RETURN: REVERSIBLE MEMORY MANAGEMENT FOR LLM AGENTS WITH SUPPORT FOR TOKEN-LEVEL SUPERVISION

Anonymous authors
Paper under double-blind review

ABSTRACT

Summarizing agent histories can discard details needed later, while recalling historical text repeats prefill computation. We introduce RETURN, a framework for reversible context management that keeps active context compact while retaining historical KV states. It restores selected turns with updated positions, without re-prefilling their text. An ephemeral probe selects history from turn digests, then rolls back its tokens and execution state to keep memory management outside the task trace. This enables supervision from a full-context teacher without management labels. Our backend offloads inactive KV to CPU, while path-aware parallel replay reconstructs recorded history selections and position mappings and propagates gradients through reused states. Across memory and agentic benchmarks, RETURN generally delivers stronger and more consistent performance than text-recall baselines without additional training. It also improves accuracy beyond the native context window. Despite probing overhead with short outputs, reduced GPU KV memory enables higher concurrency and improves serving throughput. On-policy distillation with parallel replay improves task performance, demonstrating compatibility with token-level supervision on formal assistant tokens. Code is available at `https://anonymous.4open.science/r/ReTurn`.

1 INTRODUCTION

With the rapid evolution of agent frameworks (Yao et al., 2023; Yang et al., 2024), the long-context bottleneck in Large Language Models (LLMs) has shifted from ultra-long single-turn prompts to massive multi-turn interactions. For example, in search and coding agent benchmarks like BrowseComp (Wei et al., 2025) and SWE-Bench (Jimenez et al., 2024), deriving the correct answer sometimes requires over 100 turns of tool-mediated interaction.

Context-Folding, ReSum, and MEM1 reduce working context through summaries or compact memory (Sun et al., 2025; Wu et al., 2026; Zhou et al., 2025). Compression requires anticipating future needs: a detail omitted during compression may become essential when an agent revisits a subtask or diagnoses a later failure. Retrieval-based approaches, including Memex(RL) and HiAgent, mitigate this limitation by retaining historical content for selective recall (Wang et al., 2026; Hu et al., 2024). However, restoring text without its original KV states requires history re-prefill, repeating computation whenever that content is recalled.

To address these limitations, we preserve recoverable history while avoiding repeated history re-prefill. We introduce RETURN, which manages active context by folding and restoring historical KV at turn granularity. Folding excludes a turn from attention while retaining its computed states. Restoration reactivates those states without re-encoding the historical text (Figure 1, *left*).

This fold-and-restore strategy is closely related to KV reuse in RAG, where independently encoded segments lack cross-segment dependencies. CacheBlend and KVLink compensate through selective recomputation and trained linking tokens, respectively (Yao et al., 2025; Yang et al., 2025). Agent histories offer a different starting point: each turn is encoded under the active prefix, so its cached states already incorporate preceding context. We retain these contextualized states and adjust their positional encoding when restoring them. Specifically, keys are stored before RoPE (Su et al., 2023)

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.

and rotated for their updated positions when reused. Section 2 provides empirical evidence that this KV reuse preserves task performance.

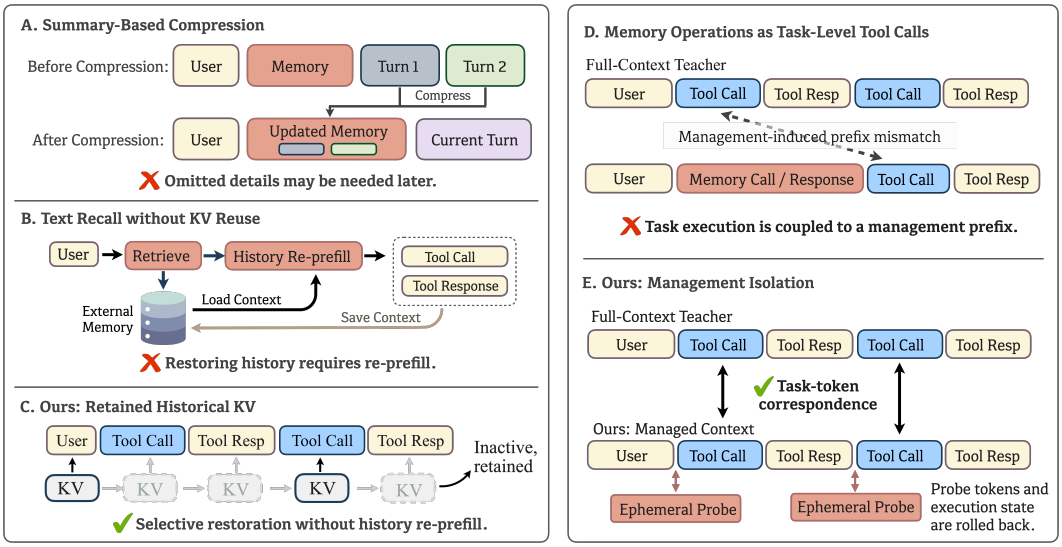


Figure 1: **Motivation.** *Left:* Summaries may omit details needed later (A), while text recall without KV reuse requires history re-prefill (B). RETURN retains historical KV for selective restoration without re-prefill (C). *Right:* Task-level memory calls couple task execution to a management protocol (D). Our ephemeral probe keeps this protocol outside the formal task trace by rolling back temporary tokens and execution state (E), preserving task-token correspondence for full-context teacher supervision despite different attention histories.

Selecting which turns to restore introduces a further challenge. Coupling memory management with task execution introduces two difficulties, whether history is compressed into summaries or retrieved as text (Wang et al., 2026; Yu et al., 2026b). First, the added management protocol can disrupt task behavior and require further training for adaptation. Second, it complicates supervised learning: SFT training data must include both task execution and appropriate memory operations, while on-policy distillation (OPD) (Agarwal et al., 2024) over both types of tokens requires a teacher capable of supervising the management protocol as well as the task.

We address these difficulties by isolating memory selection in an *ephemeral probe* (Figure 1, *right*). The probe reads turn-level digests and returns history IDs, after which its tokens and temporary execution state are rolled back. Only the selected history updates the active KV context. Formal task generation therefore remains free of management messages, allowing a full-context teacher to supervise the student's task tokens without processing the probe protocol.

Training must also reproduce the context under which each task token was generated. A standard causal pass over the formal transcript does not capture changing active histories and position mappings. Our path-aware parallel replay reconstructs these recorded choices under the training parameters and accumulates gradients through reused states. At each attention layer, given its inputs, different turns can be evaluated in parallel using their respective position mappings, with key gradients mapped back to the shared pre-RoPE states. Replay thus supports training with reversible context management while retaining task tokens as the supervision targets.

Together, these mechanisms make RETURN a framework for reversible context management: agents can stop attending to history without discarding its computed states, restore those states when needed, and keep management operations outside the task trace during both inference and training.

Our main contributions are:

(1) **History restoration without re-prefilling:** We separate KV retention from active attention and restore selected historical turns with updated positions.
(2) **Isolated memory management:** An ephemeral probe keeps history selection outside the task trace, enabling full-context teacher supervision without management labels.

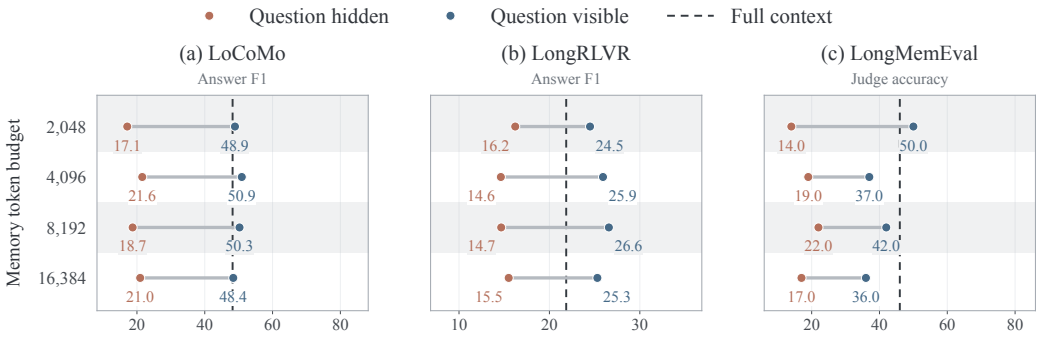


Figure 2: **Observation 1**: ReSum (Wu et al., 2026) performs worse than full context when history is compressed without knowing the question. Providing the question beforehand consistently improves answer quality.

(3) **Efficient serving and parallel training:** Our backend combines selective KV restoration with automatic offloading to reduce attention cost and GPU memory usage, enabling higher request concurrency. Path-aware parallel replay reproduces recorded history selections and position mappings while propagating gradients through reused KV states.

2 Motivating Observations

Unlike summary-based compression, ReTurn preserves access to original history and is therefore closer to text recall. Its distinction is to restore the corresponding KV states without history re-prefill. This design rests on two empirical premises: (1) preserving recoverable history is useful when future information needs are unknown, and (2) reusing turn-level historical KV can preserve task quality after folding and restoration. We examine these premises with Qwen3.5-4B (Qwen Team, 2026a).

2.1 The Need for Recoverable History

Progressive compression must decide what to retain before all future information needs are known. We examine this limitation using a training-free inference adaptation of ReSum (Wu et al., 2026) on LoCoMo (Maharana et al., 2024), LongRLVR (Chen et al., 2026), and LongMemEval (Wu et al., 2025). For example, LoCoMo requires recalling and reasoning about personal details and events scattered across multiple sessions of everyday conversation. History is compressed before the evaluation question is revealed, and the model subsequently answers using the retained memory. Full-context inference provides a reference using the same backbone and answer settings. For comparison, we also provide the evaluation question before compressing the history.

As shown in Figure 2, compressing history before the evaluation question is revealed reduces performance relative to the full-context baseline, while revealing the question improves scores at every tested token limit. This gap highlights the limitation of committing to a compressed memory before future information needs are known, motivating text recall and our method: both preserve access to original history so that relevant content can be recovered when needed. Further experimental details are provided in Appendix A.1.

2.2 The Viability of Historical KV Reuse

To assess whether selective KV restoration can preserve task quality, we evaluate two stages of KV restoration on LoCoMo, LongRLVR, LongMemEval, and RULER (Hsieh et al., 2024). Historical KV is constructed before the question is revealed. We first progressively restore oracle-identified supporting units, termed *Gold*. We then keep all Gold KV active and progressively restore the remaining history, comparing both stages with full-context inference.

Figure 3 shows that restoring Gold KV recovers answer quality close to or above the full-context reference while activating only a small fraction of history. Adding the remaining history provides little

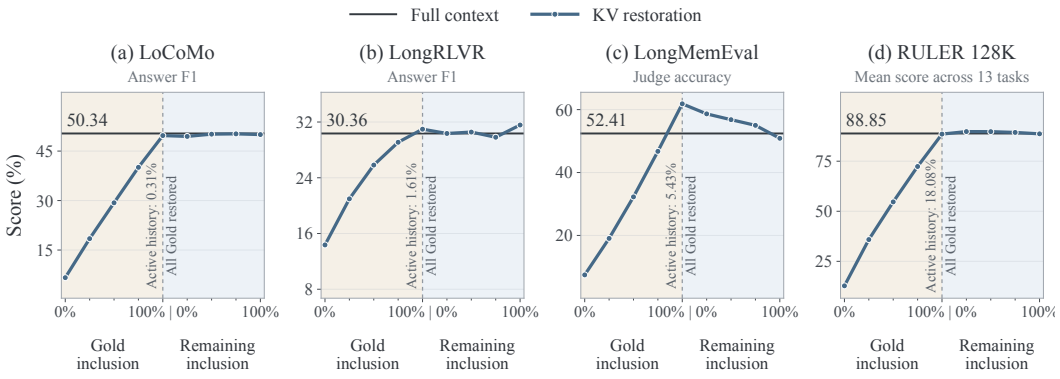


Figure 3: **Observation 2**: Selective KV restoration recovers answer quality. Restoring Gold-selected KV approaches or exceeds full-context scores while activating a small fraction of history. Adding the remaining history yields limited gains or reduces quality.

additional benefit and sometimes reduces quality. These results support recovering relevant history from retained KV without re-prefill. Further experimental details are provided in Appendix A.2.

3 METHOD

Building on these observations, we introduce RETURN, which retains historical KV for selective restoration without re-prefill and isolates memory management from task execution. We first describe the inference backend, then the ephemeral selection probe, and finally path-aware parallel replay for training. Figure 4 summarizes the inference pipeline.

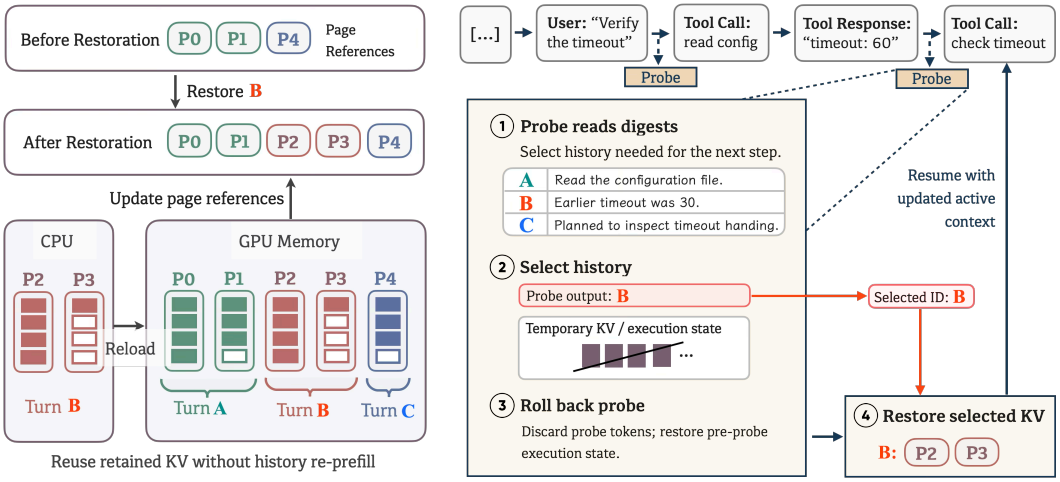


Figure 4: **Pipeline.** *Left:* Selected KV pages are reloaded from CPU to GPU and reactivated by updating page references, without history re-prefill. *Right:* Ephemeral probes isolate history selection from task execution, keeping management messages out of the task trace.

3.1 EFFICIENT KV MANAGEMENT

Our backend builds on paged KV management (Kwon et al., 2023) with three extensions: (1) turn-aligned allocation, (2) position adaptation within attention kernels, and (3) automatic offloading of inactive pages. Together, these mechanisms support selective restoration while reducing attention computation and GPU memory overhead.

Turn-aligned page management We manage history in complete turns. In agent workflows, a turn is an assistant or tool message. Input batches need not align with these boundaries: a prefill may span multiple messages or include the generation prefix of the next message. We therefore detect message boundaries in the token stream and begin each turn on a fresh KV page. For Qwen models (Qwen Team, 2026a), we use <|im_start|> as the boundary marker. Each turn maps to its own pages and valid token counts, so folding or restoring one turn does not affect another. Folding removes page references from active attention while retaining the underlying KV. Restoration reinserts those references in chronological order without re-encoding history.

Position adaptation within attention Folding changes the logical positions of active tokens. We retain keys before RoPE and apply the rotation for their current positions inside the attention kernel. Let $\mathcal{A}_t$ be the ordered set of historical turns active for turn t, with turns indexed in their original chronological order, and let n_i be the number of tokens in turn i. For token r in historical turn $i \in \mathcal{A}_t$,

$$p_t(i,r) = \sum_{j \in \mathcal{A}_t,\, j<i} n_j + r, \qquad \widetilde{k}^{(t)}_{i,r} = R(p_t(i,r))\, k^{\mathrm{pre}}_{i,r}, \tag{1}$$

where $p_t(i,r)$ is the token's logical position in the active context for turn t, obtained by counting tokens in the preceding active turns. The vector $k^{\mathrm{pre}}_{i,r}$ is its retained key before RoPE, $R(p)$ is the RoPE rotation matrix at position p. Layer and head indices are omitted for clarity. Given the positional coefficients, the attention kernel applies RoPE to shared keys on the fly, without materializing rotation matrices or writing rotated key copies back to HBM.

Automatic KV offloading We offload inactive KV pages to CPU to reduce GPU memory usage. We analyze BrowseComp (Wei et al., 2025) and SWE-Bench (Jimenez et al., 2024) trajectories with histories longer than 64K tokens. Each turn's selection frequency is its selection count divided by the number of probes for which it is eligible. Figure 5 shows a highly skewed distribution: most turns are rarely restored, while only a few are selected frequently. Appendix A.3 details the statistical protocol. This skewed reuse pattern motivates offloading inactive KV pages to CPU to reduce GPU memory usage. We use a least-recently-used (LRU) policy. Under allocation pressure, we offload inactive pages in LRU order until enough GPU memory is available for the requested allocation. Selected pages are loaded back before task generation resumes.

Although CPU–GPU transfers introduce latency, we batch page transfers at probe boundaries, with at most one offload batch and one reload batch per cycle. The selected pages remain on the GPU throughout the following assistant generation and tool-response processing, amortizing transfer overhead across these steps.

Offloading frees GPU memory to accommodate more concurrent requests, while folding reduces attention computation. Higher concurrency lets the GPU serve other requests during tool waits, improving utilization and serving throughput.

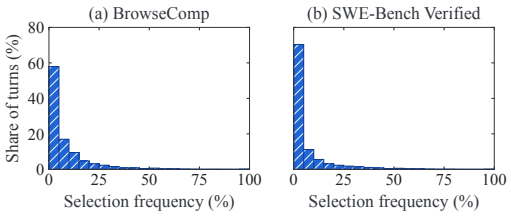


Figure 5: Most historical turns are rarely selected for restoration, making their KV states well suited to CPU offloading.

3.2 Management Isolation via Ephemeral Probes

An external LLM generates a one-line digest for each completed turn, which we pair with a unique ID. Before each assistant generation, we fold all eligible historical turns, including previously restored turns, while retaining the system and user prompts and recent turns. The agent then reads the ID–digest table of folded turns in an ephemeral probe and returns the IDs of turns to restore.

We discard the probe tokens and restore the pre-probe execution state, including any recurrent states, before reactivating the selected KV pages. This keeps memory management outside the task trace, simplifying SFT and OPD. Digests only guide selection. Details omitted from them remain recoverable through the retained KV. Our inclusive probe encourages restoring uncertain candidates and permits selecting all eligible history when feasible (Appendix A.5.1).

Task-specific settings and complete probe and digest prompts are provided in Appendix A.5.

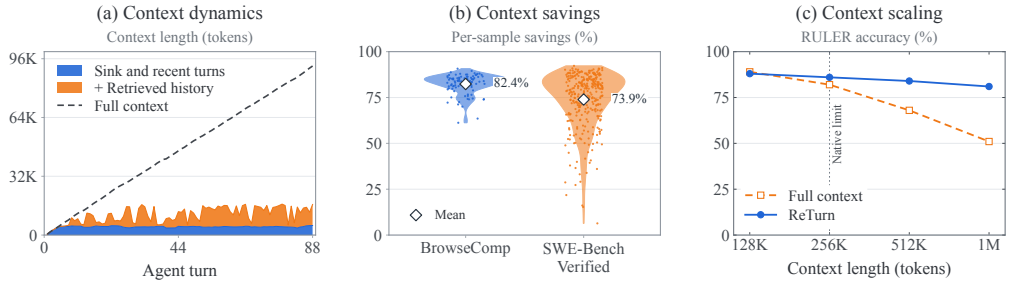


Figure 6: **Context savings and scaling.** (a) Active context remains compact as history grows, with approximately 75% savings on a BrowseComp trajectory. (b) Context savings distributions on BrowseComp and SWE-Bench Verified. (c) RULER accuracy as context length exceeds the native window, comparing RETURN with full-context inference.

3.3 PATH-AWARE PARALLEL REPLAY

Parallel replay reproduces recorded history selections and position mappings while supervising only formal task tokens. Attention masks alone are insufficient because shared keys can occupy different RoPE positions across turns.

Parallel forward pass At each attention layer, one kernel processes query blocks from all turns in parallel. Each block reads its turn's selected KV pages and applies RoPE at that turn's logical positions, reusing shared keys without storing rotated copies.

Gradient accumulation Let $\mathcal{U}_i$ denote the turns that reuse turn i's KV. Using Equation 1, we map each key gradient back to its pre-RoPE coordinates and sum across uses:

$$\frac{\partial \mathcal{L}}{\partial k^{\text{pre}}_{i,r}} = \sum_{t \in \mathcal{U}_i} R(p_t(i,r))^\top \frac{\partial \mathcal{L}}{\partial \widetilde{k}^{(t)}_{i,r}}. \tag{2}$$

The backward kernel atomically accumulates these gradients in a shared buffer. For hybrid models, recurrent layers process the complete non-probe sequence, carrying state across turns regardless of KV folding.

4 EXPERIMENTS

We evaluate RETURN on memory benchmarks including LoCoMo (Maharana et al., 2024), LongMemEval (Wu et al., 2025), RULER (Hsieh et al., 2024), and LongRLVR (Chen et al., 2026), and agentic benchmarks including BrowseComp (Wei et al., 2025) and SWE-Bench Verified (Jimenez et al., 2024). We use three models: Qwen3.5-4B (Qwen Team, 2026a), Qwen3.6-35B-A3B (Qwen Team, 2026b), and Qwen3.8-27B. We run evaluations on 8 NVIDIA H200 GPUs. We serve the evaluated models with our own inference engine and use Qwen3.5-4B, served with SGLang (Zheng et al., 2024), as the external summarizer unless otherwise specified. A performance comparison between our engine and SGLang is provided in Appendix B. Unless otherwise specified, RETURN retains 3 recent turns and 1 sink turn, with dynamic offloading enabled. The sink turn contains the system prompt for memory tasks and both the system prompt and user question for agentic tasks. Further experimental details are provided in Appendix A.

4.1 EFFECTIVENESS

Context Savings and scaling Using Qwen3.5-4B, we examine active-context savings on agentic benchmarks and context-length scaling on RULER. Figure 6(a) provides a concrete example of how RETURN keeps the active context compact as history grows. Figure 6(b) shows the distribution of context savings across trajectories whose nominal context lengths exceed 64K tokens. For context

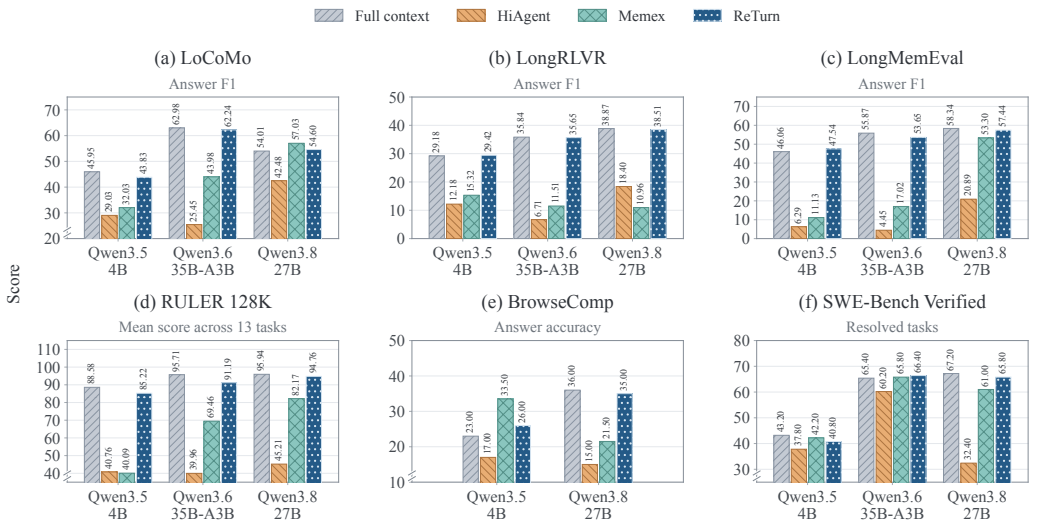


Figure 7: **Benchmark performance.** RETURN approaches full-context performance across the reported configurations. HiAgent (Hu et al., 2024) and Memex (Wang et al., 2026) perform well on some agentic benchmarks, but their performance varies substantially across models and datasets. We omit Qwen3.6-35B-A3B results on BrowseComp because all methods yield near-zero accuracy, with per-turn outputs repeatedly truncated at the 32K-token limit.

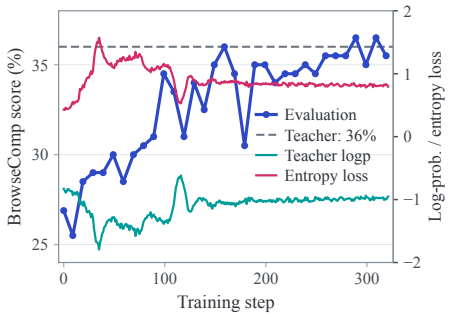


Figure 8: **Training compatibility.** Isolated memory management and parallel replay enable RETURN to train with dense token-level supervision. OPD performance gains demonstrate this compatibility.

scaling, RETURN achieves substantially higher accuracy than the full-context baseline on RULER beyond the model's native 256K window (Figure 6(c)).

Benchmark Results In a training-free setting, we compare RETURN with full-context inference, HiAgent (Hu et al., 2024), and Memex (Wang et al., 2026) on memory and agentic benchmarks using Qwen3.5-4B, Qwen3.6-35B-A3B, and Qwen3.8-27B. We evaluate BrowseComp on a randomly sampled subset of 200 questions. Figure 7 reports completed evaluations. Across the reported configurations, RETURN performs consistently close to the full-context baseline.

Parallel Replay Training Isolated memory management and parallel replay allow RETURN to use standard training methods with dense token-level supervision. We demonstrate this compatibility through OPD with a Qwen3.5-4B student and a Qwen3.8-27B teacher, using 1,066 BrowseComp questions for training and 200 for evaluation. Figure 8 shows that task performance improves under this training procedure.

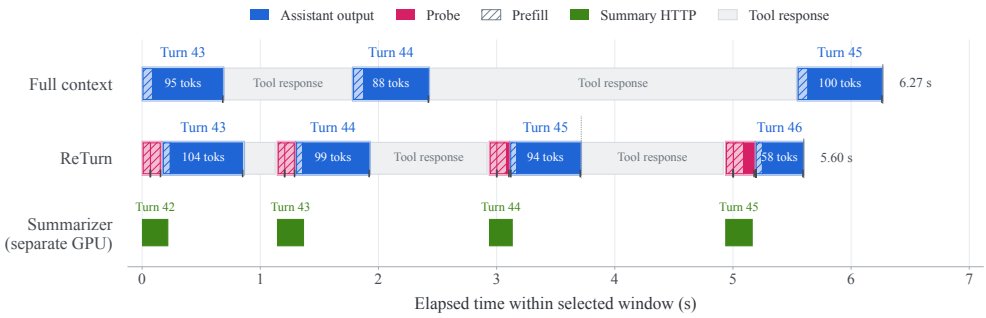
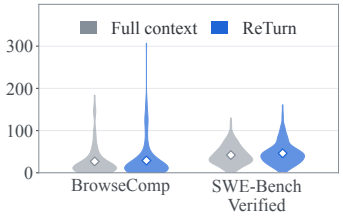


Figure 9: **Latency breakdown.** (1) Summarization overlaps with inference. (2) Probing adds modest inference overhead. The assistant generates only about 100 tokens per turn in this setting, so probe overhead is not negligible.[1] (3) The serving benefit comes from lower GPU KV cache usage, which enables higher concurrency and shorter workload completion time, as shown in Table 1.

Table 1: **Serving efficiency.** Stress-test results for 1,000 BrowseComp trajectory replays submitted at 0.3-second intervals over approximately five minutes. Both systems use 8 GPUs: 8 inference engines for full-context inference, versus 7 inference engines and one summarizer for RETURN.

Method	Workload completion time (s) ↓	Mean decoding throughput (tps) ↑	Mean first-turn TTFT (s) ↓	Mean later-turn TTFT (s) ↓
Full context	5938.28	5869.95	1693.91	**0.18**
RETURN	**3021.13**	**11206.47**	**1.42**	0.47

Figure 10: **Interaction turns.** Assistant-turn distributions, excluding probes. Diamond markers indicate the means.



4.2 EFFICIENCY

All experiments in this subsection use Qwen3.5-4B. Serving efficiency results for Qwen3.8-27B are provided in Appendix A.6.

Serving Efficiency Figure 9 shows that summarization overlaps with inference, while probing adds modest inference overhead. The assistant generates only about 100 tokens per turn in this setting, so probe overhead is not negligible. The main efficiency gain comes from lower GPU KV cache usage, allowing more requests to run concurrently. While one request waits for a tool response, the GPU can process others, increasing overall throughput.

Table 1 shows that the higher concurrency translates into shorter workload completion time. We cap baseline concurrency at 256, the largest value tested in increments of 64 without KV pool overflow, and impose no concurrency cap on RETURN. The baseline's high mean first-turn TTFT is primarily due to queueing before admission.

Interaction Turns Figure 10 compares the number of assistant turns between RETURN and the full-context baseline on BrowseComp and SWE-Bench Verified, excluding probe calls. The mean increases from 27 to 29 on BrowseComp and from 42 to 46 on SWE-Bench Verified. This slight increase with RETURN may reflect uncertainty when relevant history is absent from the active context.

4.3 ABLATIONS

On History Selection RETURN keeps one sink turn and three recent turns active, and restores additional history selected by the probe. On BrowseComp, we test two ablations: disabling history restoration and selecting history at random. The random variant selects exactly as many historical turns as the probe at each step. Table 2 shows that both ablations reduce accuracy.

[1]Each probe currently uses two separate prefill passes. The observation and probe prompt arrive together, but we split prefill at their boundary to save a rollback snapshot. This overhead remains to be optimized.

On Summarizer Choice To assess sensitivity to summarizer choice, we compare the default Qwen3.5-4B summarizer with GPT-5-mini and GPT-5.4 on BrowseComp. Table 3 shows only modest accuracy gains from stronger summarizers for both task models.

Table 2: **History selection.** BrowseComp accuracy (%) ↑ on 200 questions.

Variant	Qwen3.5 4B	Qwen3.8 27B
RETURN	**26.0**	**35.0**
+ No history restoration	14.5	19.5
+ Random history selection	17.0	21.5

Table 3: **Summarizer choice.** BrowseComp accuracy (%) ↑ on 200 questions.

Variant	Qwen3.5 4B	Qwen3.8 27B
RETURN	26.0	35.0
+ GPT-5-mini summarizer	**26.5**	36.0
+ GPT-5.4 summarizer	**26.5**	**38.0**

On KV Reuse and CPU Offloading Using Qwen3.5-4B, we evaluate two variants: re-prefilling all selected and recent turns as in text-recall methods, and disabling CPU offloading. We measure workload completion time under the serving setup in Section 4.2. Table 4 shows that KV reuse and CPU offloading both reduce workload completion time.

Table 4: **KV reuse and CPU offloading.** Serving ablations with Qwen3.5-4B. Relative time is normalized to RETURN.

Variant	Workload completion time (s) ↓	Relative time (×) ↓
RETURN	3021.13	1.00
+ Text recall re-prefill	7145.83	2.37
+ No CPU offloading	6649.76	2.20

5 RELATED WORK

Context management Existing methods reduce agent context through summary-based compression or text recall without KV reuse. ReSum (Wu et al., 2026), MEM1 (Zhou et al., 2025), MemAgent (Yu et al., 2026a), and FoldGRPO (Sun et al., 2025) condense earlier interactions into compact summaries or memory states, potentially losing details needed later. HiAgent (Hu et al., 2024) and Memex (Wang et al., 2026) instead support selective recall of archived interactions, but restoring history as text requires re-prefill. RETURN uses summaries to guide selection while retaining the original historical KV, allowing selected turns to be restored without re-encoding their text.

Sparse attention Sparse attention and RETURN both reduce computation by restricting attention to selected history. Quest (Tang et al., 2024) selects top-k KV pages for every query token during decoding using query vectors and key statistics. InfLLM (Xiao et al., 2024) retrieves blocks through query–key similarity and supports CPU offloading. Such fine-grained selection introduces recurring lookup and memory-access costs. RETURN selects complete turns from natural-language digests at probe boundaries, using explicit semantic selection rather than attention-level scores. The selected KV remains on the GPU throughout subsequent generation and tool-response processing, while inactive history can stay on the CPU. This coarser selection granularity amortizes selection and transfer costs across multiple tokens.

6 CONCLUSION AND LIMITATIONS

We present RETURN, a framework for reversible context management in LLM agents that supports token-level supervision on formal assistant tokens. It restores historical KV without re-prefill and combines isolated memory management with parallel replay to enable supervision from a full-context teacher without management labels. Across memory and agentic benchmarks, RETURN generally performs better and more consistently than text-recall baselines without additional training and improves accuracy beyond the native context window. CPU offloading reduces GPU KV memory usage, enabling higher concurrency and serving throughput.

We have not yet trained RETURN on larger-scale reinforcement learning tasks. Our evaluation covers recent mainstream hybrid models, leaving earlier full-attention models untested.

AI USE STATEMENT

We used generative AI tools to refine language and formatting, check grammar, logical consistency, citations, and typographical errors, and assist with figure-generation scripts. The authors take responsibility for the final manuscript and artifacts.

ETHICS STATEMENT

We do not identify ethical concerns specific to this work on reversible context management and efficient LLM inference.

REPRODUCIBILITY STATEMENT

We provide the RETURN inference engine, parallel replay, benchmark evaluation scripts, and custom acceleration kernels at `https://anonymous.4open.science/r/ReTurn`. Experimental settings, prompts, and training details are provided in Appendix A.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. *arXiv preprint arXiv:2306.13649*, 2024.

Guanzheng Chen, Michael Qizhe Shieh, and Lidong Bing. Longrlvr: Long-context reinforcement learning requires verifiable context rewards. *arXiv preprint arXiv:2603.02146*, 2026.

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

Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. Hiagent: Hierarchical working memory management for solving long-horizon agent tasks with large language model. *arXiv preprint arXiv:2408.09559*, 2024.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2024.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, et al. Efficient memory management for large language model serving with pagedattention. *arXiv preprint arXiv:2309.06180*, 2023.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents. *arXiv preprint arXiv:2402.17753*, 2024.

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

Qwen Team. Qwen3.6-35B-A3B: Agentic coding power, now open to all, 2026b. URL `https://qwen.ai/blog?id=qwen3.6-35b-a3b`.

Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. *arXiv preprint arXiv:2407.08608*, 2024.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2023.

Weiwei Sun, Miao Lu, Zhan Ling, Kang Liu, Xuesong Yao, Yiming Yang, and Jiecao Chen. Scaling long-horizon llm agent via context-folding. *arXiv preprint arXiv:2510.11967*, 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.

Zhenting Wang, Huancheng Chen, Jiayun Wang, and Wei Wei. Memex(rl): Scaling long-horizon llm agents via indexed experience memory. *arXiv preprint arXiv:2603.04257*, 2026.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, et al. Browsecomp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*, 2025.

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*, 2025.

Xixi Wu, Kuan Li, Yida Zhao, Liwen Zhang, Litu Ou, Huifeng Yin, Zhongwang Zhang, et al. Resum: Unlocking long-horizon search intelligence via context summarization. *arXiv preprint arXiv:2509.13313*, 2026.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, et al. Infllm: Training-free long-context extrapolation for llms with an efficient context memory. *arXiv preprint arXiv:2402.04617*, 2024.

Jingbo Yang, Bairu Hou, Wei Wei, Yujia Bao, and Shiyu Chang. Kvlink: Accelerating large language models via efficient kv cache reuse. In *NeurIPS*, 2025.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *arXiv preprint arXiv:2405.15793*, 2024.

Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, et al. Cacheblend: Fast large language model serving for rag with cached knowledge fusion. 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*, 2023.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, et al. Memagent: Reshaping long-context llm with multi-conv rl-based memory agent. *arXiv preprint arXiv:2507.02259*, 2026a.

Yi Yu, Liuyi Yao, Yuexiang Xie, Qingquan Tan, Jiaqi Feng, Yaliang Li, and Libing Wu. Agentic memory: Learning unified long-term and short-term memory management for large language model agents. *arXiv preprint arXiv:2601.01885*, 2026b.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. Sglang: Efficient execution of structured language model programs. *arXiv preprint arXiv:2312.07104*, 2024.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, et al. Mem1: Learning to synergize memory and reasoning for efficient long-horizon agents. *arXiv preprint arXiv:2506.15841*, 2025.

A Additional Experimental Details

A.1 Question Visibility in ReSum

Scoring Scoring follows Appendix A.2.

Compression and question visibility We adapt ReSum (Wu et al., 2026) without training, using Qwen3.5-4B to summarize history incrementally and answer questions. Each update merges the previous summary with new history, triggered at approximately 28.6K active tokens or 1,000 historical units; remaining history is summarized before answering. The key comparison is whether the evaluation question is visible during compression. When hidden, the prompt asks the model to preserve facts for future questions, and one summary is reused across questions sharing a history. Both conditions answer from the final summary and question, without access to reference answers.

Prompts We retain the official templates for both initial and cumulative summaries.[2] Only the question field changes between the visible and hidden conditions, as shown in Figure 11.

Official / question visible	Question hidden
`Question` `- {{{question}}}`	`Question` `+ The evaluation question has not been revealed. Preserve explicit factual details for future questions about this history. Do not ask for clarification.`

Figure 11: **ReSum prompt adaptation.** When the question is hidden, it is replaced with an instruction to preserve facts for future questions.

Decoding Summaries use temperature 0.6, top-p 0.95, no top-k filtering or repetition penalty, and output limits of 2,048, 4,096, 8,192, or 16,384 tokens. Each trajectory is sampled once, without matched generation seeds, retries, or truncation. Answers use greedy decoding with thinking disabled and limits of 32, 128, and 256 tokens for LoCoMo, LongRLVR, and LongMemEval, respectively. Full-context inference uses the same questions and answer settings.

A.2 Oracle KV Restoration

Figure 3 evaluates KV restoration with oracle-selected history, without a probe or summarizer. Historical states are computed before the question is revealed; all conditions share the same questions and answer settings.

Oracle evidence Gold consists of complete historical units, restored in chronological order.

Benchmark	Evaluation samples	Oracle evidence
LoCoMo	1,435 questions from categories 1, 2, and 4; excludes invalid mappings and three ambiguous or incomplete evidence chains.	Evidence-linked turns, with session dates preserved.
LongRLVR	All 61 examples in the selected 64K–128K split.	Chunks identified by `extra_info.ref_chunks`.
LongMemEval	456 questions; excludes abstention cases and invalid mappings.	Complete sessions identified by `answer_session_ids`.
RULER	50 examples per task across 13 tasks.	1,024-token chunks covering supporting key–value statements for NIAH; answer-containing chunks as oracle proxies for CWE, FWE, VT, and QA.

Gold may not capture all useful evidence, so unselected history can still contribute to answering.

[2] https://github.com/Alibaba-NLP/DeepResearch/blob/f72f75d8c3eb842f2bbbab096a12206ff66e270f/WebAgent/WebResummer/src/prompt.py

Restoration protocol We vary the restored history in two stages. First, we restore only Gold units, selecting each independently with probability $p \in \{0, 0.25, 0.5, 0.75, 1\}$. We then keep all Gold units and select each remaining unit with probability $p \in \{0.25, 0.5, 0.75, 1\}$. These nine settings span no restoration, Gold-only restoration, and full-history restoration. The probability controls unit selection; the resulting active-history token fraction is measured separately.

For each question and setting, we average scores over ten sampling seeds. Within each seed, higher probabilities retain all units selected at lower probabilities. Dataset scores average over questions; RULER averages the mean scores of its 13 tasks. Confidence intervals use 2,000 bootstrap replicates, clustered by conversation for LoCoMo and by question otherwise, with task stratification for RULER.

Scores are percentages: category-aware F1 for LoCoMo, token F1 for LongRLVR, and the official evaluators for RULER and LongMemEval. LongMemEval uses its question-type-specific judging prompts with `gpt-4o-2024-08-06` at temperature 0.

Model and KV state Qwen3.5-4B uses a 262,144-position limit, unchanged RoPE scaling, and no history truncation. Answers use greedy decoding with thinking disabled and output limits of 32, 128, 256, and 128 tokens for LoCoMo, LongRLVR, LongMemEval, and RULER, respectively.

Restoration changes full-attention KV visibility and assigns compact RoPE positions to restored keys. The Gated DeltaNet state retains information from the complete history and is shared across selections. Query and answer updates are rolled back between trials. Active-history fractions therefore measure explicit KV attention, not all historical information available to the hybrid model.

A.3 Turn Selection Statistics

Trajectories Figure 5 uses the Qwen3.5-4B trajectories analyzed for context savings. We retain histories longer than 65,536 tokens, excluding temporary probe tokens. All probe steps within each retained trajectory are counted.

Statistic Each assistant or tool message is one turn. We reconstruct eligibility and selection from recorded fold and restore operations. A turn's frequency is its selection count divided by its eligible probe count. Eligible turns that are never selected receive zero. Turns that are never eligible are excluded. We pool turns with equal weight and plot their percentages in bins of five percentage points.

A.4 Main Experiment Settings

Context savings and scaling For trajectory i, context saving is

$$s_i = 1 - \frac{\sum_t A_{it}}{\sum_t N_{it}}, \tag{3}$$

where A_{it} is the active token count after restoration and N_{it} is the count for the same history without folding, both measured before assistant generation. Each trajectory contributes all its turns and receives equal weight. Probe and summarizer work is excluded.

Benchmark performance We evaluate all 1,986 LoCoMo questions across 10 conversations and all 61 examples in LongRLVR's 64K–128K test split. RULER uses 50 examples per task across 13 tasks at 128K tokens. For LongMemEval, we use the same fixed set of 200 questions randomly sampled from the S split across methods and models. BrowseComp uses a randomly sampled subset of 200 out of 1,266 questions. SWE-Bench Verified uses all 500 test instances.

Full-context inference retains the complete history. We evaluate HiAgent and Memex without parameter updates. For agentic tasks, we adapt their interfaces to the same task tools. HiAgent retains subgoal summaries and retrieves complete subgoal histories, using Qwen3.5-4B for summarization. Memex uses its RAG implementation with summaries generated by the task model and BM25 top-1 retrieval. Both restore historical text through re-prefill.

LoCoMo and LongRLVR scoring follows Appendix A.2. All scores in the main LongMemEval comparison use Answer F1; the motivating observations use the separate cohorts and official judging

protocol described in Appendices A.1 and A.2. RULER reports the mean score across its 13 tasks. BrowseComp reports answer accuracy, and SWE-Bench Verified reports the resolved-instance rate.

Prompts and decoding settings for RETURN are provided in Appendix A.5. Memory tasks are evaluated once. For agentic tasks, we report the mean score over three runs.

Parallel replay training The OPD objective combines forward KL, reverse KL optimized via policy gradients, and an entropy bonus, weighted by 3, 1, and 0.01, respectively. Forward KL is estimated using the teacher's top-10 tokens and a residual bucket containing the remaining probability mass. RETURN computes student gradients through parallel replay, with supervision restricted to formal assistant tokens.

We train all student parameters with AdamW at a learning rate of 10^{-6} on 24 NVIDIA H200 GPUs. Each batch contains 16 prompts with 4 samples per prompt.

Serving efficiency Figure 9 shows the latency breakdown for selected turns. Table 1 evaluates concurrent serving with the same request arrival order for both methods. Workload completion time spans the first arrival to the last completion. For both methods, TTFT spans engine submission to the first formal assistant token, including queueing and all intervening processing. We report separate means for first and subsequent turns. Peak concurrency excludes requests awaiting admission.

We select BrowseComp questions with histories over 64K tokens under both methods and assistant-turn counts within 10%, then repeat their trajectories into 1,000 requests in a fixed shuffled order. Each method replays its own recorded trajectories, using teacher forcing for the recorded assistant outputs and reusing the recorded tool responses. All model computation, including probes and summaries, is executed during replay, and sleep calls reproduce each trajectory's recorded web latency. This preserves each method's original trajectory while using the same request arrival order and replay protocol across methods. Admission requires a resident slot and sufficient KV capacity, prioritizing ongoing sessions. The baseline keeps KV on GPU. RETURN uses LRU CPU offloading only under allocation pressure. Mean decoding throughput (tokens/s) is the time average of server-reported throughput summed across inference engines, including probe tokens but excluding summarizer tokens. Each configuration has three timed runs, and we report the mean across runs, excluding model loading, CUDA graph capture, and replay warm-up from timing.

Interaction turns Each formal assistant response, including tool calls and the final answer, counts as one turn. Probes and summarizer calls are excluded. Both methods use the same 200 BrowseComp questions and 500 SWE-Bench Verified instances. The BrowseComp traces use an inclusive probe with an assistant guide, Qwen3.5-4B summarization, and a full-context baseline with matched answer sampling. All recorded outcomes are included, including repetitions and truncated attempts.

A.5 PROBE AND DIGEST PROMPTS

Memory tasks comprise LoCoMo, LongMemEval, RULER, and LongRLVR. Agentic tasks comprise BrowseComp and SWE-Bench Verified. Both use inclusive probes with an assistant guide and the settings below.

Setting	Memory tasks	Agentic tasks
Probe temperature	0	0
Answer temperature	0	0.6
Answer top-p / top-k	Greedy decoding	0.95 / 20
Digest length anchor (words)	100	12
Probe output limit	256 tokens	256 tokens

Digest word counts serve as prompt anchors, not enforced length limits.

Inclusive probe Each turn has a stable uppercase alphabetic ID, such as `A`, `B`, or `C`. The template below retains inclusive selection while using space-separated IDs and `NONE` for an empty selection.

```
The table lists passages currently hidden from your context. The
digests are brief and may omit useful details.

id: digest
{table}

Select every passage that could help answer the question, including
passages that may supply missing context. If you are unsure whether a
passage is needed, include it. There is no fixed number to select.

Reply only with selected uppercase ids from the table, each at most
once. Separate ids with spaces. Use a single line; no commas,
explanations, answers, or tool calls. If none are needed, output NONE.
```

Assistant guide We prefill the following guide after the probe's no-thinking assistant prefix, ending at the colon. The model then generates the selected IDs.

```
Selected ids (space-separated, no reasoning):
```

Memory digest Memory tasks use the factual retrieval-index template below.

```
Write a one-line retrieval index for the passage below, at most 100
words.
Use concrete names and distinctive facts rather than broad topic
labels. Keep who did, said, or experienced what clear.
Do not let lengthy generic advice crowd out brief factual statements
from the speakers.
If the passage covers several distinct events or topics, include cues
to each instead of describing only the dominant topic.
Use the passage's wording where possible. Include dates or temporal
links when they distinguish events.
Do not invent facts or replace specific details with generic
commentary. Output only the index entry.

PASSAGE:
{passage}
```

Tool observation digest Agentic tasks use separate templates for tool observations and assistant messages. Each template receives only the corresponding turn text.

```
One line. This is an INDEX ENTRY for a tool result, not a summary: the
full
text stays retrievable on demand, so do NOT try to preserve it. The
single question the
line must answer is "would an agent hunting for some fact need to pull
this back?"

If the result is empty, an error, a login/paywall page, or off topic,
say exactly that
and nothing else -- that tells the agent never to pull it back.
Write in the language of the observation.

Be pathologically terse. No preamble, no meta-commentary. At most 12
words.

OBSERVATION:
{text}
```

Assistant turn digest

```
One line. This is an INDEX ENTRY of what the agent did.

Format:  <conclusion>; <tool>(<args>)

If the turn calls a tool, write it as a call expression: the tool's
own name, lowercase
and verbatim as it appears in the turn, then its argument values in
parentheses. Never
describe it in prose ("searched for X", "retrieved Y via the tool")
and never reproduce
<function=...> / <parameter=...> markup or JSON envelopes. Strip URLs
down to
host/last-path-segment, truncate long values -- but the call
expression is mandatory and
the tool name is never dropped, no matter how tight the word budget
gets.
If the turn calls no tool, write only the conclusion.

Shape of the line (substitute the turn's real tool name and arguments):
    nothing conclusive yet; TOOLNAME("first argument")
    need the missing field; TOOLNAME(short-arg, short-arg)
    answer settled: ANSWER

Be pathologically terse. No preamble, no meta-commentary. At most 12
words.

AGENT TURN:
{text}
```

A.5.1 Selection with Incomplete Digests

Digests provide retrieval cues without needing to enumerate every detail in a turn. A detail omitted from a digest can still be recovered by restoring the turn based on broader cues, such as its topic, entities, or tool operation. To support recall under uncertainty, the inclusive prompt above explicitly instructs the probe: "If you are unsure whether a passage is needed, include it. There is no fixed number to select."

This policy allows the probe to broaden its selection, up to all eligible folded turns, when the digests do not sufficiently narrow the relevant history. Full restoration is feasible when the selected IDs fit within the probe's output budget and the restored context fits within the model's execution limits. The selected history is then accessible through its retained KV without history re-prefill. Actual recall still depends on the probe's selections, and broader restoration increases the active-context cost.

A.6 Serving Efficiency of Qwen3.8-27B

Table 5: **Serving efficiency of Qwen3.8-27B.** Measured results for the workload in Table 1.

Method	Workload completion time (s) ↓	Mean decoding throughput (tps) ↑
Full context	24311.35	1498.10
ReTurn	10770.69	3156.82

B Inference Engine Comparison with SGLang

We summarize our backend comparison using Qwen3.5-4B in BF16 on the same H200 GPUs, with one GPU per run. The reference is SGLang 0.5.10rc0 (Zheng et al., 2024) with the FA3 attention backend (Shah et al., 2024). Our engine uses 64-token KV pages.

Table 6 reports decode throughput excluding prefill and serving overhead. For mixed-length batches, both engines use the same prompts sampled uniformly between 0 and 64K tokens.

Table 6: **Decode throughput relative to SGLang.** Values are our engine's throughput divided by SGLang's. Missing entries indicate out-of-memory errors during SGLang prefill.

Batch size	1	2	4	8	16	32
Equal-length prompts: decode						
4K tokens	0.96	0.95	0.97	0.92	0.90	0.94
8K tokens	0.97	0.97	0.97	0.96	0.95	0.91
16K tokens	0.94	0.97	0.96	0.98	0.95	0.94
32K tokens	0.94	0.98	0.97	1.01	1.03	–
64K tokens	0.94	1.00	1.10	1.00	–	–
Mixed-length prompts: 0–64K tokens						
Decode	0.79	0.85	0.82	0.82	1.03	–
One 128K prompt with additional 4K prompts						
4K requests	1	2	4	8	16	
Total batch size	2	3	5	9	17	
Decode	0.82	0.84	0.83	0.82	0.83	

Our engine reaches 0.90–1.10× SGLang's decode throughput for equal-length batches and 0.79–1.03× for mixed-length batches. A separate stress test combining one 128K prompt with 1–16 short 4K prompts yields 0.82–0.84×.