LIVEMEM: MAINTAINING LLM LONG-RUNNING MEMORY BY LIVING MEMORY STATE

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-running assistants and agents stand or fall by memory; yet prevailing memory systems keep memory outside the model: they store the interaction history as textual records and retrieve them back into the context on demand. This paradigm preserves access to history but leaves inference itself *stateless*; and the preserved texts are also limited to the information that can be verbalized. This paper studies an alternative approach to long-running memory: a memory state that lives inside the computation, which is updated online, is persistent across context turnovers, and continues to influence the model's behavior after its source evidence has left the context. We introduce LiveMem, which augments a pretrained full-attention LLM with a parallel recurrent memory branch: the branch maintains a living memory state across the lifecycle, while the original attention path provides exact access to a bounded working context. LiveMem updates and preserves this state as the active context turns over, and uses memory-oriented post-training to teach the model how to use the state. Experiments show that LiveMem achieves leading overall performance compared with RAG and other intrinsic memory methods. On LongMemEval, LiveMem answers questions from its memory state even when the supporting evidence has been fully removed from the context, and its test-time learning results indicate that the state carries patterns distilled from past experience rather than only retrievable records. Therefore, LiveMem establishes a performant implementation of state-based long-running memory as a practical complement to existing retrieval-based memory systems.

1 INTRODUCTION

Language models are evolving from isolated question-answering systems into continuously active assistants and agents over long-term interactions (OpenClaw Project, 2026; Nous Research, 2026; Park et al., 2023; Wang et al., 2024a). As such systems run longer, their usefulness increasingly hinges on *memory*: the ability to retain what a long interaction history contains and to exploit it later. Since the history inevitably outgrows any bounded context, a growing line of work equips models with external memory—textual records that are extracted, organized, and retrieved by an engineered system and read back into the context when needed (Packer et al., 2024; Chhikara et al., 2025; Xu et al., 2025; Li et al., 2025; Latimer et al., 2025). Memory has thus become a research direction of its own, but its dominant form treats memory as *data*: explicit records that live outside the model and re-enter its computation through the context window.

We argue that this view captures only one dimension of what memory must be for a continuously running model. First, external memory leaves inference itself *stateless*: the context is not merely a transcript of the past but the model's working state, and when it is replaced, an external system can restore records but not the computation that had been carried forward. Second, text can only carry what can be verbalized: much of what long-running interaction leaves behind—for instance, user habits, accumulated calibration, regularities distilled from past examples—is expressed as changed behavior rather than as retrievable facts. This mirrors the division of human memory into declarative systems, which store what can be stated, and non-declarative systems, which hold skills and tendencies that are enacted rather than described (Squire, 2004). In that sense, external memory captures the declarative dimension; the non-declarative dimension is seldom implemented in today's LLM systems.

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.

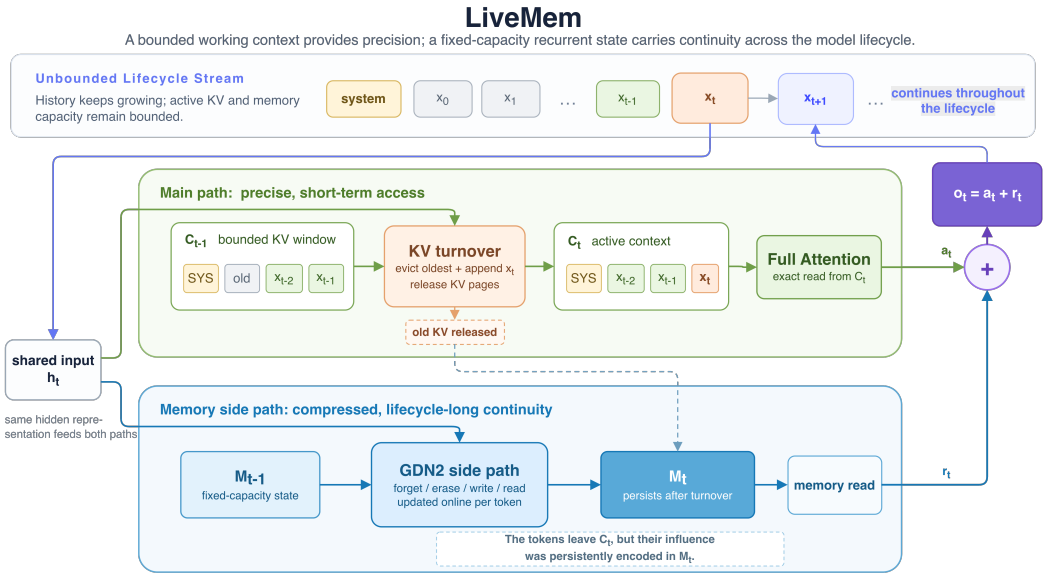


Figure 1: LiveMem enables stateful continual inference under context turnover. A bounded working context provides precise short-term access, while a fixed-capacity recurrent memory state preserves information across the model lifecycle.

We therefore ask a different question: *Can a persistent latent state be maintained, where memory resides?* We study memory realized as a recurrent state that accompanies inference over the entire lifecycle. We define *state continuity* as the ability of such a state to be updated online, persist across bounded-context turnovers, and affect model predictions after the source evidence is no longer available to attention.

Recent work has begun to explore memory at this scale (Section 2), but a regular recurrent state does not by itself constitute memory. If a recurrent state replaces attention entirely, the model loses the precise short-term access that attention enables; and if a memory branch is trained while the full history remains visible to attention, the model can learn to bypass the state instead of relying on it (Yang et al., 2026). A living memory state must be made load-bearing: the model has to experience context turnover, and should learn how to write, preserve, and read the state via relevant training tasks.

We introduce **LiveMem**, a pretrained full-attention LLM augmented with a parallel recurrent memory branch that maintains a living memory state. The branch, built on Gated DeltaNet-2 (GDN2) (Hatamizadeh et al., 2026), reads and updates a fixed-size memory state at every decoder layer, while the original attention path provides exact access to a bounded working context. Context turnover releases evicted KV chunks from the attention path once the context reaches its limit, and the information they carried persists in the memory state. A memory-oriented post-training scheme—supervised fine-tuning followed by reinforcement learning on tasks whose evidence is deliberately phased out of the window—teaches the model to genuinely rely on the state.

Our experiments show that this state-based memory is both real and useful. LiveMem achieves the leading overall performance across four memory task suites. It is uniquely robust when many questions share the same packed history, a regime in which competing methods often fail. On LongMemEval, LiveMem remains accurate when the supporting evidence has been completely evicted from the context, where stateless baselines drop sharply. On test-time learning tasks, the memory state carries patterns distilled from streams of past text–label pairs—a form of memory that textual records cannot easily express. At the same time, the state is lossy rather than an exact archive, which positions LiveMem as complementary to, rather than a replacement for, retrieval-based memory. In summary, our contributions are:

- We formulate *state continuity* as the missing state dimension of long-running memory, beyond the two limits of textual memory: stateless inference and the verbalization bottleneck.

- We present LiveMem, which makes a living memory state load-bearing on a pretrained attention LLM through a parallel GDN2 branch, consistent context turnover, and memory-oriented post-training.
- We show through comprehensive evaluations and eviction-conditioned analyses that the state genuinely carries evicted history, and we characterize its boundaries.

2 Related Work

Textual memory. Retrieval-augmented generation conditions a parametric generator on passages from an external corpus (Lewis et al., 2020), and related systems retrieve model-side representations such as cached key–value pairs or memory banks (Wu et al., 2022; Wang et al., 2023; Xiao et al., 2024a). Agent frameworks maintain editable textual records over time: MemGPT tiers information between the context and external storage (Packer et al., 2024), Mem0 extracts and consolidates salient facts from interactions (Chhikara et al., 2025), A-Mem links structured memory notes into an evolving network (Xu et al., 2025), and MemOS manages memory as a first-class system resource (Li et al., 2025). These systems decide *which records* the model reads; they provide declarative memory outside the computation. LiveMem is orthogonal to them and, as our analysis suggests, complementary.

Intrinsic memory and recurrent architectures. Past information can also live inside the model: internalized in parameters through history-specific LoRA adapters (Back et al., 2026; Hu et al., 2022) or differentiable MLP memory (Wei et al., 2026), or held in latent states that self-update from text (Wang et al., 2024b; 2025; Das et al., 2024). This line of work builds on a long lineage of recurrent computation: segment recurrence in Transformer-XL and RMT (Dai et al., 2019; Bulatov et al., 2022), subquadratic hybrids such as Mamba, Infini-attention, and Jamba (Gu & Dao, 2024; Munkhdalai et al., 2024; Lieber et al., 2024), and linear attention interpreted as fast-weight programming, which motivates delta-rule updates (Schlag et al., 2021; Yang et al., 2024) with gated variants (Yang et al., 2025b; Kimi Team, 2025) and GDN2's separated erase and write controls (Hatamizadeh et al., 2026). Test-Time Training, Titans, and Nested Learning further treat the recurrent state itself as a learned memory updated at test time (Sun et al., 2025; Behrouz et al., 2025b;a), and δ-Mem augments a frozen backbone with an associative state whose readout corrects attention (Lei et al., 2026). LiveMem builds on this state view but targets the full lifecycle: GDN2 runs as a parallel memory branch in every layer of a pretrained attention model rather than replacing the backbone, and the state is maintained online through context turnover and trained to be load-bearing after its source tokens leave attention.

Bounded attention and context turnover. KV-cache methods bound active attention without carrying a latent state (Xiao et al., 2024b; Zhang et al., 2023; Li et al., 2024). MemAgent maintains a textual memory context that is rewritten after each processing round (Yu et al., 2026), and IndexMem retains query-relevant KV entries exactly while compensating evicted entries with a state (Yang et al., 2026). Query-aware retention assumes that future queries are known when information is committed to memory; LiveMem instead maintains a query-independent memory state that remains functional across context turnover.

3 LiveMem

In this section, we first formalize inference with a memory state under context turnover, and then present the LiveMem design: its architecture, turnover scheduling, and memory-oriented post-training. At a high level, LiveMem adds a recurrent memory path to the attention module of a pretrained decoder transformer, which operates in parallel with the attention computation. These two paths function at different timescales: the attention mechanism provides precise access to the working context, while the recurrent memory path carries memory information across the entire lifecycle.

3.1 Stateful Inference under Context Turnover

Let $H_t = [C_1, C_2, \ldots, C_t]$ denote the whole interaction history at turn t, and $C = [x_1, x_2, \cdots, x_n]$ denote the context, where x_i is the token at position i. An attention-only decoder with fixed parameters can be abstracted as $y_t \sim p_\theta(\cdot \mid C_t)$. The context C_t can thus be viewed as the inference state of the transformer: a collection of key–value pairs that grows with the history and is destroyed once it

exceeds the context limit and a turnover occurs (Kwon et al., 2023; Zhang et al., 2023). Reusing that history for later predictions then relies on mechanisms that retrieve C_t, or some representation derived from it, back into the context (Lewis et al., 2020; Packer et al., 2024). We refer to this paradigm, in which the model reconstructs its working state by reading explicit representations of relevant past information, as *reconstructive inference*.

As a top level abstraction, let M_t be a memory state after the model has processed the input up to time t, maintained by a learned transition function U_ϕ with parameters ϕ introduced by the memory module:

$$\begin{aligned} M_{t+1} &= U_\phi(M_t, C_{t+1}), \\ y_t &\sim p_{\theta,\phi}(\cdot \mid C_t, M_t). \end{aligned} \tag{1}$$

The working context C_t provides fine-grained access to recent information, while M_t carries compressed historical information of fixed size and is not required to reconstruct every token exactly. We refer to this setup as *continual inference with an accompanying memory state*, where inference is viewed as an ever-evolving state transition process that persists throughout the lifecycle. The state is complementary to external retrieval: a standalone retrieval system can only recall based on the model's understanding of the current query, whereas an accompanying memory state gives the model a standing "awareness" of the past.

The mere presence of a recurrent module does not prove that it carries memory. We reserve the term *living memory state* for a recurrent state that meets the requirements of state continuity stated in the introduction: a model truly carries memory only when it can rely on the state to make decisions after its critical evidence has been completely removed from the context.

3.2 Recurrent Memory State and Side Attention

The memory module of LiveMem is attached as a side branch to each attention layer (Figure 4 in Appendix A gives an overview). It reads and maintains the memory state via Gated DeltaNet-2 (GDN2) (Hatamizadeh et al., 2026), whose output is directly added back to the original attention output. Let h_t^ℓ be the normalized hidden representation at layer ℓ, and S_t^ℓ be the corresponding memory state, then the block computes as follows:

$$\begin{aligned} a_t^\ell &= \mathrm{Attn}_\ell(h_{\le t}^\ell; C_t), \\ (r_t^\ell, S_t^\ell) &= \mathrm{GDN2}_\ell(h_t^\ell, S_{t-1}^\ell), \\ o_t^\ell &= a_t^\ell + r_t^\ell, \end{aligned} \tag{2}$$

after which the o_t^ℓ returns to the original model. In this way, the historical memory information is injected into the model through the side branch of the attention module.

For one recurrent head, let $S_t \in \mathbb{R}^{d_k \times d_v}$ denote its state (omitting layer and head indices). GDN2 produces a log-decay coefficient $\exp(g_t)$, in which $g_t \le 0$, an erase gate $b_t \in [0,1]^{d_k}$, and a write gate $w_t \in [0,1]^{d_v}$. The complete update process is as follows:

$$\overline{S}_t = \mathrm{Diag}(\exp(g_t))\, S_{t-1}, \qquad \text{forget,} \tag{3}$$

$$\rho_t = w_t \odot v_t - \overline{S}_t^\top (b_t \odot k_t), \qquad \text{erase and prepare,} \tag{4}$$

$$S_t = \overline{S}_t + k_t \rho_t^\top, \qquad r_t = S_t^\top q_t, \qquad \text{write and read.} \tag{5}$$

The decay operation selectively forgets the current memory state along the key dimension, and the erase operation selectively removes old information to reduce the interference when writing new information; these behaviors are learned during training. The read and update operations are performed per token to ensure real-time memory utilization and updating, which makes sure that when context turnover occurs, no additional write operations are performed.

The q, k and v parameters of the side branch are initialized from those of the attention backbone, while other parameters are newly initialized. The output projection is initialized with zeros, so that the introduction of the entire side branch is initially equivalent to the original model itself. During training, the model gradually learns how much information from the memory branch should contribute to its predictions. See Appendix A.1 for details.

3.3 LIVE MEMORY UNDER CONTEXT TURNOVER

LiveMem processes the input as an ordered sequence of chunks $X_0, X_1, \ldots X_N$. We retain the system prompt, which serves as a structural attention sink (Xiao et al., 2024b) and semantically preserves the most important system instructions, ensuring that they are always visible to the model. All other chunks processed through a bounded attention window, which is implemented with a first-in-first-out queue. When adding a new chunk would exceed either the chunk-count or token-count limit, we evict the oldest chunks until both limits are satisfied. This behavior can be expressed by the following formula:

$$\text{visible}(i, j) = \mathbf{1}[i \leq j]\ \mathbf{1}\big[\text{in_mem}(\text{chunk}(i), j)\big]. \tag{6}$$

where j is the position of the current query token, i denotes the position of a historic token, chunk(i) represents the chunk containing token i, and the function in_mem(chunk(i),j) indicates whether chunk i is within the window when the query is at position j. This behavior ensures that the activation boundary changes only at chunk boundaries, and guarantees that no out-of-bounds situations would occur. Equation (6) applies only to the main attention path, whereas eqs. (3) to (5) apply to the entire sequence. When an old chunk leaves C_t, its KV caches become invisible in the main attention, but the information it contains has already been preserved in the memory state.

We apply the context turnover behavior during both training and inference to ensure training-inference consistency. During training, we implement it via dynamic attention masks. At inference time, we directly release the evicted KV pages to make them inaccessible. Detailed explanations of the infrastructure are provided in Appendix A

3.4 MEMORY-ORIENTED POST-TRAINING

Adding recurrent states does not guarantee that the model will actually utilize it to carry memory. Therefore, we design memory-oriented post-training tasks. In these tasks, each example consists of a segment of historical memory and one or more corresponding queries. We leverage the context turnover mechanism described in Section 3.3 to ensure that most of the history is removed from attention during training, thereby forcing the model to learn how to utilize the information in the memory state to produce answers, and also how to better preserve information into the memory state. We train the full memory branch while keeping the backbone parameters frozen. We design two training paradigms: supervised fine-tuning (SFT) and reinforcement learning (RL).

Answer-based Supervised Learning. In SFT, each example concatenates the history, queries, and reference answers into a single causal stream. The loss is applied only to the answer positions $\mathcal{A}$:

$$\mathcal{L}_{\text{SFT}} = -\frac{1}{|\mathcal{A}|} \sum_{t \in \mathcal{A}} \log p_{\theta,\phi}(x_t \mid C_t, S_t). \tag{7}$$

We first use long-document question-answering data to establish stable memory-utilization behavior, and help activate the memory side branch. We then train on a broader mixture of data including QA, classification, and multiple-query tasks. Through SFT, the model learns how to use the side branch to carry memory information.

Exploration with Memory. In RL, we optimize the memory branch with GRPO (Shao et al., 2024), incorporating practices from DAPO (Yu et al., 2025): clip-higher, token-level loss, dynamic sampling, and removal of KL; the full objective is deferred to Appendix A. Each prompt contains a memory segment with one or more questions, and we sample $G = 8$ responses per prompt. The reward of a response is its per-question accuracy plus an all-correct bonus and a format penalty:

$$R_i = a_i + 0.5 \cdot \mathbf{1}[k_i = n_i] - 0.5 \cdot \mathbf{1}[F_i = 0], \tag{8}$$

where $a_i = k_i/n_i$ is the ratio of questions answered correctly, and $F_i \in \{0, 1\}$ indicates whether it contains formatting failures. We first compute accuracy using exact match, and for negative cases, we fall back to an LM judge for equivalence judgment to keep the reward unbiased. Through this process, the model learns to better utilize the side branch for carrying memory.

4 EXPERIMENTS

Our experiments primarily address three research questions:

1. How does LiveMem perform compared to other textual memory and intrinsic memory approaches? (Section 4.2)
2. To what extent can LiveMem retain memories outside the window, and how well can these memories be maintained as the lifecycle evolves? (Section 4.3)
3. How can one effectively train a LiveMem module? (Section 4.4)

4.1 Experimental Setup

Tasks and metrics. We evaluated our approach on Wiki QA, Conversation QA, Test-Time Learning (TTL), and Long Document QA (Long QA). For the Wiki QA suite, we used 2WikiMultiHopQA, HotpotQA, and MuSiQue as evaluation sets (Ho et al., 2020; Yang et al., 2018; Trivedi et al., 2022), and additionally packed multiple samples together to form a multi-question (MQ) task, for which RAG's top-k retrieval cannot cover the joint evidence required by multiple questions at once and is therefore omitted. For the Conversation suite, we employed LoCoMo, LongMemEval, and the FactConsolidation dialogue data from MemoryAgentBench (MAB) for evaluation (Maharana et al., 2024; Wu et al., 2025; Hu et al., 2026). For the TTL suite, we used the six TTL subsets from MAB, namely Banking77, CLINC150, NLU, TREC-coarse/fine, and ReDial (Casanueva et al., 2020; Larson et al., 2019; Liu et al., 2021; Li & Roth, 2002; Li et al., 2018). For Long QA, we adopted the English QA task from ∞Bench (Zhang et al., 2024), EventQA from MAB, and NarrativeQA (Kočiský et al., 2018). We report per-question accuracy, using LLM-Judge (Zheng et al., 2023) to determine whether open-ended answers are consistent with the reference answers, using exact match to judge whether the answers to classification tasks are correct, and using recall@5 for ReDial. Detailed introductions to the evaluation tasks can be found in appendix B.

Evaluation protocol. All systems are compared using Qwen3-4B-Instruct-2507[1] (hereinafter referred to as Qwen3-4B) (Yang et al., 2025a). To fully test the performance of each approach under limited context, we restrict the context window to 32k tokens; for shorter datasets (Wiki QA suite, LoCoMo), we further restrict it to 8k tokens. This means that if the compared method does not have a corresponding memory mechanism, the part of the input memory document that exceeds the window limit will be truncated from the head. For all generation, we uniformly adopt the sampling parameters officially recommended by Qwen: temperature 0.7, top-p 0.8, and top-k 20[1].

Systems. We use Qwen3-4B as the direct baseline for comparison, while δ-Mem (Lei et al., 2026) adds a side-path state to it. We also compare against Context2LoRA—our shorthand for the history-specific LoRA memory setting of Back et al. (2026)—for which we construct the LoRA parameter memory module following its official implementation. RAG uses Qwen3-Embedding-0.6B (Zhang et al., 2025) as the retrieval model and retrieves relevant evidence via vector recall for generation. In addition, we design a Recurrent baseline, which behaviorally references MemAgent (Yu et al., 2026): it maintains a memory context, updates the memories within it upon each context turnover, outputs the entire memory after each round of processing, and feeds it into the prompt in the next round of memory processing. For LiveMem, we compare the models obtained from the two training steps described in Section 3.4, namely LiveMem-SFT and LiveMem-RL. More implementation details can be found in Appendix B.

4.2 Main Results

Table 1 reports the experimental results under the limited context window setting. On the Wiki QA suite, the MQ tasks—where several questions share one packed history—separate the methods most clearly: the baselines drop dramatically from their single-question results, while LiveMem maintains a clear lead (e.g., 0.587 vs. 0.336 for the strongest baseline on 2Wiki-MQ). This indicates that existing baselines are good at handling only a single multi-hop logic chain, but struggle when faced with multiple logic chains simultaneously. As a result, LiveMem outperforms the other methods overall on Wiki QA.

We also observe that RAG and Context2LoRA achieve SOTA results on Conversation and TTL suites, respectively. However, they are too specialized on their suites. They show little performance on tasks other than their respective SOTA ones. On Long QA, although RAG and Recurrent each excel on

[1] https://huggingface.co/Qwen/Qwen3-4B-Instruct-2507

Table 1: Results under bounded-context configurations. Best result per row in bold, second best underlined.

	Qwen3-4B	RAG	Recurrent	Context2LoRA	δ-Mem	LiveMem-SFT (ours)	LiveMem-RL (ours)
Wiki QA							
2Wiki	<u>0.817</u>	0.117	0.775	0.213	**0.824**	0.768	0.784
2Wiki-MQ	0.336	-	0.204	0.009	0.232	<u>0.551</u>	**0.587**
Hotpot	<u>0.846</u>	0.399	0.819	0.234	**0.853**	0.755	0.772
Hotpot-MQ	0.329	-	0.256	0.008	0.226	<u>0.477</u>	**0.487**
MuSiQue	<u>0.461</u>	0.117	**0.483**	0.095	**0.483**	0.369	0.383
MuSiQue-MQ	0.200	-	0.165	0.003	0.151	<u>0.327</u>	**0.347**
All	0.498	0.211	0.450	0.094	0.462	<u>0.541</u>	**0.560**
Conversation							
LoCoMo	0.255	**0.405**	0.175	0.140	0.256	0.295	<u>0.314</u>
LME-S	0.224	**0.548**	<u>0.386</u>	0.168	0.230	0.286	0.302
FactCons	**0.177**	0.090	0.100	0.081	0.171	<u>0.175</u>	<u>0.175</u>
All	0.219	**0.348**	0.220	0.130	0.219	0.252	<u>0.264</u>
TTL							
Banking	0.770	0.740	0.000	0.860	0.770	<u>0.920</u>	**0.940**
CLINC	0.870	0.660	0.000	0.830	0.890	**0.930**	<u>0.920</u>
NLU	0.760	0.600	0.010	**0.860**	0.770	<u>0.820</u>	<u>0.820</u>
TREC-C	0.730	0.720	0.340	**0.940**	0.740	<u>0.780</u>	0.770
TREC-F	0.460	0.290	0.220	**0.860**	<u>0.510</u>	0.480	0.470
Redial	0.141	0.069	0.081	0.081	<u>0.146</u>	0.131	**0.149**
All	0.622	0.513	0.109	**0.739**	0.638	0.677	<u>0.678</u>
Long QA							
∞Bench-QA	0.117	0.228	**0.316**	0.083	0.114	0.234	<u>0.251</u>
EventQA	0.445	**0.612**	0.447	0.269	0.433	0.463	<u>0.506</u>
NarrQA	0.313	0.233	0.282	0.150	0.326	<u>0.332</u>	**0.371**
All	0.282	<u>0.358</u>	0.348	0.167	0.291	0.343	**0.376**
Overall							
Overall	0.458	0.389	0.281	0.327	0.451	<u>0.505</u>	**0.519**

particular tasks, LiveMem maintains an overall lead in this group by achieving balanced and strong scores across all three tasks.

In summary, LiveMem is a well-rounded method that performs consistently across all tasks. The result shows that LiveMem achieves SOTA results on Wiki QA and Long QA, and while it does not rank first on Conversation and TTL, its performance is second only to the SOTA baselines on these tasks. Meanwhile, other baselines each have their own strengths and weaknesses. Once they encounter tasks they are not good at, their performance almost completely collapses. Overall, LiveMem substantially outperforms the other baselines, improving the macro-averaged accuracy from 0.458 (the strongest baseline, Qwen3-4B) to 0.519 (LiveMem-RL). The same trend holds under a 256K context cap (Appendix C).

Table 2: The results of LiveMem performance w. and w.o. historical information stored in memory state. "State" processes the complete history and stores them in memory state, whereas "Trunc." sees only the truncated suffix and begins from zero state. Δ is State minus Trunc.

	LiveMem-SFT			LiveMem-RL		
Dataset	Trunc.	State	Δ	Trunc.	State	Δ
2Wiki-MQ	0.512	0.551	+0.039	0.545	0.587	+0.042
LoCoMo	0.277	0.295	+0.019	0.299	0.314	+0.015
LME-S	0.296	0.286	-0.010	0.294	0.302	+0.008
Banking	0.900	0.920	+0.020	0.930	0.940	+0.010
∞Bench-QA	0.228	0.234	+0.006	0.222	0.251	+0.028
NarrQA	0.332	0.332	+0.000	0.357	0.371	+0.014

4.3 Does the Recurrent State Carry Evicted Evidence?

We compare the performance of LiveMem under two settings: one with historical memory incorporated, and the other with historical information directly truncated. For models that include historical memory in their memory state, they can refer to the history to a certain extent when answering

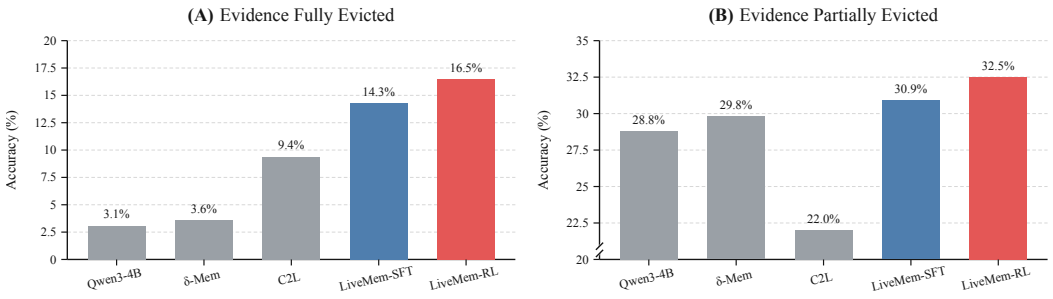


Figure 2: LongMemEval-s accuracy grouped by whether the supporting evidence is fully or partially displaced from the active context. This analysis tests whether the memory state truly benefits performance when the evidence leaves the working context.

questions; for the truncation setting, without historical information, the model can only rely on the information within the current window to answer questions. Table 2 reports representative experimental results. It can be observed that for LiveMem-SFT, except for a performance drop on LongMemEval, incorporating historical information leads to improvements on all other datasets. For LiveMem-RL, on the reported datasets, performance consistently improves after incorporating historical information. This may be attributed to the fact that the model after RL is better at maintaining and leveraging historical memories.

We further filter the results on LongMemEval and select instances where all key evidence lies outside the window, as well as those where key evidence is partially outside the window, and report the accuracy on these subsets. Figure 2 presents the accuracy comparison results. It can be seen that, regardless of whether the evidence is entirely or partially removed, leveraging historical information from LiveMem's memory state can achieve higher accuracy than other methods. Notably, under the full-eviction setting, while Qwen3-4B and δ-Mem, which have no access to historical information, achieve extremely low accuracy, LiveMem outperforms them by more than ten percentage points. This demonstrates the necessity of the memory state in carrying historical memories.

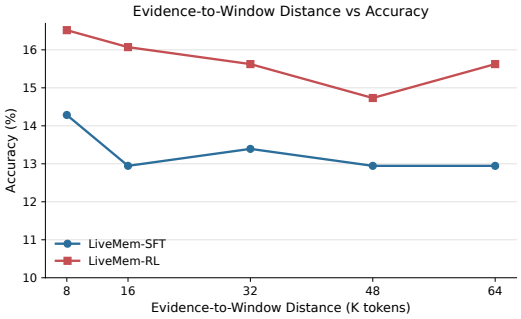


Figure 3: LongMemEval-S accuracy as supporting evidence is placed progressively farther from the active-context window. Accuracy remains measurable after multiple turnover intervals rather than collapsing at the eviction boundary.

We also manually control the distance from evidence to the current window in the LongMemEval subset of Figure 2(A) to investigate whether the retention duration of evidence in the memory state affects LiveMem's utilization of memories. Figure 3 shows the results as the evicted evidence moves progressively farther away from the active window. The model's accuracy exhibits a slight downward trend as the distance increases, but overall remains at a stable level. This indicates that the memory update and forgetting mechanism based on GDN2 does affect the usability of key evidence in the memory state to some extent, yet it is still capable of keeping such evidence retainable within the memory state.

4.4 Architecture and Training Ablations

Trainable capacity. We compare the performance of LiveMem under different training capacity settings, including side-path LoRA, side-path full-parameter, and side-path full-parameter + main-path LoRA. We present the comparison results of the SFT checkpoints, which are shown in Table 3 (a). The results indicate that using full-parameter training for the side-path yields more stable performance, likely because the distribution gap between Attention and GDN2 is relatively large, and LoRA—which optimizes within a small subspace—struggles to adapt to the behavior of GDN2. Adding

Table 3: SFT ablations under bounded active context. Panel (a) varies which parameters are trainable; Panel (b) varies the proportion and order of long-context data.

	2Wiki-MQ	LoCoMo	LME-S	Banking	TREC-F	∞Bench-QA	EventQA
(a) Trainable capacity							
LoRA side	0.479	**0.256**	**0.362**	0.880	0.440	0.214	**0.490**
Full side	**0.551**	0.234	0.286	**0.920**	0.480	0.234	0.463
Full side + main-path LoRA	0.520	0.220	0.280	0.910	**0.540**	**0.236**	0.412
(b) Training curriculum							
Direct mixture	0.439	0.261	0.298	0.740	0.420	**0.222**	0.448
Rebalanced mixture	0.452	**0.263**	0.326	0.770	0.260	0.208	0.412
Long warmup + rebalanced	**0.479**	0.256	**0.362**	**0.880**	**0.440**	0.214	**0.490**

LoRA to the main path does not bring significant improvements; we suspect that the main-path LoRA training may absorb some of the memory that should have been directed to the side-path.

Data Mixture and Training curriculum. We also compare the effects of data mixing ratios and curriculum training in Table 3 (b). Specifically, we compare three settings: (1) using all data directly, (2) using rebalanced data , where we removed the majority of the long-text QA data, which had originally constituted a large proportion of the overall dataset, and (3) using long-text QA data for cold-start to open the side-path, followed by formal training with the rebalanced data. The results show that simply mixing all data together does not perform well, because the long-text data imposes low recall requirements, leaving the model with little incentive to learn how to maintain the memory state. After we reduced the long-text data, the model's performance on recall tasks improved, but its performance on Long QA declined. We attribute that the rebalanced data is too small to sufficiently activate the side-path, which prevents the main path from fully leveraging the information in memory state. Therefore, we first use long-text data for warm-up to activate the side-path, and then use the rebalanced data to train the model to maintain the memory state. This curriculum yields the greatest overall performance improvement.

5 Conclusion

In this paper, we have argued that memory for long-running LLM systems has been understood mainly as *data*—textual records kept outside the model—while inference itself remains stateless and text can only carry what can be verbalized. We formulated the complementary requirement of *state continuity*: a memory state that lives inside the computation, persists across context turnover, and remains load-bearing after its source evidence has left the context.

LiveMem realizes this state axis on a pretrained full-attention LLM: a parallel GDN2 memory branch in every decoder layer maintains a fixed-size living memory state, while the attention path retains a bounded working context. LiveMem updates and preserves the state as the active context turns over, and memory-oriented post-training teaches the model to genuinely rely on it, so that the state still carries information after its originating tokens are released—without pretraining a recurrent backbone from scratch.

Empirically, LiveMem achieves the leading overall performance against textual, parametric, and intrinsic memory baselines (0.519 vs. 0.458 for the strongest baseline), with a clear advantage when multiple questions share a packed history. Eviction-conditioned analyses show that the state remains behaviorally effective after the supporting evidence has fully left the context, and test-time learning results show that it carries distilled patterns of experience rather than only retrievable records. The state is lossy rather than an exact archive (Appendix D), which delimits its role.

Retrieval and state continuity are complementary axes of long-running memory: retrieval provides exact, addressable access to what can be written down, while a living memory state maintains the computation itself—including what cannot be verbalized—across the lifecycle. We see stateful inference as a practical step toward LLM systems whose memory persists as long as they run.

AI USE STATEMENT

We used AI tools (Claude Opus 4.8 / Fable 5 with Claude Code) to assist in code implementation. Additionally, we used AI tools (GPT5.6-Sol with Codex, and Kimi-K3 with Claude Code) to assist with writing. All AI-assisted content was reviewed and verified by the authors: the experimental code was run and tested for correctness by the authors, and all AI-assisted text was manually checked. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of AI.

REPRODUCIBILITY STATEMENT

We provide the details needed to reproduce our results: the memory-branch architecture and state parameterization (Section 3.2, Appendix A.1), the context-turnover policy used consistently in training and serving (Section 3.3, Appendix A), the full post-training recipe including data mixtures, objectives, and hyperparameters (Section 3.4, Appendix A), and the evaluation protocol, dataset construction, and baseline implementations (Section 4.1, Appendix B). Experiment code and data are submitted as supplementary material.

REFERENCES

Seungju Back, Dongwoo Lee, Naun Kang, Taehee Lee, S. K. Hong, Youngjune Gwon, and Sungjin Ahn. Understanding LoRA as knowledge memory: An empirical analysis. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, Seoul, South Korea, 2026. URL https://arxiv.org/abs/2603.01097.

Yushi Bai, Xin Lv, Jiajie Zhang, Yuze He, Ji Qi, Lei Hou, Jie Tang, Yuxiao Dong, and Juanzi Li. LongAlign: A recipe for long context alignment of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 1376–1395, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.74. URL https://aclanthology.org/2024.findings-emnlp.74/.

Ali Behrouz, Meisam Razaviyayn, Peilin Zhong, and Vahab Mirrokni. Nested learning: The illusion of deep learning architectures. In *Advances in Neural Information Processing Systems*, 2025a. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/4309616aaed8e848009bc4a7ef73b493-Abstract-Conference.html.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. In *Advances in Neural Information Processing Systems*, 2025b. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/a4ca07aa108036f80cbb5b82285fd4b1-Abstract-Conference.html.

Aydar Bulatov, Yuri Kuratov, and Mikhail S. Burtsev. Recurrent memory transformer. In *Advances in Neural Information Processing Systems*, volume 35, pp. 11079–11091, 2022. URL https://arxiv.org/abs/2207.06881.

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. Efficient intent detection with dual sentence encoders. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pp. 38–45, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.nlp4convai-1.5. URL https://aclanthology.org/2020.nlp4convai-1.5/.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. LongLoRA: Efficient fine-tuning of long-context large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=6PmJoRfdaK. Introduces the LongAlpaca long-context instruction-following dataset.

Zhi Chen, Qiguang Chen, Libo Qin, Qipeng Guo, Haijun Lv, Yicheng Zou, Hang Yan, Kai Chen, and Dahua Lin. What are the essential factors in crafting effective long context multi-hop instruction datasets? insights and best practices. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27129–27151, Vienna, Austria, July

2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.1316. URL https://aclanthology.org/2025.acl-long.1316/.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. In *ECAI 2025 – 28th European Conference on Artificial Intelligence*, volume 413 of *Frontiers in Artificial Intelligence and Applications*, pp. 2993–3000. IOS Press, 2025. doi: 10.3233/FAIA251160. URL https://doi.org/10.3233/FAIA251160.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc 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. URL https://aclanthology.org/P19-1285/.

Payel Das, Subhajit Chaudhury, Elliot Nelson, Igor Melnyk, Sarathkrishna Swaminathan, Sihui Dai, et al. Larimar: Large language models with episodic memory control. In *Proceedings of the 41st International Conference on Machine Learning*, 2024. URL https://proceedings.mlr.press/v235/das24a.html.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021. URL https://aclanthology.org/2021.naacl-main.365/.

Juechu Dong, Boyuan Feng, Driss Guessous, Yanbo Liang, and Horace He. Flex attention: A programming model for generating optimized attention kernels. *arXiv preprint arXiv:2412.05496*, 2024. URL https://arxiv.org/abs/2412.05496.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024. URL https://arxiv.org/abs/2312.00752.

Ali Hatamizadeh, Yejin Choi, and Jan Kautz. Gated DeltaNet-2: Decoupling erase and write in linear attention. *arXiv preprint arXiv:2605.22791*, 2026. URL https://arxiv.org/abs/2605.22791.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, 2020. URL https://aclanthology.org/2020.coling-main.580/.

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. URL https://arxiv.org/abs/2404.06654.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating memory in LLM agents via incremental multi-turn interactions. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=DT7JyQC3MR.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.550. URL https://aclanthology.org/2020.emnlp-main.550/.

Kimi Team. Kimi linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025. URL https://arxiv.org/abs/2510.26692.

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328, 2018. URL https://aclanthology.org/Q18-1023/.

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. URL https://arxiv.org/abs/2309.06180.

Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. An evaluation dataset for intent classification and out-of-scope prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 1311–1316, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1131. URL https://aclanthology.org/D19-1131/.

Chris Latimer, Nicoló Boschi, Andrew Neeser, Chris Bartholomew, Gaurav Srivastava, Xuan Wang, and Naren Ramakrishnan. Hindsight is 20/20: Building agent memory that retains, recalls, and reflects. *arXiv preprint arXiv:2512.12818*, 2025. doi: 10.48550/arXiv.2512.12818. URL https://arxiv.org/abs/2512.12818.

Jingdi Lei, Di Zhang, Junxian Li, Weida Wang, Kaixuan Fan, Xiang Liu, Qihan Liu, Xiaoteng Ma, Baian Chen, and Soujanya Poria. δ-mem: Efficient online memory for large language models. *arXiv preprint arXiv:2605.12357*, 2026. URL https://arxiv.org/abs/2605.12357.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Raymond Li, Samira Ebrahimi Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Chris Pal. Towards deep conversational recommendations. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://papers.nips.cc/paper/8180-towards-deep-conversational-recommendations.

Xin Li and Dan Roth. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002. URL https://aclanthology.org/C02-1150/.

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*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/28ab418242603e0f7323e54185d19bde-Abstract-Conference.html.

Zhiyu Li, Chenyang Xi, Chunyu Li, Ding Chen, Boyu Chen, Shichao Song, Simin Niu, Hanyu Wang, Jiawei Yang, Chen Tang, Qingchen Yu, Jihao Zhao, Yezhaohui Wang, Peng Liu, Zehao Lin, Pengyuan Wang, Jiahao Huo, Tianyi Chen, Kai Chen, Kehang Li, Zhen Tao, Huayi Lai, Hao Wu, Bo Tang, Zhengren Wang, Zhaoxin Fan, Ningyu Zhang, Linfeng Zhang, Junchi Yan, Mingchuan Yang, Tong Xu, Wei Xu, Huajun Chen, Haofen Wang, Hongkang Yang, Wentao Zhang, Zhi-Qin John Xu, Siheng Chen, and Feiyu Xiong. MemOS: A memory OS for AI system. *arXiv preprint arXiv:2507.03724*, 2025. doi: 10.48550/arXiv.2507.03724. URL https://arxiv.org/abs/2507.03724.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, et al. Jamba: A hybrid transformer–mamba language model. *arXiv preprint arXiv:2403.19887*, 2024. URL https://arxiv.org/abs/2403.19887.

Xingkun Liu, Arash Eshghi, Pawel Swietojanski, and Verena Rieser. Benchmarking natural language understanding services for building conversational agents. In *Increasing Naturalness and Flexibility in Spoken Dialogue Interaction: 10th International Workshop on Spoken Dialogue Systems (IWSDS)*, pp. 165–183. Springer, 2021. doi: 10.1007/978-981-15-9323-9_15. URL https://arxiv.org/abs/1903.05566.

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*, 2024. URL https://aclanthology.org/2024.acl-long.747/.

Moonshot AI. Kimi K3: Open frontier intelligence. Technical blog, 2026. URL https://www.kimi.com/blog/kimi-k3.

Tsendsuren Munkhdalai, Manaal Faruqui, and Siddharth Gopal. Leave no context behind: Efficient infinite context transformers with infini-attention. *arXiv preprint arXiv:2404.07143*, 2024. URL https://arxiv.org/abs/2404.07143.

Nous Research. Hermes Agent: The self-improving ai agent. https://github.com/NousResearch/hermes-agent, July 2026. Version 0.19.0 (release tag v2026.7.20), accessed July 27, 2026.

OpenClaw Project. OpenClaw: A personal, open-source ai assistant. https://github.com/openclaw/openclaw, July 2026. Version 2026.7.1, accessed July 27, 2026.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. In *Proceedings of the 1st Conference on Language Modeling (COLM)*, 2024. URL https://arxiv.org/abs/2310.08560.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *ACM Symposium on User Interface Software and Technology (UIST)*, 2023. URL https://arxiv.org/abs/2304.03442.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. ZeRO: Memory optimizations toward training trillion parameter models. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2020. URL https://arxiv.org/abs/1910.02054.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *Proceedings of the 38th International Conference on Machine Learning*, 2021. URL https://proceedings.mlr.press/v139/schlag21a.html.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Larry R. Squire. Memory systems of the brain: A brief history and current perspective. *Neurobiology of Learning and Memory*, 82(3):171–177, 2004. doi: 10.1016/j.nlm.2004.06.005.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. doi: 10.1016/j.neucom.2023.127063. URL https://arxiv.org/abs/2104.09864.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, et al. Learning to (Learn at test time): RNNs with expressive hidden states. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025. URL https://proceedings.mlr.press/v267/sun25h.html.

Together AI. Long-Data-Collections. Hugging Face dataset, 2023. URL https://www.together.ai/blog/llama-2-7b-32k. A collection of long-context pretraining and instruction-finetuning datasets, released July 2023.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022. URL https://aclanthology.org/2022.tacl-1.31/.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhi-Yuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 2024a. URL https://arxiv.org/abs/2308.11432.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. Augmenting language models with long-term memory. In *Advances in Neural Information Processing Systems*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/ebd82705f44793b6f9ade5a669d0f0bf-Abstract-Conference.html.

Yu Wang, Yifan Gao, Xiusi Chen, Haoming Jiang, Shiyang Li, Jingfeng Yang, et al. MEMORYLLM: Towards self-updatable large language models. In *Proceedings of the 41st International Conference on Machine Learning*, 2024b. URL https://proceedings.mlr.press/v235/wang24s.html.

Yu Wang, Dmitry Krotov, Yuanzhe Hu, Yifan Gao, Wangchunshu Zhou, Julian McAuley, Dan Gutfreund, Rogerio Feris, and Zexue He. M+: Extending MemoryLLM with scalable long-term memory. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025. URL https://proceedings.mlr.press/v267/wang25au.html.

Rubin Wei, Jiaqi Cao, Jiarui Wang, Jushi Kai, Qipeng Guo, Bowen Zhou, and Zhouhan Lin. MLP Memory: A retriever-pretrained memory for large language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=1SMdxRtLBp.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=UBvm2bIyxz.

Yuhuai Wu, Markus N. Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=TrjbxzRcnf-.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. InfLLM: Training-free long-context extrapolation for LLMs with an efficient context memory. In *Advances in Neural Information Processing Systems*, 2024a. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, 2024b. URL https://arxiv.org/abs/2309.17453.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-MEM: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/19909c36f51abc4856b4560aff3d36d6-Abstract-Conference.html.

An Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a. URL https://arxiv.org/abs/2505.09388.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In *Advances in Neural Information Processing Systems*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/d13a3eae72366e61dfdc7eea82eeb685-Abstract-Conference.html.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. In *International Conference on Learning Representations*, 2025b. URL https://arxiv.org/abs/2412.06464.

Xintong Yang, Hao Gu, Binxing Xu, Lujun Li, Bei Liu, Jiacheng Liu, Qiyuan Zhu, Yike Guo, and Sirui Han. IndexMem: Learned KV-cache eviction with latent memory for long-context LLM inference. *arXiv preprint arXiv:2605.25475*, 2026. URL https://arxiv.org/abs/2605.25475.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018. URL https://aclanthology.org/D18-1259/.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, et al. MemAgent: Reshaping long-context LLM with multi-conv RL-based memory agent. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2507.02259.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, et al. DAPO: An open-source LLM reinforcement learning system at scale. In *Advances in Neural Information Processing Systems*, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/a4277440d50f1f15d2cb4c14f7e0c0d2-Abstract-Conference.html.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/250cf8b51c773f3f8dc8b4be867a9a02-Abstract.html.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, et al. ∞Bench: Extending long context evaluation beyond 100K tokens. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024. URL https://aclanthology.org/2024.acl-long.814/.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, et al. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/6ceefa7b15572587b78ecfcebb2827f8-Abstract-Conference.html.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, 2023. URL https://arxiv.org/abs/2306.05685.

A Infrastructure and State Lifecycle Details

This appendix details the parameterization, training-time, and serving-time mechanics of LiveMem's memory state; Figure 4 gives an overview of how they connect.

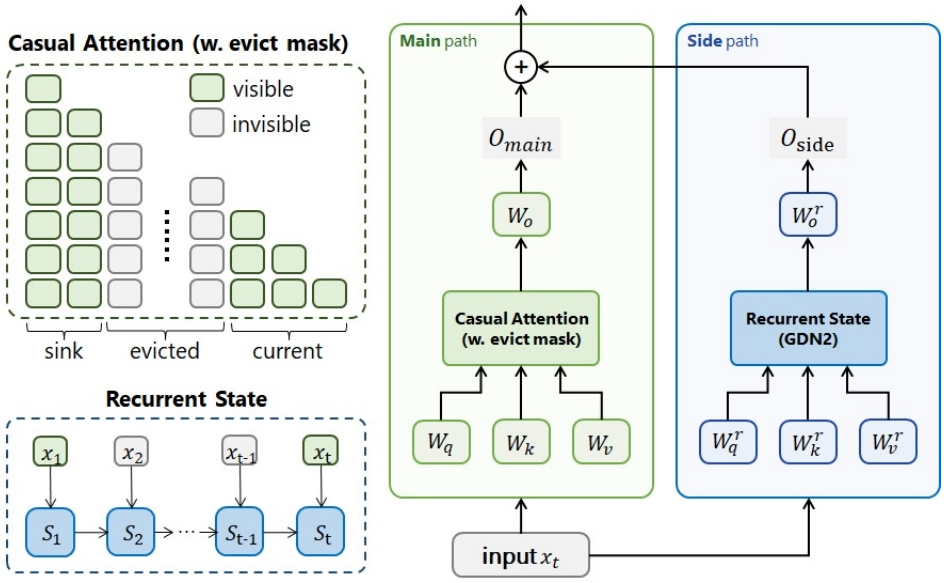


Figure 4: Overview of LiveMem. A full-attention main path maintains a bounded working context, while a recurrent side path updates the recurrent memory state from the same hidden representations. Their outputs are combined at every decoder layer. Context turnover evicts old KV entries from the main path, keeps only the system instruction as attention sink, and the current context in KV cache. The information carried by the evicted context is kept in the recurrent memory state.

A.1 GDN2 Parameterization and State Geometry

The implementation uses a short causal convolution of width four before the GDN2 query, key, and value projections. For completeness, the single-head state update in eqs. (3) to (5) is equivalent to

$$S_t = \left(I - k_t(b_t \odot k_t)^\top\right) \mathrm{Diag}(\exp(g_t)) S_{t-1} + k_t(w_t \odot v_t)^\top. \tag{9}$$

Queries and keys are ℓ_2-normalized inside the recurrent kernel. The readout $S_t^\top q_t$ passes through a gated RMS normalization and an output projection before it is added to the softmax-attention output.

The Qwen3-4B backbone has 36 layers, 32 query heads, 8 KV heads, and head dimension 128. The memory path uses 32 query/key/value heads of dimension 128 in every layer. Backbone KV projections are repeated across the corresponding query-head groups for initialization and can diverge during post-training. Consequently, each layer and request has a recurrent tensor of shape $32 \times 128 \times 128$ in addition to the short-convolution state. The state uses FP32.

The query, key, and value projections of the memory branch are initialized from the matching backbone projections. The decay, erase, write, output-gate, and short-convolution parameters are initialized independently, and the final memory projection is initialized to zero.

A.2 Chunk-Turnover Training with FlexAttention

Given chunk spans $[u_i, v_i)$, the canonical policy appends each non-sink chunk to a live queue and removes the oldest entries while either the chunk count or token budget is exceeded. The newest chunk is never removed at its arrival step. The policy returns an eviction step for every chunk, which is expanded into the per-token $c(i)$ and $e(i)$ fields used by eq. (6). SFT examples use dataset-defined semantic or fixed-token chunks; serving and RL recomputation use deterministic fixed-stride chunks. Both instantiate the same oldest-first state transition.

During SFT, FlexAttention (Dong et al., 2024) combines the turnover predicate with causality and, for packed data, a sequence-identity predicate. The GDN2 chunk kernel receives cumulative sequence boundaries and starts a zero state at each packed-example boundary. Thus a 64K-token packed row can be evaluated in one differentiable forward pass without allowing either attention or recurrent state to cross between constituent examples. Only answer tokens contribute to eq. (7).

RL rollouts use a length-dependent fixed-stride policy. The trainer derives the same per-token chunk and eviction fields from the rollout tokens when it recomputes log probabilities, so the policy update sees the attention visibility used to generate the response. Chunk boundaries are multiples of the 32-token serving page size, which makes token-level mask changes and page release agree exactly.

Let L_p denote the length of the fully templated prompt and let M be the generation allowance in a rule-table row. The rollout server and training recomputation select the first row of Table 4 satisfying $L_p + M$ in the indicated half-open interval. The system-instruction prefix is rounded up to a 32-token page boundary and retained as a sink; all subsequent chunks follow the selected stride, including during generation. For determinism, even a partially filled frontier chunk is charged at its full chunk size when enforcing the live-token budget.

Table 4: Fixed-stride turnover parameters used for RL rollout serving and policy recomputation. All quantities are tokens.

Range of $L_p + M$	Chunk size	Live-token budget	M
$[1\text{K}, 2\text{K})$	64	512	256
$[2\text{K}, 8\text{K})$	256	1,024	512
$[8\text{K}, 16\text{K})$	512	2,048	1,024
$[16\text{K}, 32\text{K})$	512	4,096	2,048
$[32\text{K}, 64\text{K})$	512	8,192	4,096

Benchmark inference uses the same oldest-first transition and page-alignment rules, while overriding the chunk size to 1,024 and the live-token budget to 8K or 32K to realize the fixed evaluation conditions in Section 4.1. Samples outside the RL table are filtered during training; serving clamps an out-of-range request to the nearest row rather than changing the lifecycle semantics.

A.3 Serving

The serving implementation follows paged KV management (Kwon et al., 2023) and maintains two pieces of per-request state. The ordinary attention manager holds KV pages for the instruction sink and recent live chunks. A separate paged slot holds the convolution state and GDN2 matrix state for every memory-augmented layer. At each turnover boundary, complete middle KV pages are returned to the allocator and omitted from the attention read set; the memory slot is unaffected. Prefill slices are clamped to the next chunk boundary so a scheduler batch cannot apply a future turnover state to earlier tokens in the same slice.

For a prefill or decode step, the engine gathers the request's recurrent state, runs the same GDN2 update as training, and scatters the final state back to its slot. Requests without an initial slot read a zero state. When generation finishes, the scheduler removes the request's turnover record and the state manager releases its recurrent slot. This explicit lifecycle implements continuity within a stream while preventing accidental state leakage between requests. Learned decay and erasure in eq. (9) operate within that lifecycle; whole-state reset is a separate systems operation.

A.4 Post-Training Recipe

SFT uses fixed 64K-token packs in two stages. The first stage runs for 100 steps on long-form instruction and document data using LongAlign, LongAlpaca, LongMIT, and Long-Data-Collection (Bai et al., 2024; Chen et al., 2024; 2025; Together AI, 2023), with a 50-step learning-rate warmup followed by a constant rate. The second runs for 500 steps on a mixture of MuSiQue, 2WikiMultihopQA, NarrativeQA, Qasper, long-form instruction data, AG News, and DBpedia (Dasigi et al., 2021; Zhang et al., 2015). We use an answer-token cross-entropy objective, AdamW with weight decay 0.01, gradient clipping at 1.0, and a peak learning rate of 5×10^{-5}. In the second stage, the rate remains constant for 50 steps and then follows a cosine schedule. LoRA dropout is zero.

SFT is run in BF16 on two nodes with eight A800-80GB GPUs each. Each device processes one 64K pack per step, giving a global batch of 16 packs, or approximately one million tokens per optimizer step. We use gradient checkpointing, distributed data parallelism, and ZeRO-1 (Rajbhandari et al., 2020) optimizer-state sharding. The GDN2 recurrent state remains FP32.

The RL stage optimizes the following objective. Given a prompt q containing memory, we sample $G = 8$ responses o_i and compute their advantages with group normalization:

$$\widehat{A}_i = \frac{R_i - \mu_R}{\sigma_R}, \qquad \mu_R = \frac{1}{G}\sum_{j=1}^{G} R_j, \quad \sigma_R = \mathrm{Std}_j(R_j). \tag{10}$$

For response token $o_{i,t}$, let $\rho_{i,t} = \pi_\theta(o_{i,t} \mid q, o_{i,<t}) / \pi_{\theta_{\mathrm{old}}}(o_{i,t} \mid q, o_{i,<t})$ and $\bar{\rho}_{i,t} = \mathrm{clip}(\rho_{i,t}, 1 - \epsilon_{\mathrm{low}}, 1 + \epsilon_{\mathrm{high}})$. The maximization objective is

$$J_{\mathrm{RL}}(\theta) = \mathbb{E}\left[\frac{1}{\sum_i T_i}\sum_{i=1}^{G}\sum_{t=1}^{T_i}\Phi(\rho_{i,t}, \widehat{A}_i)\right], \tag{11}$$

$$\Phi(\rho, A) = \begin{cases} \min(\rho A, \bar{\rho} A), & A \geq 0, \\ \max(\gamma A, \min(\rho A, \bar{\rho} A)), & A < 0, \end{cases}$$

where $\epsilon_{\mathrm{low}} = 0.20$, $\epsilon_{\mathrm{high}} = 0.28$, and $\gamma = 3$. Token-level normalization uses the number of tokens in the generated responses, and dynamic sampling retains only groups with $\mathrm{Std}_i(a_i) > 0$.

Rollouts use groups of eight responses and the reward described in Section 3.4, with an actor learning rate of 10^{-5}. Training uses FSDP2 actors, asynchronous vLLM rollouts, two eight-GPU trainer nodes, one eight-GPU rollout node, and a separate two-GPU tensor-parallel judge service. Per-request generation limits are coupled to the turnover bucket so generated tokens cannot silently exceed the context policy used for their prompt.

B Additional Experimental Details

B.1 Evaluation Dataset Construction

2WikiMultiHopQA, HotpotQA, and MuSiQue are multi-hop knowledge-based question-answering datasets, whose document stores are taken from the 2018-12 Wikipedia passage corpus preprocessed by Karpukhin et al. (2020). These datasets evaluate the performance of memory mechanisms in multi-hop knowledge QA scenarios. Each instance in HotpotQA and 2WikiMultiHopQA consists of 10 documents, which include both relevant and irrelevant documents, with the evidence required to answer the question distributed across the relevant documents. MuSiQue is similar, except that it consists of 20 documents per instance. For the MQ setting, each instance is formed by packing together 5 randomly sampled instances from the corresponding dataset (10 for 2WikiMultiHopQA), meaning that the model must answer all questions simultaneously based solely on what it has retained from the packed data. For each of 2WikiMultiHopQA, HotpotQA, and MuSiQue, we evaluate a fixed seed-0 sample of 2,000 single-question instances and 400 multi-question instances. For the MQ setting, all questions are answered at once after the memory is processed.

LoCoMo, LongMemEval, and FactConsolidation are all dialogue-based QA datasets that evaluate the capability of memory mechanisms in realistic dialogue scenarios. We use all 500 questions in LongMemEval-S, all questions associated with the ten released LoCoMo conversations, and all questions in MAB FactConsolidation.

The TTL suite is based on existing text-label pairs and requires labeling the input text-label pairs. It demands that the memory mechanism generalize patterns and distill experience from past text-label pairs, rather than simply memorizing the text verbatim. We directly use the MAB's TTL episodes, and follow its evaluation settings.

The LongQA suite focuses on completing question-answering tasks over long input texts, where the information required to answer the questions is scattered throughout the lengthy text. It requires the memory mechanism to possess the ability to gather such distributed information and answer questions when faced with extremely long texts. NarrativeQA is sampled by document: we select 120 test documents with seed 0 and evaluate every retained question for those documents.

Open-ended QA and conversation answers are evaluated by a Qwen3.6-35B-A3B[2] judge prompted with the question, reference answers, and prediction and asked for a binary correctness decision. The judge uses greedy decoding with thinking disabled. EventQA and the five TTL classification sources use normalized exact match. ReDial uses recall@5 after resolving the released entity identifiers to movie names.

B.2 Context Accounting and Generation

Every method receives the same system instruction, task template, and answer format. When an input exceeds a direct-context cap, the instruction and query are retained and the memory portion is truncated from the oldest end. The 256K condition additionally reserves the task-specific generation allowance and 1,024 tokens for chat-template overhead before allocating the remaining budget to memory. The bounded direct-context controls expose the corresponding recent 8K or 32K suffix; LiveMem instead scans up to 256K input tokens while maintaining only the stated live KV budget.

Except for separately identified official-protocol checks, decoding uses temperature 0.7, top-p 0.8, and top-k 20. Single-question requests permit up to 1,024 generated tokens and multi-question requests up to 8,192. Prompts that contain multiple questions are never shortened by dropping questions; the memory prefix is the only truncation target.

B.3 Baseline Implementation Details

For the evaluation of all approaches, we construct inputs in the format of [system prompt, user prompt (memory documents, question), assistant answer]. For Qwen3-4B, when the context exceeds the window limit, we retain the system prompt and perform head truncation only on the memory documents. For δ-Mem, we adopt its TSW setting and strictly follow the behavior of its official prompt protocol, which directly discards the portion of the input that exceeds the context window limit. For Context2LoRA, for each input memory instance, it pre-constructs multiple QA pairs and then uses these QA pairs to train a LoRA module (rank=4, alpha=8, lr=5×10^{-4}). During evaluation, it leverages the memories stored in the LoRA parameters to answer questions. We apply a separate LoRA module for each dataset. For TTL, since it inherently consists of text-label pairs and is not suitable for constructing QA pairs, we choose to follow Context2LoRA's original assumption about memory, and directly fine-tune LoRA using these text-label pairs. This may be the reason why Context2LoRA performs particularly well on TTL. For the RAG method, we use Qwen3-Embedding-0.6B as the vector model and build a vector retrieval environment over the memory documents for retrieval and recall. For datasets with fewer than 20 memory documents, we use top-1; for others, we use top-3. If the input does not have a strict partitioning into memory documents (e.g., Long QA), we split it into chunks of 512 tokens each. Since RAG's top-k fundamentally cannot meet the evidence requirements of the MQ task, we omit RAG's performance on MQ. For Recurrent, we draw on MemAgent's setting of maintaining a memory context through recurrent cycles. We set the upper limit of the memory context to 8k tokens, so the maximum length of memory text it can process at one time is 16k tokens. The model is required to extract important information from the memory documents and, after processing is complete, answer questions based on the memory context.

B.4 Training Dataset

The datasets we used for training are listed in Table 5.

C Reference Results: Under 256K Context Limit

We additionally evaluate a context-limited protocol in which up to 256K tokens remain materialized in the attention context and no pages are released during the request. Inputs that exceed the model budget lose the oldest memory tokens while preserving the recent suffix, query, and generation reserve. This protocol mirrors the operational behavior of the released δ-Mem code, which fills a configured context budget and keeps the recent tail on overflow.

[2] https://huggingface.co/Qwen/Qwen3.6-35B-A3B

Table 5: Overview of the datasets used for training.

Dataset Name	Categories
Qasper	Question Answering (Ret.)
NarrativeQA	Question Answering (Ret.)
MuSiQue	Multi-hop Question Answering (Ret.)
2WikiMultihopQA	Multi-hop Question Answering (Ret.)
AG News	Text Classification (TTL)
DBpedia	Text Classification (TTL)
LongAlign	Long-Context Instruction Tuning (Long)
LongAlpaca	Long-Context Instruction Tuning (Long)
Long-Data-Collections	Long-Context Instruction Tuning (Long)
LongMIT	Long-Context Instruction Tuning (Long)

Table 6: Performance with a 256K materialized-context cap. Overflow is tail-preserving: the oldest memory tokens are removed before inference. This is a context-limited reference rather than a state-continuity test.

	Qwen3-4B	RAG	Recurrent	Context2LoRA	δ-Mem	LiveMem-SFT	LiveMem-RL
Wiki QA							
2Wiki	0.683	0.117	<u>0.775</u>	0.218	0.637	0.771	**0.787**
2Wiki-MQ	0.511	-	0.204	0.009	0.187	<u>0.625</u>	**0.653**
Hotpot	0.632	0.399	**0.819**	0.237	0.590	0.756	<u>0.772</u>
Hotpot-MQ	0.361	-	0.256	0.025	0.144	<u>0.706</u>	**0.718**
MuSiQue	<u>0.400</u>	0.117	**0.483**	0.096	0.365	0.362	0.379
MuSiQue-MQ	0.305	-	0.165	0.002	0.201	<u>0.381</u>	**0.417**
Conversation							
LoCoMo	0.575	0.405	0.175	0.210	<u>0.585</u>	0.554	**0.599**
LME-S	0.384	**0.548**	0.386	0.156	0.360	0.422	<u>0.438</u>
FactCons	**0.175**	0.090	0.100	0.096	0.165	0.168	<u>0.170</u>
TTL							
Banking	0.830	0.740	0.000	0.910	0.860	**0.950**	<u>0.940</u>
CLINC	0.860	0.660	0.000	<u>0.920</u>	0.850	**0.950**	**0.950**
NLU	0.800	0.600	0.010	<u>0.820</u>	0.790	<u>0.820</u>	**0.830**
TREC-C	0.680	0.720	0.340	**0.940**	0.690	<u>0.800</u>	0.660
TREC-F	0.460	0.290	0.220	**0.870**	0.460	<u>0.570</u>	0.510
Redial	<u>0.169</u>	0.069	0.081	0.142	**0.172**	0.093	<u>0.169</u>
Long QA							
∞Bench-QA	0.291	0.228	0.316	0.131	0.302	<u>0.407</u>	**0.419**
EventQA	<u>0.624</u>	0.612	0.447	0.252	0.608	0.515	**0.654**
NarrQA	0.410	0.233	0.282	0.181	<u>0.411</u>	0.383	**0.412**

As demonstrated in Table 6, increasing the context limit does not fundamentally alter the overall trend. The conclusions are similar to those in Section 4.2: RAG and Context2LoRA continue to perform well on the tasks to which they are best suited, but remain highly task-specific. The key difference is that LiveMem achieves comprehensive improvements over the baselines on LoCoMo and Long QA tasks, indicating that a larger context limit is particularly beneficial to LiveMem. Overall, LiveMem continues to maintain the leading position with strong and well-rounded performance across tasks.

D LIMITATIONS

D.1 A LOSSY STATE IS NOT AN EXACT ARCHIVE

The task variants in this section follow the RULER long-context evaluation suite (Hsieh et al., 2024).

Table 7: Needle-in-a-haystack accuracy with full and bounded active contexts. The bounded-context result shows that the current LiveMem model does not reliably recover arbitrary fine-grained needles from its recurrent state.

	S-NIAH-2	S-NIAH-3	MK-NIAH-1	MQ-NIAH
		256k		
Qwen3-4B	1.00	1.00	0.97	0.97
δ-Mem	1.00	1.00	0.97	0.97
LiveMem	1.00	0.86	0.99	0.97
		32k		
Qwen3-4B	0.19	0.18	0.21	0.23
δ-Mem	0.19	0.18	0.21	0.23
LiveMem	0.19	0.18	0.21	0.23

The results are shown in table 7. With the 256K materialized context, where the needles remain directly addressable through attention, all systems are near ceiling on most variants; LiveMem adds no systematic benefit. Under the 32K bounded context, all three systems obtain the same low scores. The current LiveMem memory state therefore does not reliably reconstruct arbitrary token-level needles after their KV entries have been released.

This is consistent with the intended role of the memory state, but it remains a real limitation. NIAH asks for lossless archival recall of sparse strings whose later relevance is not predictable from their content. LiveMem instead compresses an indefinitely growing stream into a fixed-size, lossy memory state and trains that state to preserve features useful for subsequent behavior. The evidence-conditioned LongMemEval analyses in figs. 2 and 3 show that this state can carry task-relevant influence across turnover; they do not imply that it stores every past token verbatim.

The distinction from query-aware KV retention is important here. IndexMem's long-prefill indexer can score the prompt after a terminal query is present and keep the query-relevant needle exactly in attention (Yang et al., 2026). That is a natural advantage on NIAH, but it does not require a latent state to carry information whose future relevance was unknown at eviction time. In a continuing stream, no fixed KV budget can retain every token that might become a future needle. Applications requiring exact archival recall should therefore combine LiveMem with retrieval or an external store; these mechanisms provide addressable history access while LiveMem provides state continuity.

D.2 FINITE POSITIONAL HORIZON

LiveMem's recurrent memory state and active KV budget are independent of the number of previously processed tokens, but the current system is not literally unbounded. Its Qwen3 backbone uses rotary position encodings (Su et al., 2024) with a configured maximum position of 262,144 tokens (Yang et al., 2025a). KV eviction preserves the original absolute position indices of retained tokens, so continuing past the trained and configured range requires positional extrapolation whose quality is not guaranteed. The experiments therefore process at most 256K input tokens even though the memory state itself could continue updating.

Position-free attention offers a plausible route around this architectural ceiling. Kimi Linear applies NoPE to its full-attention layers and delegates positional and recency information to recurrent KDA layers, demonstrating the design at 48B total parameters and a 1M-token context (Kimi Team, 2025). Kimi K3 adopts a KDA-based hybrid architecture into a publicly deployed 2.8T-parameter model with a 1M-token context (Moonshot AI, 2026). This provides deployment-scale evidence that the direction is practical. Adapting LiveMem to a NoPE or otherwise length-generalizing backbone remains as future work.