Hallucination-Filtered Memory Supervision Enables Memory Agents Without Reinforcement Learning

Anonymous ACL submission

Abstract

Long-context question answering requires extracting and integrating sparse evidence from streams dominated by irrelevant information. Recent memory agents address this challenge by learning to update an explicit memory while processing long inputs, but their memory-update policies are typically learned through reinforcement learning over long-horizon rollouts, requiring specialized infrastructure and thousands of GPU-hours. Supervised fine-tuning is a more efficient alternative, but faces two obstacles. First, rollout trajectories grow with the number of memory updates and routinely exceed standard SFT context windows. Second, the trajectories themselves contain hallucinated memory updates: we find that over half of memory updates on unsupported inputs inject entities or numerical facts absent from both the input and the prior memory, a bias that a naive supervised model inherits To address both obstacles, we introduce Hallucination-filtered Memory Supervision (HaMS). HaMS first filters the source trajectories at the level of individual memory-update steps, removing examples whose target memory adds ungrounded content. It then applies stepwise supervision on the filtered pool, decomposing each trajectory into independent training examples that match the inference-time protocol and bound per-example context length to a single update step. Using over $55\times$ less post-pretraining compute than RL-trained memory agents, HaMS exceeds the RL source on HotpotQA across the full distractor sweep up to 0.9M-token inputs, and matches or surpasses RL-trained memory agents on average across 2WikiMultiHopQA and five LongBench-QA tasks.

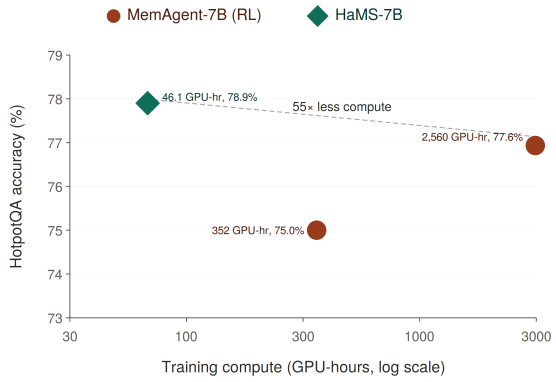

Figure 1: **Atomic memory update SFT matches RL at a fraction of the cost.** Replacing reinforcement learning with per-step supervised fine-tuning over RL teacher trajectories, HaMs matches the RL-trained MemAgent-7B on HotpotQA at $55\times$ less post-pretraining compute. All systems share the Qwen2.5-7B-Instruct base.

1 Introduction

Long-context question answering requires extracting sparse evidence from inputs dominated by irrelevant information. Prior work has focused on extending LLM context windows through modifications to positional embeddings (Peng et al., 2024; Ding et al., 2024; Shang et al., 2025) or pretraining on longer sequences (Yang et al., 2025; Wan et al., 2025), but these models show sharp performance degradation before the input reaches their theoretical context capacities (Yu et al., 2026). Memory agents (Yu et al., 2026; Shi et al., 2026; Yan et al., 2025; Zhou et al., 2025; Wang et al., 2025b) employ an explicit memory updated as new information arrives, allowing them to process arbitrarily long inputs while preserving and aggregating task-relevant signal over the stream.

Memory-update policies in this paradigm are typically learned with reinforcement learning (Shao et al., 2024; Yu et al., 2025), requiring specialized RL infrastructure and thousands of GPU-hours. We examine whether memory-update policy can be learned through substantially more efficient training recipes. Supervised fine-tuning has emerged as an efficient substitute for RL in reason-

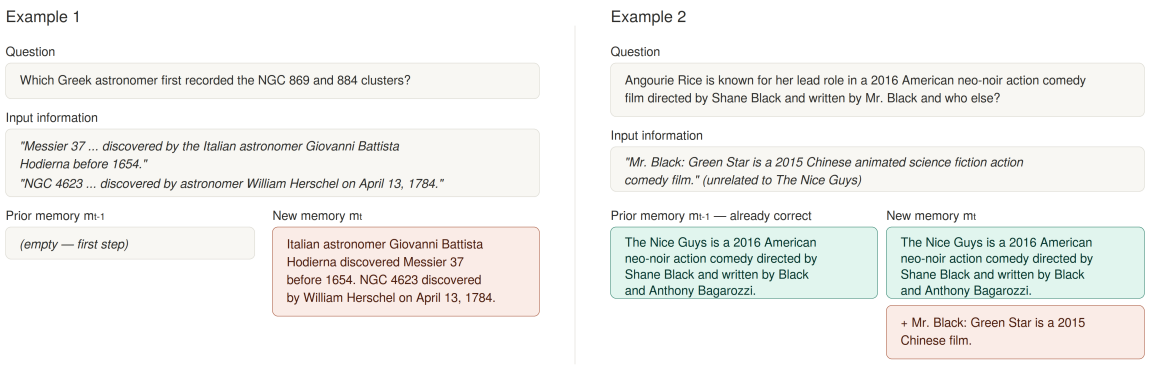

Figure 2: **Examples of hallucination in memory-update steps.** Cold-start hallucination on empty memory (left) and accumulation hallucination when memory already contains the answer (right). Red boxes mark hallucinated content (entities or numerical facts absent from both the input chunk and prior memory); green boxes mark grounded content carried over from m_{t-1}.

ing LLMs (DeepSeek-AI, 2025; Muennighoff et al., 2025), which train models on curated reasoning trajectories. However, memory-agent trajectories scale linearly with the number of memory-update steps and routinely exceed standard SFT context windows by orders of magnitude.

Beyond this scaling issue, we observe that memory agents emit memory states with a high rate of ungrounded content. Our investigation reveals that over half of memory updates on unsupported inputs inject ungrounded entities or numbers (Section 3.2; see Figure 2 for representative cases). A supervised model trained on the raw trajectories inherits this bias, harming performance on tasks where ungrounded content is a liability.

To this end, we introduce a simple RL-free training method called Hallucination-filtered Memory Supervision (HaMS). HaMS first applies hallucination filtering to the source trajectories, removing training examples whose memory updates inject ungrounded content. The filter must operate on individual memory updates rather than entire trajectories: each trajectory mixes grounded and hallucinated updates, so trajectory-level filtering is vacuous, while overwriting hallucinated updates with the prior memory introduces a distinct shortcut-learning failure mode. The filtered examples are produced by stepwise memory supervision, which decomposes each source trajectory into independent training examples that match the inference-time protocol exactly. The decomposition additionally addresses the scaling issue above by bounding per-example length to a single update step.

We evaluate HaMS on multiple long-context QA benchmarks including HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), and five LongBench-QA tasks. Despite requiring over $55\times$ less post-pretraining compute than RL-trained memory agents (Figure 1), HaMS matches or exceeds them on in-distribution and out-of-distribution long-context question answering. These results show that hallucination filtering and stepwise memory supervision recover the performance of RL-trained memory agents at a small fraction of the cost.

2 Preliminary

Long context question answering. We consider long context question answering with a question q and an evidence stream $\mathcal{D}$. The stream can contain millions of tokens, while the evidence needed to answer q is sparse, dispersed across distant locations, and mixed with large amounts of irrelevant content. A successful system must therefore process inputs beyond a single LLM context window, preserve evidence across the stream, and remain robust as distractor content increases. Long context LLMs, retrieval based systems, and memory agents make different trade offs among these requirements. We focus on memory agents because they process the stream sequentially with bounded working memory, allowing explicit cross chunk integration with inference cost that scales linearly in the number of chunks.

Memory agent. A memory agent processes the evidence stream $\mathcal{D}$ as a sequence of segments $s_1, s_2, \ldots, s_T$, where each segment s_t is a contiguous portion of $\mathcal{D}$. At step t, the agent observes the question q, the current segment s_t, and the prior

memory m_{t-1}, then generates an updated memory

$$m_t \sim \pi_\theta(\cdot \mid q, s_t, m_{t-1}), \qquad m_0 = \varnothing. \quad (1)$$

After the final segment, the same model generates an answer from the question and final memory,

$$a \sim \pi_\theta(\cdot \mid q, m_T). \quad (2)$$

The memory m_t is a bounded token sequence intended to serve as a compact task-conditioned representation of the past segments relevant to q. Our goal is to learn the update policy π_θ by supervised fine tuning from source trajectories, while keeping the inference-time protocol fixed.

3 Method

3.1 Motivation

The natural recipe for replacing RL with supervised fine tuning is to sample trajectories from an RL-trained memory agent and fit them with cross entropy, as in long chain-of-thought distillation (DeepSeek-AI, 2025). For memory agents, this recipe faces two obstacles. The first is straightforward: trajectories are too long for standard SFT infrastructure. The second, which we focus on, is that the source trajectories themselves contain memory updates that are not grounded in their input.

Source trajectories contain hallucinated memory updates. Long-context agents that maintain explicit memory emit memory states with a high rate of ungrounded parametric content on inputs without relevant evidence. A substantial fraction of memory updates on unsupported memory-update steps contain named entities or numerical facts absent from both the chunk and the prior memory. A supervised model trained on the raw trajectories inherits this hallucination behavior, harming performance on tasks where ungrounded content is a liability. Prompt-side mitigations do not reduce the rate and can amplify it, indicating that the bias is a property of the trajectory generation process rather than the generation interface.

Trajectories exceed SFT context windows. A source trajectory $\tau = (q, c_1, m_1, \ldots, c_T, m_T, a)$ is a flat concatenation of T per-step blocks of size ≈7K tokens, so its length scales as $L(\tau) = \mathcal{O}(T \cdot 7\text{K})$. In our setting (Section 4), the median training trajectory grows from 40.9K tokens at T=6 to 1.07M at T=180, exceeding standard SFT context windows by one to two orders of magnitude. Shortening trajectories by reducing the distractor count removes precisely the signal of interest: long trajectories are exactly those in which most chunks carry no relevant evidence, and the agent must learn to leave memory unchanged on such chunks at inference time.

3.2 Hallucination in Source Trajectories

We analyze hallucination at the level of individual memory-update steps rather than entire trajectories. This finer granularity is required for filtering hallucinated steps, which discards ungrounded updates while preserving grounded ones from the same trajectory. We use *hallucination* in the narrow sense of *grounding violation*: target content that does not appear in either the new input or the prior memory of the step. This usage is more specific than the standard LLM sense of factually incorrect output.

Defining hallucinated updates. A memory update step $(q, m_{t-1}, c_t) \rightarrow m_t$ is *unsupported step* if the step's input c_t contains no document whose title matches the question's supporting facts under the dataset's ground-truth annotations. For an unsupported step, we identify hallucinated content via a deterministic surface heuristic: named entities (capitalized multi-word sequences) and numerical tokens (digit sequences of length ≥ 2) that appear in m_t but not in c_t or m_{t-1}. The step is flagged as hallucinated if m_t adds at least one named entity or at least one numerical token. We use surface heuristics rather than formal NER to keep the criterion trivially reproducible.

Dominance of hallucination in memory-update steps. Our investigation throughout memory-update trajectories reveals that hallucinated updates dominate memory-agent trajectories. Specifically, 51.4% of unsupported memory updates inject at least three hallucinated entities or a numerical fact, with a mean of 4.45 added entities per step. Across all sampled steps, the source never produced a verbatim copy of the prior memory, showing that the source never emits the no-op output the inference-time protocol requires.

Hallucinated steps resist mitigation. Such hallucinated steps are persistent to polishing. Specifically, explicitly informing the source that the step's input contains no relevant documents and forbidding new facts amplifies the rate by $+6.0$pp, and asking the source to self-correct its prior output

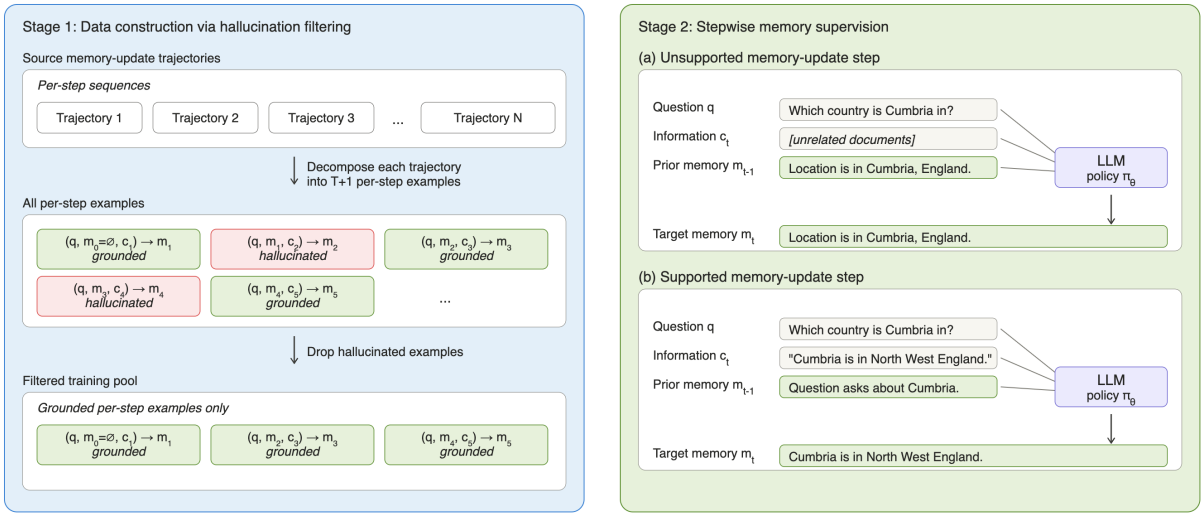

Figure 3: **Overview of HAMS.** The pipeline consists of two stages. **Stage 1** (left) constructs the training pool: source memory-update trajectories are decomposed into per-step examples, and examples whose target memory adds entities or numerical facts absent from both the input information c_t and the prior memory m_{t-1} are dropped as hallucinated (red); the remaining grounded examples (green) form the filtered pool. **Stage 2** (right) trains the policy π_θ on this pool via stepwise supervision over two transition types: unsupported memory-update steps (a), where the input information does not entail any update and the target preserves the prior memory, and supported memory-update steps (b), where the input information provides new evidence that the target memory integrates.

leaves it essentially unchanged. External polishing also fails: regenerating the hallucinated steps with a strong general-purpose reasoning LLM (Claude Sonnet 4.6) and training on the polished trajectories degrades downstream performance rather than improving it. Negative instructions cannot override the source's autoregressive bias, which places high probability on facts related to named entities in the question. We therefore address the bias at the data level, removing hallucinated updates from the training pool.

Addressing hallucination at the data level. We address the issue at the data-level rather than the generation-level. We apply the criterion defined above as a filter, i.e., each memory update step in the training pool is examined, and any step flagged as hallucinated is removed from supervision. Two design choices warrant explanation. First, the filter operates on individual updates rather than on entire trajectories, since each trajectories contains both grounded and hallucinated updates and trajectory-level filtering is vacuous. Second, we remove hallucinated updates rather than replacing them with unsupported steps ($m_t \leftarrow m_{t-1}$), which we found induces a shortcut where the model copies verbatim on every unsupported step including ones with legitimate evidence.

3.3 Step-wise Memory-Update Supervision

To address the trajectory-length obstacle, we supervise at the level of individual memory update steps rather than entire trajectories. Given a source trajectory $\tau = (q, c_1, m_1, \ldots, c_T, m_T, a)$, we extract $T{+}1$ supervised examples: one per memory update step pairing (q, m_{t-1}, c_t) with target m_t (where $m_0 = \varnothing$), plus a final answer step pairing (q, m_T) with a. Each example is presented as a single (user, assistant) message pair under the same prompt template that defines the inference-time protocol of Equations (1) and (2), and contributes a token-level cross-entropy loss on the assistant span:

$$\mathcal{L}_t(\theta) \;=\; -\sum_{k=1}^{|m_t|} \log \pi_\theta\Big(m_t^{(k)} \,\Big|\, m_t^{(<k)},\, q,\, c_t,\, m_{t-1}\Big)\,. \tag{3}$$

The total training loss is $\sum_t \mathcal{L}_t$ over all examples in the filtered pool, with no reward model, rollout, or policy gradient.

The decomposition has two notable properties. First, each example contains exactly one memory update step, so its context length is bounded by the per-step prompt template regardless of source trajectory depth: a trajectory at $T{=}6$ and a trajectory at $T{=}180$ contribute examples of the same size, allowing long trajectories to be admitted under stan-

Table 1: Comparison on needle-in-a-haystack (NIAH) We compare HaMS with Naive SFT on the multi-key Level 2 task of the NIAH benchmark. Hallucination filtering produces a model that generalizes far beyond its training-length regime. On the other hand, Naive SFT collapses between 32K and 64K, while HaMS remains accurate up to 256K.

Length	Naive SFT	HaMS-7B	Δ
8K	100.0	100.0	0.00
16K	96.1	97.7	+1.56
32K	85.2	93.8	+8.59
64K	67.2	87.5	+20.31
128K	54.7	71.1	+16.41
256K	51.6	65.6	+14.06
Average	75.8	85.9	+10.16

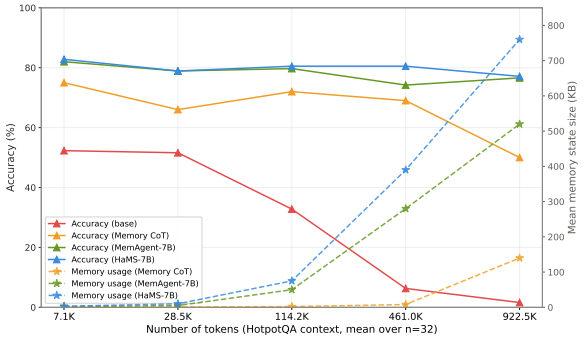

Figure 4: **Accuracy and memory state size as input length increases.** We compare the base model, Memory CoT, MemAgent 7B, and HaMs on HotpotQA. Solid lines show accuracy on the left axis, while dashed lines show mean memory state size on the right axis. HaMS maintains accuracy comparable to MemAgent 7B at long inputs, but uses substantially larger memory states than Memory CoT.

dard SFT infrastructure without truncation. Second, every memory update decision receives direct supervision as its own labeled example, exposing each transition type as its own training signal, in contrast to outcome-reward RL, whose trajectory-level signal collapses these transitions into a single end-of-rollout reward.

4 Experiments

4.1 Setup

Benchmarks. The in-domain benchmark is HotpotQA (Yang et al., 2018) under the chunked long-context evaluation protocol of RULER (Hsieh et al., 2024): each test instance is a question paired with d distractor-padded documents that the model reads chunk by chunk while maintaining a 1,024-token working memory. We use 128 test questions, chunk size 5,000 tokens, and a logarithmic distractor sweep $d \in \{50, 100, 200, 400, 800, 1{,}600, 3{,}200, 6{,}400\}$, corresponding to total input lengths from 7K to 896K tokens. Out-of-distribution generalization is evaluated on five LongBench-QA tasks encompassing 2WikiMultiHopQA, HotpotQA, MuSiQue, NarrativeQA, and Qasper, none of which contribute trajectories to the supervised pool.

Baselines. We compare HaMS against three reference points: the vanilla base model (Qwen2.5-7B-Instruct, no fine-tuning); the paradigm-pure RL system whose trajectories we use to construct HaMS's training data (MemAgent-7B). All three share the same Qwen2.5-7B backbone, isolating the effect of the training procedure. On RULER-HotpotQA, we additionally include long-context LLM baselines that do not maintain a working memory: Qwen2.5-Instruct-7B-1M, Qwen2.5-Instruct-14B-1M, and QwenLong-L1-32B. These read the full evidence stream in a single pass and serve as the natural comparison point for understanding when memory-based agents are necessary.

4.2 Main results

Effectiveness of HaMS. To verify the effectiveness of our method over naive SFT approaches. between Naive SFT and HaMS on NIAH multi-key Level 2 in Table 1. Both systems train on the same pool of source trajectories; HaMS differs only by the hallucination filter applied to the supervised pool. At training-length-equivalent contexts ($\leq$ 32K), both reach above 85%. Beyond this regime, Naive SFT collapses from 85.2 at 32K to 51.6 at 256K, while HaMS retains 65.6, a +14.06pp gap that peaks at +20.31pp at 64K. Both systems were trained without any length-targeted augmentation; the only difference is which steps in the trajectory pool received SFT supervision. We attribute the gap to drift: unfiltered trajectories teach the model to alter memory on unsupported inputs, and this behavior compounds as the number of memory-update steps grows with context length. Removing such steps from the supervised pool breaks this length-coupling.

RULER HotpotQA. Table 2 reports sub-EM accuracy on RULER HotpotQA as the distractor count grows from n=50 (7K tokens) to n=6,400 (896K tokens). Long-context LLM baselines

Table 2: **Quantitative comparison on HotpotQA .** We quantitatively compare our method and baselines as the distractor count n increases. Column headers show n and the corresponding total input length.

	n=50 7K	n=100 14K	n=200 28K	n=400 56K	n=800 112K	n=1600 224K	n=3200 448K	n=6400 896K	Average
Baseline LLMs									
Qwen2.5-Instruct-7B	52.3	57.0	51.6	44.5	32.8	13.3	6.3	1.6	32.4
Qwen2.5-Instruct-14B	75.0	67.2	57.0	54.7	44.5	21.9	10.9	2.3	41.7
Qwen2.5-Instruct-32B	69.5	64.8	60.2	51.6	44.5	21.9	14.1	7.0	41.7
Qwen2.5-Instruct-7B-1M	61.7	56.3	53.9	55.5	51.6	33.6	12.5	0.0	40.6
Qwen2.5-Instruct-14B-1M	60.2	60.9	50.0	57.0	50.0	37.5	8.6	0.0	40.5
QwenLong-L1-32B	72.7	75.0	72.7	60.9	31.3	17.2	13.3	11.7	44.4
Qwen2.5-7B + RAG (K=8)	75.0	75.0	75.8	74.2	74.2	77.3	72.7	68.8	74.1
DS-Distill-Qwen-7B	30.5	12.5	3.1	0.0	0.0	0.8	0.0	0.0	5.9
DS-Distill-Qwen-14B	64.1	64.8	57.0	40.6	14.8	8.6	3.1	6.3	32.4
DS-Distill-Qwen-32B	70.3	66.4	65.6	46.9	23.4	13.3	7.8	7.0	37.6
Memory Agents									
RL-MemAgent-7B	82.0	79.7	**78.9**	77.3	79.7	72.7	74.2	76.6	77.6
HaMS-7B (Ours)	**82.8**	80.5	**78.9**	**78.1**	**80.5**	**77.3**	**80.5**	**77.1**	**79.5**

Table 3: **Out-of-distribution generalization on 2WikiMultiHopQA.** We quantitatively compare our method and baselines as the distractor count n increases. Column headers show n and the corresponding total input length.

	n=50 7K	n=100 14K	n=200 28K	n=400 56K	n=800 112K	n=1600 224K	n=3200 448K	n=6400 896K	Average
Baseline LLMs									
Qwen2.5-Instruct-7B	53.9	49.2	61.1	34.4	28.9	15.6	16.4	17.2	34.6
Qwen2.5-Instruct-14B	57.8	52.3	49.2	34.4	18.8	9.4	17.2	13.3	31.6
Qwen2.5-Instruct-32B	69.5	60.9	60.2	38.3	25.0	20.3	25.0	22.7	40.2
Qwen2.5-Instruct-7B-1M	62.5	59.4	57.8	47.7	46.1	45.3	25.8	0.0	43.1
Qwen2.5-Instruct-14B-1M	58.6	56.3	56.3	49.2	47.7	45.3	34.4	0.0	43.5
QwenLong-L1-32B	74.2	69.5	65.6	58.6	38.3	28.1	24.6	29.9	48.6
DS-Distill-Qwen-7B	36.7	29.7	25.8	0.0	0.8	2.3	2.3	0.8	12.3
DS-Distill-Qwen-14B	71.9	57.8	52.3	42.2	28.1	29.7	28.1	32.0	42.8
DS-Distill-Qwen-32B	69.5	64.1	64.1	36.7	17.2	12.5	9.4	7.0	35.1
Memory Agents									
RL-MemAgent-7B	56.3	**60.9**	52.3	**56.3**	55.5	54.7	**50.0**	**50.0**	**54.5**
HaMS-7B	**58.6**	57.0	**53.1**	53.1	**57.0**	**57.0**	47.7	40.6	53.0

degrade sharply in the high-distractor regime: QwenLong-L1-32B falls from 72.7 at n=50 to 11.7 at n=6,400, and the 1M-token variants (Qwen2.5-Instruct-7B-1M and -14B-1M) collapse to 0.0 at n=6,400 despite their nominal context capacity. Memory agents remain substantially more stable across the sweep. HaMS-7B matches or exceeds RL-MemAgent-7B at every distractor count, with the largest margins at n=1,600 (+4.6pp), n=3,200 (+6.3pp), and an overall average of 79.5 versus 77.6 (+1.9pp). These results show that stepwise supervision on the filtered trajectory pool recovers the distractor-robust memory behavior of RL-trained agents and surpasses it at long input streams, without RL training.

4.2.1 Out-of-distribution distractor sweep on 2WikiMultiHopQA

To test whether the recipe transfers beyond the in-domain HotpotQA distribution, we evaluate on 2WikiMultiHopQA (Ho et al., 2020) under the same distractor sweep protocol as Table 2. Note that no 2WikiMultiHopQA trajectories appear in the supervised pool. HaMS is competitive with the paradigm-pure RL teacher MemAgent (Yu et al., 2026) across the full sweep and exceeds it at four of the eight distractor counts, including the longest stream (n=6,400, ≈900K tokens) where DocQA augmentation yields a +5.3pp gain over the teacher. Naive SFT degrades monotonically as n grows, dropping from 57.0 at n=50 to 36.7 at n=6,400,

Table 4: **Out-of-distribution generalization on LongBench-QA.** We quantitatively compare our method and baselines across five long-context QA tasks. AVG denotes the mean across all five tasks.

Model	2Wiki	HQA	MuSiQue	NarQA	Qasper	Average
Baseline LLMs						
Qwen2.5-Instruct-7B	52.5	59.0	24.0	19.0	19.0	34.7
Qwen2.5-Instruct-14B	67.0	63.5	39.0	20.0	20.5	42.0
Qwen2.5-Instruct-32B	68.5	66.0	37.0	24.5	22.5	43.7
Qwen2.5-Instruct-7B-1M	67.5	58.5	26.0	21.5	24.0	39.5
Qwen2.5-Instruct-14B-1M	70.5	65.5	35.0	22.0	22.0	43.0
QwenLong-L1-32B	83.0	69.5	51.0	**26.0**	24.0	**50.7**
Qwen2.5-7B + RAG (C=1024 K=8)	50.5	59.0	29.5	18.0	25.0	30.3
Qwen2.5-7B + RAG (C=2048 K=8)	50.5	58.0	25.5	21.0	27.0	30.3
DS-Distill-Qwen-7B	47.0	31.5	7.5	4.0	17.5	21.5
DS-Distill-Qwen-14B	**83.5**	67.0	42.0	22.0	22.0	47.3
DS-Distill-Qwen-32B	**83.5**	69.0	47.5	24.0	21.0	49.0
Memory Agents						
RL-MemAgent-7B	80.5	66.5	47.0	14.0	**30.0**	47.6
HaMS-7B	78.5	**70.0**	**50.0**	17.0	26.0	48.3

Table 5: **Distance evidence challenge.** HaMS consistently outperforms MemAgent at both shorter and longer contexts.

Method	28K	112K
MemAgent	40.00	29.10
HaMS	**45.57**	**39.84**

a regime in which both RL-trained and atomic-supervised systems remain stable. The gap between Naive SFT and HaMS widens with stream length, consistent with the length-coverage coupling broken by atomic decomposition. This is the central transfer result: the supervised target is constructed entirely from MemAgent rollouts on HotpotQA and DocQA-RL, yet the resulting model matches or exceeds the RL teacher on a dataset whose trajectories never appear in training. RL trajectories carry a transferable per-step memory-update policy that atomic supervision is able to extract from outcome-level reward signals.

4.3 Analyses

Distance evidence challenge. We compare HaMS with MemAgent-7B on 2WikiMultihopQA with distant-evidence distractors at 200 and 800 documents. As shown in Table 5, HaMS outperforms MemAgent at both context lengths, with a $+5.57$pp gain at 200 documents and a larger $+10.74$pp gain at 800 documents. The widening margin at longer contexts is consistent with our hypothesis that filtering hallucinated updates prevents accumulated drift across many memory-update steps.

Memory usage analysis. Figure 4 compares accuracy and mean memory state size as the HotpotQA input length increases. The base model rapidly loses accuracy as the stream grows, confirming that processing the full context without an explicit memory mechanism is not robust in high distractor settings. Memory based methods are substantially more stable. HaMS follows the accuracy profile of MemAgent 7B across the sweep and remains strong at the longest inputs, while Memory CoT shows a larger drop at the largest context length. This robustness comes with a clear memory cost: memory state of HaMS grows with input length and is much larger than that of Memory CoT. The result suggests that HaMS preserves more task relevant evidence across the stream, trading higher memory usage for stronger long input robustness.

Training-time computational comparison. Table 6 reports per-step training cost for HAMS and the two RL baselines under the same training configuration. HAMS averages 1.3s per step versus 1,247s for MemAgent, a difference of roughly three orders of magnitude. Peak GPU memory drops correspondingly, from 125-131GB to 47GB. The reduction comes from supervised fine-tuning replacing the per-step rollout-generation phase that dominates GRPO wall-clock time; end-to-end, HAMS converges in approximately 98 GPU-hours versus 2,560 for MemAgent-7B, a roughly $26\times$ reduction in total post-pretraining compute.

Table 6: **Training-time computational comparison.** We compared average per-step time, early/late step latency, and peak GPU memory usage for efficiency comparison.

Method	Avg. Time / Step (s)	Step 1 (s)	Step 10 (s)	Step 100 (s)	Peak Memory Usage (GB)
MemAgent	1247.2	1278.9	1366.9	1377.3	125.0
HaMS (Ours)	1.3	2.7	1.4	1.2	47.3

5 Related Work

RL-trained memory agents. A recent line of work trains LLM agents to maintain bounded working memory under outcome-grounded RL rewards. MemAgent (Yu et al., 2026) introduces a working-memory overwrite paradigm in which each chunk fully rewrites the memory state. ReMemR1 (Shi et al., 2026) augments this with retrieval bolt-ons via `<recall>` tags, while Memory-R1 (Yan et al., 2025), MEM1 (Zhou et al., 2025), and Mem-α (Wang et al., 2025b) explore alternative memory representations and reward designs. These systems share two properties relevant to our setting: training cost dominates inference cost by orders of magnitude, and they vary in whether trajectories follow a single operational paradigm or interleave qualitatively different operations within one rollout. We use MemAgent's public checkpoint as our source agent throughout the paper.

SFT distillation from RL trajectories. The Superficial Alignment Hypothesis (Zhou et al., 2023) and the DeepSeek-R1 distillation line (DeepSeek-AI, 2025) establish that supervised fine-tuning on traces from RL-trained models can recover much of the source model's capability in instruction following and chain-of-thought reasoning. The MemAgent authors report a small SFT-only ablation (Yu et al., 2026, App. F) that falls short of the RL system; under our hallucination filter and stepwise decomposition, we recover and exceed the source agent on in-distribution evaluation and match it on out-of-distribution evaluation. Unlike reasoning traces, which are flat token sequences, memory trajectories are sequences of structured transitions, and we treat this transition structure rather than the token stream as the unit of supervision.

Compute-efficient agents and retrieval baselines. Cost-aware RL methods (Wu et al., 2025; Wang et al., 2025a; Qian et al., 2025) reduce training cost by adding cost terms to the reward signal, and the efficient agents survey (Yang et al., 2026) catalogs efficiency metrics across tokens, time, and GPU-hours. Our approach is orthogonal: we eliminate RL entirely rather than make it cheaper. A separate line replaces bounded working memory with retrieval over the evidence stream; we include RAG with dense retrieval as a baseline in Section 4 to make the memory-vs-retrieval comparison explicit at matched distractor density.

6 Conclusion

This work asked whether the three defining properties of an RL-trained memory agent—handling of effectively unbounded inputs, stable scaling with distractor density, and linear-time decoding—are tied to the RL training regime. In this setting the answer is no: each is a property of the operational protocol, and all three transfer to a supervised model when the training data are paradigm-coherent and cover the chunk-stream lengths the model will see at test time. The resulting model matches or exceeds the source RL system at the long-distractor regime on HotpotQA at a small fraction of the post-pretraining compute, and surpasses retrieval-based baselines across the density sweep. A paradigm-coherence ablation, a chunked-versus-single-context probe, and a per-chunk failure-mode taxonomy attribute the residual gap at moderate density to evaluation protocol and data coverage rather than to the use of supervision. We therefore view paradigm-coherent supervised fine-tuning on curated, mixed-distractor RL trajectories as a practical alternative to RL for memory-augmented agents in this regime, and dual substring-EM reporting with explicit per-step input-cap disclosure as standard practice for any future comparison among open-weight memory agents.

Limitations

Our experiments are restricted to a single 7B-parameter base model (Qwen2.5-7B-Instruct). Whether the gains transfer at larger scales (14B, 32B) is unverified.

7 Ethical Considerations

We used AI assistants during the preparation of this work. ChatGPT was used for writing support, in-

cluding language polishing and drafting assistance, and Claude Code was used as a coding assistant during implementation. All experimental design, analyses, claims, and final manuscript content were reviewed and verified by the authors.

References

DeepSeek-AI. 2025. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. 2024. Longrope: Extending llm context window beyond 2 million tokens. *Preprint*, arXiv:2402.13753.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*.

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

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *Preprint*, arXiv:2501.19393.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2024. YaRN: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations*.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. 2025. ToolRL: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*.

Ning Shang, Li Lyna Zhang, Siyuan Wang, Gaokai Zhang, Gilsinia Lopez, Fan Yang, Weizhu Chen, and Mao Yang. 2025. Longrope2: Near-lossless llm context window scaling. *Preprint*, arXiv:2502.20082.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*. Source paper for GRPO.

Yaorui Shi, Yuxin Chen, Siyuan Wang, Sihang Li, Hengxing Cai, Qi Gu, Xiang Wang, and An Zhang. 2026. Look back to reason forward: Revisitable memory for long-context LLM agents. In *ICLR 2026*. ReMemR1; OpenReview forum: 1cymflI2Lh.

Fanqi Wan, Weizhou Shen, Shengyi Liao, Yingcheng Shi, Chenliang Li, Ziyi Yang, Ji Zhang, Fei Huang, Jingren Zhou, and Ming Yan. 2025. Qwenlong-l1: Towards long-context large reasoning models with reinforcement learning. *arXiv preprint arXiv:2505.17667*.

Hongru Wang, Cheng Qian, Wanjun Zhong, Xiusi Chen, Jiahao Qiu, Shijue Huang, Bowen Jin, Mengdi Wang, Kam-Fai Wong, and Heng Ji. 2025a. OTC: Optimal tool calls via reinforcement learning. *arXiv preprint arXiv:2504.14870*. Method name: OTC-PO.

Yu Wang, Ryuichi Takanobu, Zhiqi Liang, Yuzhen Mao, Yuanzhe Hu, Julian McAuley, and Xiaojian Wu. 2025b. Mem-α: Learning memory construction via reinforcement learning. In *Submitted to ICLR 2026*. OpenReview forum: dm42omwep1.

Feijie Wu, Weiwu Zhu, Yuxiang Zhang, Soumya Chatterjee, Jiarong Zhu, Fan Mo, Rodin Luo, and Jing Gao. 2025. PORTool: Tool-use LLM training with rewarded tree. *arXiv preprint arXiv:2510.26020*.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Hinrich Schütze, Volker Tresp, and Yunpu Ma. 2025. Memory-R1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *arXiv preprint arXiv:2508.19828*.

An Yang, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoyan Huang, Jiandong Jiang, Jianhong Tu, Jianwei Zhang, Jingren Zhou, Junyang Lin, Kai Dang, Kexin Yang, Le Yu, Mei Li, Minmin Sun, Qin Zhu, Rui Men, Tao He, and 9 others. 2025. Qwen2.5-1m technical report. *arXiv preprint arXiv:2501.15383*.

Xiaofang Yang, Lijun Li, Heng Zhou, Tong Zhu, Xiaoye Qu, Yuchen Fan, Qianshan Wei, Rui Ye, Li Kang, Yiran Qin, Zhiqiang Kou, Daizong Liu, Qi Li, Ning Ding, Siheng Chen, and Jing Shao. 2026. Toward efficient agents: Memory, tool learning, and planning. *arXiv preprint arXiv:2601.14192*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, and Hao Zhou. 2026. MemAgent: Reshaping long-context LLM with multi-conv RL-based memory agent. In *ICLR 2026*. OpenReview forum: k5nIOvYGCL.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Li, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, and 16

others. 2025. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023. LIMA: Less is more for alignment. In *Advances in Neural Information Processing Systems 36*.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. 2025. MEM1: Learning to synergize memory and reasoning for efficient long-horizon agents. *arXiv preprint arXiv:2506.15841*.