and inference efficiency, September 2025. URL https://qwen.ai/blog?id=4074cca80393150c248e508aa62983f9cb7d27cd.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Ayan Sengupta, Siddhant Chaudhary, and Tanmoy Chakraborty. Value-guided KV compression for LLMs via approximated CUR decomposition. In *Advances in Neural Information Processing Systems*, volume 38, pp. 88198–88229, 2025. doi: 10.52202/085713-2656.

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

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.

Hanlin Tang, Yang Lin, Jing Lin, Qingsen Han, Shikuan Hong, Yiwu Yao, Gongyi Wang, and Danning Ke. Razorattention: Efficient kv cache compression through retrieval heads. *International Conference on Learning Representations*, 2025:16632–16646, 2025.

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

Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, et al. Kimi linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025.

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

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. *arXiv preprint arXiv:2410.10813*, 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. *International Conference on Learning Representations*, 2025:37228–37253, 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*, volume 2025, pp. 56691–56709, 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.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *International Conference on Learning Representations*, volume 2025, pp. 29687–29707, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Zhiyuan Yu, Shijian Xiao, Zhangyue Yin, Xiaoran Liu, Lekai Xing, Wenzhong Li, Cam-Tu Nguyen, and Sanglu Lu. Protokv: Long-context knowledges are already well-organized before your query. In *International Conference on Learning Representations*, volume 2026, pp. 142982–143009, 2026.

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

# A Block-Wise Ridge Scoring: Adaptive Regularization and Numerical Details

This appendix provides implementation details for the block-wise Ridge scoring introduced in Section 3.2. It focuses on the accumulation of write directions, the selection of the Ridge regularization coefficient, leverage- score evaluation, and numerical safeguards. Head and observer aggregation, global budget selection, and physical KV packing are outside its scope.

## A.1 Valid Write Directions

For head $h$ in linear layer $\ell$, the implementation reads the value-side native write $\mathbf{v}^{\text{new}}_{\ell,h,t}$ directly from the production path and uses it as the write vector:

$$\mathbf{w}_{\ell,h,t} := \mathbf{v}^{\text{new}}_{\ell,h,t}, \qquad \mathbf{u}_{\ell,h,t} = \frac{\mathbf{w}_{\ell,h,t}}{\|\mathbf{w}_{\ell,h,t}\|_2}. \tag{5}$$

Normalizing the write vector retains only its direction and does not directly incorporate its magnitude into the subsequent Ridge score. Positions with a zero norm, a non-finite write, or a non-finite direction after normalization do not participate in either second-moment accumulation or leverage-score evaluation. The set of write directions in the main text implicitly contains only these valid positions.

## A.2 Block-Causal Accumulation

We partition the context in its original order into blocks $\mathcal{B}_1, \ldots, \mathcal{B}_m$. For block $b$ and linear head $(\ell, h)$, let $\mathcal{V}^{(b)}_{\ell,h}$ denote the valid write positions in that block. The implementation accumulates a second-moment matrix over the preceding blocks and the complete current block:

$$\mathbf{G}^{(b)}_{\ell,h} = \mathbf{G}^{(b-1)}_{\ell,h} + \sum_{j \in \mathcal{V}^{(b)}_{\ell,h}} \mathbf{u}_{\ell,h,j}\mathbf{u}^{\top}_{\ell,h,j}, \qquad \mathbf{G}^{(0)}_{\ell,h} = \mathbf{0}. \tag{6}$$

Thus, the statistics for block $b$ cover $\mathcal{B}_1, \ldots, \mathcal{B}_b$ but exclude future blocks $\mathcal{B}_{b+1}, \ldots, \mathcal{B}_m$. All candidate tokens in the same block share the updated $\mathbf{G}^{(b)}_{\ell,h}$. The current implementation does not apply an additional token-level causal mask within a block; it therefore uses block-level causal scoring rather than per-token online scoring.

The direction buffer for the current block is released after that block has been scored. Historical information is no longer stored as individual token directions and is instead represented by the accumulated second-moment matrix $\mathbf{G}^{(b)}_{\ell,h}$.

## A.3 Adaptive Spectral Regularization

Before computing the Ridge scores for the current block, we symmetrize the accumulated second-moment matrix:

$$\widetilde{\mathbf{G}}^{(b)}_{\ell,h} = \frac{1}{2}\left(\mathbf{G}^{(b)}_{\ell,h} + \mathbf{G}^{(b)\top}_{\ell,h}\right). \tag{7}$$

Let its eigenvalues in descending order be $\mu_1 \geq \cdots \geq \mu_d \geq 0$, with total spectral energy

$$E = \sum_{i=1}^{d} \mu_i. \tag{8}$$

Using the energy threshold $\tau = 0.90$, the implementation selects the smallest spectral rank that explains at least 90% of the cumulative spectral energy:

$$r = \min\left\{ k : \frac{\sum_{i=1}^{k} \mu_i}{E} \geq \tau \right\}. \tag{9}$$

Here, $r$ is used only to determine the Ridge regularization strength. It does not denote a token count, token retention ratio, or final compression rate. The subsequent leverage score still uses the complete accumulated matrix and the full direction space.

Let the spectral-tail energy outside the leading $r$ directions be

$$E_{\text{tail}} = \sum_{i=r+1}^{d} \mu_i. \tag{10}$$

The current implementation uses the spectral-tail energy normalized by the retained spectral rank and combines it with an FP64 numerical floor to determine the regularization coefficient for block $b$ and head $h$:

$$\lambda_{\ell,h}^{(b)} = \max\left( \frac{E_{\text{tail}}}{r}, \frac{E\,\epsilon_{64}}{d} \right). \tag{11}$$

The first term is the total spectral-tail energy divided by the retained rank; it is not the arithmetic mean of the tail eigenvalues, and the implementation does not use the tail dimension as the denominator. The second term is an FP64 numerical floor that depends on the total spectral energy and feature dimension, preventing the regularization coefficient from becoming exactly zero when the tail energy vanishes.

This adaptive rule changes $\lambda$ with the spectrum of each block and head. Regularization is weaker when the spectral-tail energy is small and stronger when it is large. The coefficient both stabilizes the solve involving $\mathbf{G} + \lambda\mathbf{I}$ and controls the amplification of directions with low spectral coverage in the leverage score.

### A.4 Full-Spectrum Leverage Evaluation

Given $\mathbf{G}_{\ell,h}^{(b)}$ and $\lambda_{\ell,h}^{(b)}$, the Ridge score of a valid candidate position $t$ in the current block is

$$s_{\ell,h,t} = \mathbf{u}_{\ell,h,t}^{\top} \left( \mathbf{G}_{\ell,h}^{(b)} + \lambda_{\ell,h}^{(b)} \mathbf{I} \right)^{-1} \mathbf{u}_{\ell,h,t}. \tag{12}$$

This computation uses the complete accumulated second-moment matrix rather than truncating it to the leading $r$ eigenvectors. The spectral rank affects only $\lambda$ and does not change the direction space used for leverage-score evaluation.

The implementation does not explicitly construct a matrix inverse. Let each row of $\mathbf{U}_{\ell,h}^{(b)}$ be the transpose of one candidate write direction, and define

$$\mathbf{L}\mathbf{L}^{\top} = \mathbf{G}_{\ell,h}^{(b)} + \lambda_{\ell,h}^{(b)} \mathbf{I}, \qquad \mathbf{Y} = \mathbf{L}^{-1} \mathbf{U}_{\ell,h}^{(b)\top}. \tag{13}$$

The score for each candidate position is the squared $L_2$ norm of the corresponding column of $\mathbf{Y}$. This Cholesky-whitening evaluation is mathematically equivalent to the inverse-based definition. Tiled or batched GPU execution can change the order of floating-point operations, however, so the result is not required to be bitwise identical to a scalar per-head reference implementation.

For numerical robustness, we symmetrize the accumulated matrix and clamp small negative eigenvalues caused by rounding, require finite positive spectral energy, restrict $r$ to $[1, d]$, and validate both the Cholesky solve and the final scores. A block–head pair with fewer than two valid writes is reported and excluded from directional statistics. Directions are stored in GPU FP16, the linear solve uses FP32, and the eigendecomposition uses CPU FP64 (`eigvalsh`). The main experiments use $\tau = 0.90$ and $B = 4096$.

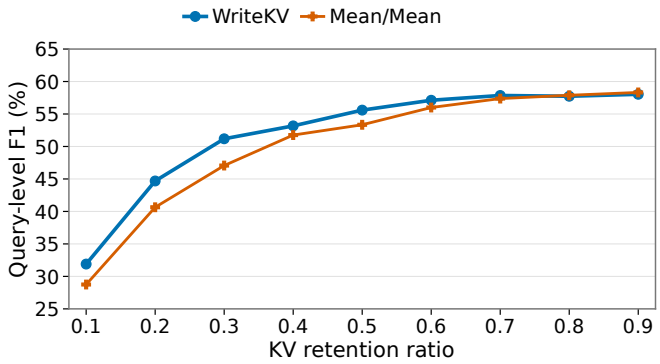



Figure 10: Aggregation ablation with Qwen3.5-9B on LoCoMo. Mean/Mean replaces the two maxima in Equation 3 with arithmetic means.

# B Additional Experimental Results

This section supplements the aggregation ablation in Section 4.3 and the model-scaling and context-length results in Section 4.2. The experiments use the main quality- evaluation settings specified in Section 4.1: BF16 precision, batch size 1, greedy decoding, a maximum generation length of 96 tokens, chunked prefill with a chunk size of 1024 tokens, and a scoring block length of 4096 for WriteKV. We treat KVzip, SnapKV, ProtoKV, and PyramidKV as compression baselines when identifying the strongest baseline. The measurements at retention rates $r = 0.1$ through $0.9$ are independent runs. For visualization, the compression curves connect to the corresponding Full-KV score at $r = 1.0$; these endpoints are not separate runs of the compression methods and are therefore excluded from claims about when a method reaches Full-KV.

## B.1 Ablation of the Aggregation Strategy

The default aggregation in Equation 3 takes the maximum first across the heads of each associated linear layer and then across the observer linear layers assigned to each full-attention layer. Mean/Mean replaces both maxima with arithmetic means while leaving the rest of the pipeline unchanged.

Figure 10 shows that Max/Max is consistently better from $r = 0.1$ to $0.7$, including a 3.15-point gain at $r = 0.1$, while the two variants converge at higher retention rates. This result supports using max aggregation to preserve sparse but strong novelty evidence under tight budgets.

## B.2 Additional Model-Scaling Results

Figure 11 extends the model-scaling evaluation to RealTalk and LoCoMo. At $r = 0.1$, WriteKVobtains F1 scores of 15.05% and 25.50% on RealTalk with Qwen3.5-4B and Qwen3.5-27B, respectively, and 24.21% and 36.26% on LoCoMo. These scores exceed the strongest compression baseline by 2.80, 10.31, 16.15, and 17.00 percentage points, respectively. The gain from scaling WriteKVfrom 4B to 27B at this retention rate is 10.45 points on RealTalk and 12.05 points on LoCoMo.

As the retention budget increases, the differences among compression methods narrow. At $r = 0.5$, WriteKVis 0.87 points above the strongest compression baseline on RealTalk with Qwen3.5-27B and within 1.66 points of it on the other three model–dataset pairs. At $r = 0.9$, its score is within 0.63 points of the strongest compression baseline in every panel. Relative to Full-KV, WriteKVis 0.86 and 0.07 points lower on RealTalk with the 4B and 27B models, respectively, and 0.05 points lower on LoCoMo with the 4B model. On LoCoMo with Qwen3.5-27B, WriteKVfirst exceeds Full-KV at $r = 0.8$ and remains 0.04 points above it at $r = 0.9$. The other three panels do not reach Full-KV at any independently measured retention rate.

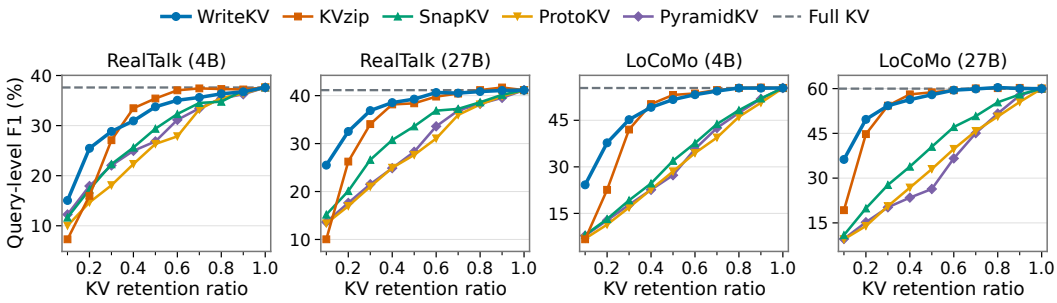



Figure 11: Additional model-scaling results on RealTalk and LoCoMo. From left to right, the panels show RealTalk with Qwen3.5-4B and Qwen3.5-27B, followed by LoCoMo with Qwen3.5-4B and Qwen3.5-27B. Dashed lines indicate the corresponding Full-KV scores. The points at $r = 1.0$ connect each compression curve to Full-KV for visualization and are not independent measurements.

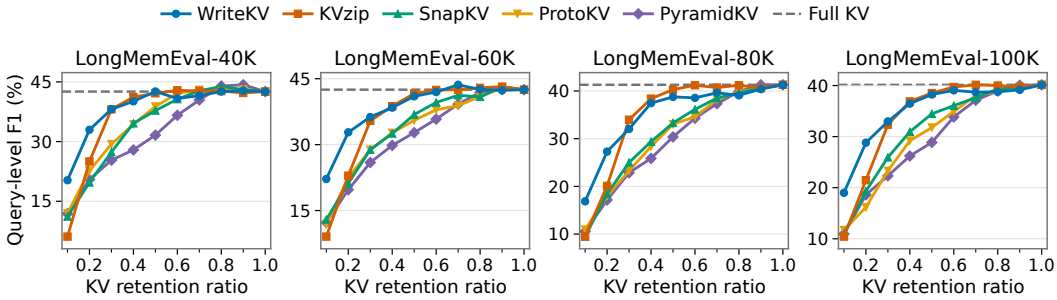



Figure 12: Context-length robustness of Qwen3.5-4B on LongMemEval-40K, LongMemEval-60K, LongMemEval-80K, and LongMemEval-100K. Dashed lines indicate the corresponding Full-KV scores. The points at $r = 1.0$ connect each compression curve to Full-KV for visualization and are not independent measurements.

### B.3 Context-Length Robustness with Qwen3.5-4B

Figure 12 reports Qwen3.5-4B results on the 40K, 60K, 80K, and 100K LongMemEval settings. At $r = 0.1$, WriteKVachieves F1 scores of 20.28%, 22.18%, 16.86%, and 19.00%, respectively. It outperforms the strongest compression baseline at every length by 8.31, 9.24, 5.93, and 7.29 percentage points. The 5.93–9.24 point range shows that the tight-budget advantage persists from 40K through 100K contexts.

The advantage is concentrated at low retention rates. At $r = 0.5$, WriteKVis 0.46 points above the strongest compression baseline at 40K, while trailing it by 0.29–1.52 points at 60K–100K. At $r = 0.9$, WriteKVobtains F1 scores of 42.74%, 42.34%, 40.40%, and 39.15% from 40K to 100K. These results are 0.18 points above, 0.16 points below, 0.90 points below, and 0.99 points below the corresponding Full-KV scores. WriteKVfirst reaches Full-KV at $r = 0.5$ for 40K and at $r = 0.7$ for 60K. It does not reach Full-KV through the highest independently measured retention rate for 80K or 100K.