MEMGYM: A LONG-HORIZON MEMORY ENVIRONMENT FOR LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon LLM agents must keep deciding which observations and tool outputs to hold in context. Most memory benchmarks test recall in multi-turn chat. Agent benchmarks where these decisions arise report only end-task success, which mixes memory failures with reasoning and tool-use failures. We present MEMGYM, an agentic memory benchmark that inserts a memory module between the interaction history and the reasoning model. Its six tracks range from tool-use dialogue and deep-research search to coding and computer use. Each track supports controlled comparisons of memory modules against a track-specific no-memory reference under a fixed environment, scaffold, and reasoner. Four tracks wrap agent benchmarks where forgotten information is often recoverable from the environment. We therefore construct MEMGYM-CODEQA and MEMGYM-DR at controlled context lengths with construction-time checks of fact recoverability and answer support. To make SWE-Gym comparisons cheaper, we train MEMRM, a 1.7B-parameter scorer that predicts whether the next action after a compression returns the recorded reference output. This local check costs one reasoner step and one classifier call. Full rollouts still measure end-task outcome. We evaluate eight memory strategies with four reasoning models. A rolling summary does not improve SWE-Gym performance. Both summary strategies yield higher mean scores on tool-use dialogue, and a structured summary does so on web tasks. ProgramBench episodes often outgrow the context window. A structured summary raises their mean fraction of hidden tests passed from 9% to 49% under the same step budget. We release MEMGYM and MEMRM with the labeled trajectory corpus and pipeline code at https://anonymous.4open.science/r/MemGym-003C.

1 INTRODUCTION

A long-horizon LLM agent accumulates observations, tool outputs, and intermediate conclusions faster than its context can hold them. It must keep deciding what to preserve, summarize, or evict. Coding agents return to earlier debugging evidence (Jimenez et al., 2024; Pan et al., 2024), and research agents carry bridge facts across retrieval turns (Ho et al., 2020; Yang et al., 2018). Dialogue agents retain user constraints and tool state (Yao et al., 2024; Barres et al., 2025), and web agents track which items a batch operation has already handled. Long-context benchmarks (Bai et al., 2025; Hsieh et al., 2024) test recall over a fixed input. An agent instead decides what it will see later, long before the consequences become visible. Current benchmarks rarely measure these decisions directly. Memory benchmarks (Maharana et al., 2024; Wu et al., 2024; Hu et al., 2025; Ai et al., 2025) mostly test recall over long conversations rather than memory inside an agent that acts in an environment. Agent benchmarks such as SWE-Gym, τ^2-bench, and WebArena do contain these decisions but report only end-task success. A memory failure then looks like a failure of reasoning or tool use.

MEMGYM makes memory an explicit intervention by placing a memory module before every model call. The module receives the accumulated history and returns the context the reasoner sees, much like the OpenHands condenser (Wang et al., 2024a). Each track keeps the environment, scaffold, and tools fixed for a given reasoner while the memory module varies. The score difference from a track-specific no-memory reference then measures the effect of one memory configuration in that setting. We interpret these differences within each track rather than across tracks. Four tracks wrap agent benchmarks in coding (SWE-Gym and ProgramBench), tool-use dialogue (τ^2-bench), and

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.

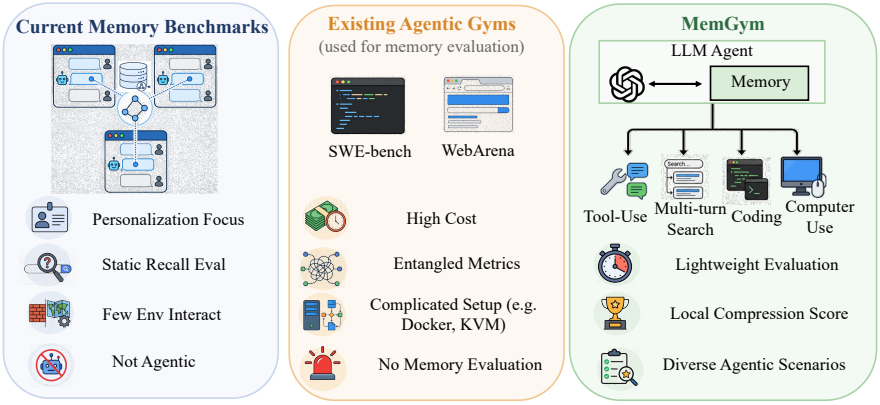


Figure 1: Overview of MEMGYM. Memory benchmarks (left) test static recall in chat without an environment, and agentic gyms (middle) entangle memory with reasoning and tool use in costly end-task scores. MEMGYM (right) inserts one memory module before the reasoning model on six tracks in four regimes. Compared runs hold the environment and reasoner fixed while the module varies. MEMRM provides a lightweight event-level check.

computer use (WebArena-Infinity). An agent in these tracks can often recover forgotten information from files, web pages, or parametric knowledge. Such recovery softens the consequences of forgetting. We therefore construct two tracks that place more direct pressure on retention. MEMGYM-CODEQA targets debugging observations that its construction checks do not recover from the repository. MEMGYM-DR places bridge facts early in a research session and fictionalizes entities to reduce direct answering from parametric knowledge.

The remaining obstacle is cost. A paired SWE-Gym comparison needs two full rollouts at about $2 each. To reduce this cost, we train MEMRM on the compressions recorded at the boundary. It predicts whether the agent's next action after a compression returns the output and exit status of the recorded reference action. This check is local, so full rollouts remain the measure of end-task outcome. Our evaluation pairs each track with the strategy family that matches how it uses memory (Table 4). Compaction suits the wrapped tracks because the agent keeps acting on its own history. The constructed tracks instead ingest a fixed session and then query it, which suits external memory. Truncation serves as a recency reference on MEMGYM-CODEQA. The results differ across tracks. A rolling summary does not improve SWE-Gym performance. Both summary strategies yield higher mean scores on tool-use dialogue, and a structured summary does so on web tasks. The leader among the three memory systems evaluated on both constructed tracks differs between MEMGYM-CODEQA and the deepest MEMGYM-DR chains. We make four contributions.

- End-task scores mix memory, reasoning, and tool-use failures. A prompt-level memory boundary makes memory an explicit intervention on six tracks in four regimes while the environment, scaffold, and reasoner stay fixed. SWE-Gym memory runs replay the baseline until the first compaction.
- Wrapped gyms often let agents recover forgotten information. We thus build MEMGYM-CODEQA (4,289 instances from 32 repositories) and MEMGYM-DR (1,194 fictionalized instances) at controlled context lengths, with construction-time checks of fact recoverability and answer support.
- Paired full rollouts are costly. MEMRM, a 1.7B-parameter scorer, predicts reference-output agreement per compression event with a 0.6-second classifier call. It reaches 0.985 AUROC on 3,007 held-out examples, and its first-compression score predicts loss of baseline success (AUROC 0.81).
- Memory systems are usually tested in one domain. We evaluate eight strategies with four reasoners on six tracks. Structured summarization raises the mean ProgramBench hidden-test pass rate from 9.3% to 48.7%, and the leading MEMGYM-CODEQA strategies close most of the gap to full context.

2 RELATED WORK

Early memory-augmented LLM systems such as MemoryBank (Zhong et al., 2024), MemGPT (Packer et al., 2023), and ReadAgent (Lee et al., 2024) added explicit memory components but were evaluated

Table 1: Comparison of memory and agent benchmarks. *#Scn* is the number of scenarios. *Interactive* means the environment responds to actions. *Controlled Comparison* means compared runs hold the environment and reasoner fixed while the memory component varies. *Min. Cost* is the lowest evaluation cost. *Train. Data* indicates released trajectories. †Four tracks are interactive and two are offline. ‡MEMRM evaluates single events cheaply. §Within each track.

Benchmark	Agentic Scenarios	# Scn	Inter-active	Controlled Comparison	Min. Cost	Train. Data	Length
Dialogue-Centric Memory Benchmarks							
LoCoMo (Maharana et al., 2024)	Long-term dialogue	1	✗	✗	Low	✗	9K
LongMemEval (Wu et al., 2024)	Long-term dialogue	1	✗	Partial	Medium	✗	115K
MemoryAgentBench (Hu et al., 2025)	Multi-turn dialogue	1	✗	Partial	Medium	✗	100K–300K
MemoryBench (Ai et al., 2025)	Continual dialogue	1	✗	✗	Medium	✗	30K–380K
Long-Horizon Agent Benchmarks							
SWE-Gym (Pan et al., 2024)	Repository coding	1	✓	✗	High	✓	Task-dep.
ProgramBench (Yang et al., 2026)	Repository-scale program reconstruction	1	✓	✗	High	✗	Task-dep.
τ^2-bench (Barres et al., 2025)	Tool-agent-user dialogue	1	✓	✗	Medium	✗	Task-dep.
WebArena-Infinity (Zhou, 2026)	Web computer use	1	✓	✗	Medium	✗	Configurable
Agent-Centric Memory Benchmarks							
AMA-Bench (Zhao et al., 2026)	Agentic apps (post-hoc QA)	1	✗	✗	Medium	✗	57K
AMemGym (Cheng et al., 2026)	Personalized conversation	1	✓	✗	Medium	✗	Configurable
MemoryArena (He et al., 2026)	Multi-session agentic tasks	4	✓	✗	Medium	✗	Multi-session
MEMGYM (Ours)	Coding, program reconstruction, web, tool dialogue, deep-research search, coding QA	**6**	✓†	✓§	Track-dep.‡	✓	**Configurable**

on long-term dialogue or document understanding rather than on memory formed while interacting with an environment. A-Mem (Xu et al., 2025) introduces agentic note evolution and is evaluated on the conversational-memory benchmark LoCoMo (Maharana et al., 2024). Its questions concern personas and temporal event graphs rather than state accumulated while debugging code, using tools, or navigating websites. LongMemEval (Wu et al., 2024), MemoryAgentBench (Hu et al., 2025), and MemoryBench (Ai et al., 2025) extend this line with longer histories, incremental multi-turn ingestion, and continual-learning feedback. The object to be remembered is still a transcript or feedback stream rather than a live trajectory in which the agent's own actions shape what it later needs. AMA-Bench (Zhao et al., 2026) and AMemGym (Cheng et al., 2026) come closer to our setting. AMA-Bench asks post-hoc questions about agentic trajectories, and AMemGym runs on-policy conversations whose latent user state evolves over time. MemoryArena (He et al., 2026) evaluates memory across interdependent multi-session agentic tasks. These benchmarks score a memory system by the answers or task outcomes it produces. MEMGYM instead places the module inside existing interactive environments and reports its effect as a difference under a fixed reasoner. Its constructed tracks also check fact recoverability and answer support at construction time.

Long-horizon agent benchmarks ask whether agents can complete extended tasks in executable environments. SWE-bench (Jimenez et al., 2024) tests the resolution of real GitHub issues, and SWE-Gym (Pan et al., 2024) adds executable training tasks with unit tests and released trajectories. Full evaluation is expensive and yields only a sparse resolve rate, too slow a signal for iterating on memory designs. Lindenbauer et al. (2025) find that simply masking old observations matches LLM summarization for a fixed SWE-bench agent, which agrees with our SWE-Gym result. ProgramBench (Yang et al., 2026) asks agents to rebuild a program from its executable and documentation and scores the result with hidden behavioral tests, producing episodes whose context often exceeds a 200K-token window. τ-bench and τ^2-bench (Yao et al., 2024; Barres et al., 2025) evaluate tool-agent-user workflows but report end-task success without separating failures of memory, policy, and tool use. WebArena (Zhou et al., 2023) provides functional web environments, WebArena-Infinity (Zhou, 2026) scales it by generating self-contained applications with verifiable tasks, and OSWorld and OSGym (Xie et al., 2024; Qin et al., 2025) extend the same line to desktop and operating-system tasks. Memory matters in all of these benchmarks, but none treats it as a controlled variable. MEMGYM wraps such environments with an explicit memory boundary and compares memory strategies by their paired gains under a fixed reasoner and scaffold.

3 THE MEMGYM EVALUATION FRAMEWORK

End-task scores reflect the combined effects of memory, reasoning, and tool use. MEMGYM makes the memory intervention explicit by holding the environment, scaffold, tools, and reasoner fixed within each comparison. The wrapped tracks measure the effect of context management on interactive

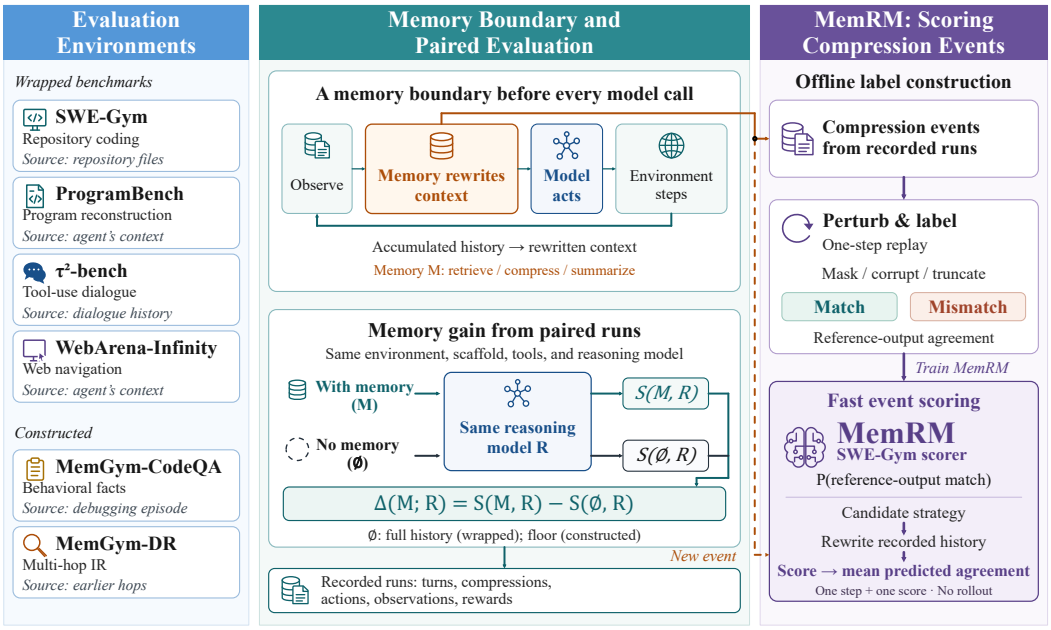


Figure 2: MEMGYM overview. The six tracks (left) comprise four wrapped from existing benchmarks and two constructed ones. A baseline run and a memory run (middle) share the environment, scaffold, tools, and reasoner. The memory module rewrites the context before each reasoner call, and the memory gain is the score difference between the two runs. Recorded compression events (right) are perturbed and labeled by whether the next action's output and exit status match the recorded reference. MEMRM is trained on them and scores new events without rolling the episode forward.

task outcomes. The constructed tracks target retention more directly by restricting access to earlier information and varying the memory-context budget or retention span.

3.1 WHERE THE MEMORY MODULE HAS TO SIT

The memory module receives the accumulated interaction history immediately before every model call and returns the context the reasoning model sees. The environment, scaffold, tools, and reasoning model are the same with and without it (Figure 2). This interface is defined on the prompt rather than the domain, as in the OpenHands condenser (Wang et al., 2024a), so the same strategy applies to every track where it fits (Appendix D). We report the *memory gain* of a memory module M under reasoning model R and track score S

$$\Delta(M; R) \;=\; S(M, R) \;-\; S(\varnothing, R). \tag{1}$$

The reference condition $\varnothing$ is track-specific. The wrapped-track reference retains the unmanaged interaction history subject to the model's input limit. The constructed-track reference receives the current prompt and the specified task metadata but no retained session context. The MEMGYM-CODEQA metadata consists of the task description and a repository overview. The wrapped-track contrast measures context management relative to unmanaged history. The constructed-track contrast measures the benefit of access to retained session information. These contrasts are interpreted within their respective tracks, not as a common scale across tracks. The difference holds the reasoner's competence and the task's difficulty constant but does not remove the reasoner. The module changes the context, the context shifts the actions, and the actions change the environment. Δ therefore measures a module as used by a particular reasoner, so every gain is reported with its reasoner.

The same interaction also means that two independent rollouts of a task can diverge long before the module first acts, and their difference would then mix the effect of memory with sampling noise. We therefore start the memory run from a recorded no-memory trajectory where a track allows it. On SWE-Gym the memory run re-executes the recorded actions up to the first point at which the module rewrites the context and continues live from there. Both runs thus reach that point in the same environment state. An episode that never reaches the compaction trigger is identical under both

Table 2: The six MEMGYM tracks. *Information sources* lists representative sources of task information, and *Recovery* summarizes the corresponding recovery routes. Recovery entries for the constructed tracks refer to the construction checks in Appendix C.

Track	Information sources	Recovery	∅	Metric	Size
SWE-Gym	Repository files	Re-read files (cheap)	Full history	Resolve rate	1,054
ProgramBench	Agent's context	Re-probe binary (costly)	Full history*	Tests passed	200†
τ^2-bench	Dialogue history	Re-ask or re-query	Full history	Task success	288
WebArena-Infinity	Agent's context	Not shown on pages	Full history	Task success	140
MEMGYM-CODEQA	Debugging episode	Construction checks	None (ref.)	Accuracy	4,289
MEMGYM-DR	Earlier hops	Construction checks	None (ref.)	Judge score	1,194

*No context management. The history is bounded by the 200K-token window less an output reserve (154,944 input tokens). †Benchmark size. We evaluate 53 tasks sampled at random.

conditions and keeps its baseline outcome. We inject the first half of a recorded WebArena-Infinity trajectory as a fixed prefix, and a no-memory run after the same prefix serves as the control. On τ^2-bench and ProgramBench both conditions run from the start of the episode, and both τ^2-bench strategies are compared with the same no-memory runs of the same tasks. The constructed tracks need no rollout at all. The module ingests a fixed session and is queried only after the turns carrying the target facts have left the context (Section 3.2).

3.2 CONSTRUCTING TRACKS THAT TARGET RETENTION

The constructed tracks target retention by restricting access to earlier information and applying checks of fact recoverability and answer support. Table 2 summarizes representative information sources and recovery routes across the six tracks. ProgramBench (Yang et al., 2026) provides a natural source of context pressure among the wrapped benchmarks. The agent rebuilds a program from its compiled executable, and what it learns by probing the binary can be recovered only by probing again at the cost of further steps. The median episode among the 192 trajectories released with the official Claude Sonnet 4.6 run reaches about 264K tokens of full conversation. This exceeds a 200K window, so the context window alone forces memory management. We therefore compare a full-history run, which stops once the next request no longer fits, with a structured summary that lets the episode continue.

Both constructed tracks are offline. The memory module ingests a fixed session and is then queried. Construction-time checks assess fact recoverability and answer support under the specified reference inputs. MEMGYM-CODEQA starts from SWE-smith (Yang et al., 2025) bugs and targets information observed during debugging. The pipeline retains an instance when at least two facts are not recovered by its repository and gold-patch checks. It checks the associated QA pairs for solvability from the retained facts, distractor robustness, and answer leakage. The recovery criterion applies at the instance level rather than establishing memory dependence for every scored QA pair.

MEMGYM-DR has no repository to hide facts from, so it makes them scarce by distance. Instances are grown as chains of bridge facts through iterative search over academic sources (Kinney et al., 2023; Priem et al., 2022). The first three hops of a four-hop instance are followed by three, two, and one later hop. The earliest facts thus need the longest retention. Frontier models in a pilot reached 0.70–0.85 with no memory by answering from pretraining. We therefore replace entities consistently with invented ones. The no-memory reference then scores 0.33, 0.29, and 0.009 at 3, 4, and 5/6 hops. Figure 3 shows a condensed instance. Construction-time filtering retains instances on which curated notes score at least as highly as the full context of the same 100K-token corpus. MEMGYM-DR therefore evaluates strategies on this selected distribution, rather than providing an unrestricted comparison with long-context agents. Both pipelines add distractors intended to reduce shortcuts and scale instances to a target context length (Appendix C).

3.3 A LEARNED SCORER FOR COMPRESSION EVENTS

Each side of a paired comparison is a full episode. A SWE-Gym episode without memory already costs about $2.10 in input tokens at Claude Sonnet 4.5 list prices. This cost multiplies with every strategy, seed, and environment added (Section B.1). A far cheaper local signal is available at each

Question. Why have codec-trained detectors become the best approach for identifying the newest synthetic voice forgeries, and what advantage do they keep despite the change in how forgeries are generated?

Hop 1 (needed for the final answer). Chromspec and logfreq features outperform melfreq features for voice-forgery verification, by 37% in balanced error on average.
Hop 2. Codec-trained detectors reach 0% balanced error on most test conditions for the newest forgeries.
Hop 3. Codec-trained detectors lower average balanced error by 41.4% relative to vocoder-trained detectors on the AudioShield test set.
Hop 4. The newest forgeries are generated directly from discrete neural codecs and skip the final vocoder step, which changes the artifacts a detector must find.

Answer. Codec-trained detectors match how the newest forgeries are produced (hop 4), cut average error by 41.4% (hop 3), and draw on chromspec and logfreq features that beat melfreq features (hop 1).

Construction-time verifier scores are 0.1 (no memory), 0.6 (long-context dump), and 0.85 (curated notes).

Figure 3: A four-hop MEMGYM-DR instance, condensed from the released data (instance 55899). Entity and dataset names are the invented replacements. The hop-1 fact is absent from the question, yet the full answer needs it three hops later.

compression, namely whether the strategy changes what the agent does next. MEMRM reads a compression event and predicts the probability that the agent's next action after the compression returns the same output as the action recorded at that point in the original episode. It does not observe the execution results used to determine the label.

A candidate strategy rewrites the recorded history at each compression point, and the reasoner proposes its next action from that view (Figure 2, right). MEMRM estimates the probability that the action's output and exit status match the recorded reference. Averaging these probabilities summarizes predicted agreement on the recorded event set, not the task success rate of the deployed strategy. Each evaluation requires one reasoner call and one scorer query. Full rollouts remain necessary to measure end-task outcomes. We have not evaluated whether event-level averages reproduce strategy-level rollout rankings. Section 4.4 tests these predictions and their link to episode outcomes.

Training labels are obtained through execution. Each example applies one of six perturbations that mask, corrupt, or truncate the content of a recorded SWE-Gym compression event. We restore the environment to that point and execute the next action proposed from the perturbed view. An example is labeled a *match* when the normalized output and exit status agree with the recorded reference, and a *mismatch* otherwise, including when no executable action is produced. This is a local behavioral proxy. A different valid action may disagree with the reference, and matching outputs need not imply equivalent changes to the environment or preservation of eventual task success. We train MEMRM with a class-weighted cross-entropy because mismatches dominate,

$$\mathcal{L}_{\text{MEMRM}}(\theta) \;=\; -\,\mathbb{E}_{(c,y)\sim\mathcal{D}_{\text{aug}}}\big[\, w_y \,\log p_\theta\big(y \mid c\big)\,\big], \qquad y \in \{\text{match}, \text{mismatch}\}, \tag{2}$$

where c is a compression event from the augmented set $\mathcal{D}_{\text{aug}}$ and w_y is the class weight (Appendix E).

4 EVALUATING MEMORY STRATEGIES ACROSS TRACKS

We evaluate eight memory strategies with four reasoning models across six tracks, and Table 4 lists the evaluated combinations. We first measure the effect of the tested compaction strategies within each wrapped track (Section 4.2). We then compare configurations across memory-context budgets on MEMGYM-CODEQA and hop groups on MEMGYM-DR (Section 4.3). Finally, we evaluate MEMRM's held-out event predictions and their association with episode outcomes (Section 4.4).

4.1 EXPERIMENTAL SETUP

The strategies we compare cover the two ways an agent can keep information beyond its recent turns. Compaction strategies rewrite the prompt once the history reaches a trigger length. A rolling summary in the style of the OpenHands condenser (Wang et al., 2024a) replaces the middle of the history with a free-form account of the actions, findings, and decisions it contained. A structured summary instead fills a fixed environment-specific schema (the objective, test results, and rejected approaches in coding) so that state the environment does not display is kept explicitly. Truncation

keeps only the latest messages and thus selects by recency rather than relevance. External-memory strategies store what the agent has seen and retrieve from it when a question arrives, and they differ in how they judge what an earlier observation contributes to a later need. BM25 (Robertson & Zaragoza, 2009) and Naive RAG (Lewis et al., 2020) score stored passages one at a time by lexical or embedding similarity. A-Mem (Xu et al., 2025) links LLM-written notes so that a passage with little direct relevance can be reached through a related one. MemoryBank (Zhong et al., 2024) and LightMem (Fang et al., 2025) consolidate memories over time. The boundary accepts either family on any track, and on each track we evaluate the family that matches how the track uses memory. Compaction runs on the wrapped gyms because the agent keeps acting on its own history. External memory runs on the constructed tracks because they ingest a fixed session and then query it, and MEMGYM-CODEQA adds truncation as a recency reference. We compare strategies within each track and use the collection of tracks to characterize their behavior under different task, reasoner, and context settings. Each strategy runs as released by its authors, and on MEMGYM-CODEQA all share a common memory-context budget, the pressure factor under study there (Appendix D).

On SWE-Gym we use Claude Sonnet 4.5 as the reasoner, with Claude Haiku 4.5 and GPT-OSS-120B as cross-reasoner controls, over 1,003 to 1,041 SWE-bench-style instances per reasoner. SWE-Gym runs are paired by replay as described in Section 3.1. Haiku 4.5 is the reasoner on τ^2-bench (the 288-task base split across the mock, telecom, airline, and retail domains) and on WebArena-Infinity (140 tasks from the benchmark's hard split in the Gmail, PayPal, and GitLab applications under a fixed-prefix protocol). We compare the rolling summary with the no-memory reference, and on τ^2-bench and WebArena-Infinity we add the structured summary. We set compaction triggers per track (100 messages on SWE-Gym, 10 on τ^2-bench, and 20 on WebArena-Infinity) and keep them identical for every strategy within a comparison. Haiku 4.5 is the summarizer of both summary strategies. On ProgramBench we run Claude Sonnet 4.6 on 53 randomly sampled tasks in the official harness. We compare the agent without context management with a structured summary that Haiku 4.5 writes into a bounded state replacing the oldest turns. The unmanaged history is bounded by an input limit of 154,944 tokens (the 200K window less room for output). We give both conditions the same step budget rather than the same dollar budget because compaction invalidates prompt caching and a dollar cap would cut memory runs short. We score a task by the fraction of its hidden tests that pass and average over tasks.

Sonnet 4.5 is the reasoner on both constructed tracks, and Claude Haiku 4.5 is the judge on MEMGYM-DR. On both tracks we compare A-Mem, LightMem, MemoryBank, and a no-memory *None* control. On MEMGYM-CODEQA these strategies and truncation must fit what they pass to the reasoner into a memory-context budget of 64, 128, 256, or 512 tokens. The no-memory control still sees the task description and the repository structure, and a full-context reference sees the entire episode. The two serve as reference points for what a memory can recover. We evaluate strategies on 670 of the 4,289 released instances (2,324 QA pairs) drawn by a seeded hash of the instance identifiers without regard to any result. A QA pair counts as correct when a keyword scorer finds at least half of its key facts in the answer (Section B.1). On MEMGYM-DR we add the two retrievers and vary the hop count over 3-, 4-, and 5/6-hop questions on a fictionalized 100K-token corpus. Browser settings and per-track configuration are in Section B.1.

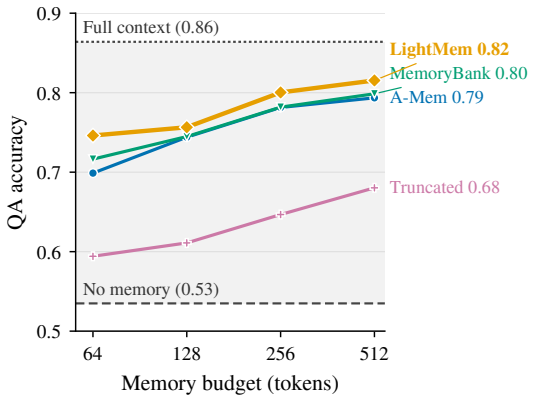
4.2 MEMORY ACROSS FOUR WRAPPED GYMS

On SWE-Gym (Table 3) the rolling summary leaves the resolve rate of Sonnet 4.5 unchanged and lowers it by 1.0 point for Haiku 4.5 and 3.2 points for GPT-OSS-120B. The drops may reflect reasoners that work less well from compressed context or, for GPT-OSS-120B, from a summary written by a different model. Possible explanations include access to information in repository files, the fact that only 55–65% of instances reach the compaction trigger, and the modest compression ratios of 1.32–1.47 (Section B.3). The present comparison does not distinguish these explanations. On ProgramBench the structured summary raises the mean fraction of hidden tests passed from 9.3% to 48.7% across three seeds. The unmanaged-history reference terminates when its next request exceeds the input limit, whereas compaction allows episodes to continue. No other compaction strategy is evaluated on this track, so the result supports context management in this setting rather than the superiority of structured summaries over other ways of keeping an episode running.

Mean success on τ^2-bench across three seeds rises from 53.5% without memory to 61.2% with the rolling summary and 56.8% with the structured summary, both compared with the same reference

Table 3: Task success in percent without memory (Baseline) and with memory (With mem.) on four wrapped gyms. Success is the resolve rate on SWE-Gym, the fraction of hidden tests passed on ProgramBench, and the success rate on τ^2-bench and WebArena-Infinity. Δ is the difference in points and n the number of tasks. *Compr.* is the compression ratio, tokens before compaction divided by tokens after. We report the mean and standard deviation across 3 independent seeds on all four gyms. Δ is computed before rounding.

Gym	Model	Strategy	n	Baseline	With mem.	Δ	Compr.
SWE-Gym	Sonnet 4.5	Summary	1041	42.8±1.2	42.8±1.5	0.0	1.47×
SWE-Gym	Haiku 4.5	Summary	1003	44.0±1.4	43.0±1.6	−1.0	1.32×
SWE-Gym	GPT-OSS-120B	Summary	1003	22.3±1.8	19.1±1.5	−3.2	1.45×
ProgramBench	Sonnet 4.6	Structured	53	9.3±0.9	48.7±2.4	+39.3	3.08×
τ^2-bench	Haiku 4.5	Summary	288	53.5±3.1	61.2±2.5	+7.7	2.29×
τ^2-bench	Haiku 4.5	Structured	288	53.5±3.1	56.8±2.8	+3.3	1.86×
WebArena-Infinity	Haiku 4.5	Structured	140	34.3±2.1	38.6±2.5	+4.3	1.37×
WebArena-Infinity	Haiku 4.5	Summary	140	34.3±2.1	35.0±2.2	+0.7	1.45×



(a) MEMGYM-CODEQA QA accuracy.

Strategy	3-hop	4-hop	5/6-hop
A-Mem (Xu et al., 2025)	0.709	0.540	**0.518**
BM25 (Robertson & Zaragoza, 2009)	**0.808**	**0.555**	0.425
Naive RAG (Lewis et al., 2020)	0.753	0.538	0.442
MemoryBank (Zhong et al., 2024)	0.699	0.537	0.482
LightMem (Fang et al., 2025)	0.610	0.467	0.400
None	0.330	0.290	0.009

(b) MEMGYM-DR judge score.

Figure 4: Memory strategies on synthetic benchmarks. **(a)** MEMGYM-CODEQA QA accuracy across memory-context budgets from 64 to 512 tokens on 670 instances. Cluster bootstrap 95% CIs span at most ±2.1 points and are omitted for clarity. The shaded band spans the no-memory and full-context references. **(b)** MEMGYM-DR judge scores at 3, 4 and 5/6 hops with best values in bold. Scores are absolute. None is the no-memory reference, and differences from it are interpreted within each track.

runs. On WebArena-Infinity the structured summary raises success from 34.3% to 38.6%. The 0.7-point change under the rolling summary is within seed variation (Table 3). Differences in access to externally stored task information offer one possible explanation for this cross-track pattern. Reasoners, tasks, and compaction settings also differ. These gains measure the overall effect of context management rather than recovery of facts withheld from the reference, because the reference runs retain unmanaged histories.

4.3 MEMORY STRATEGIES UNDER CONTROLLED PRESSURE

The synthetic benchmarks complement the wrapped gyms by asking which tested configuration scores highest when pressure varies mainly along one axis, namely the memory-context budget on code in MEMGYM-CODEQA and hop count on scientific text in MEMGYM-DR (Figure 4). Every strategy on MEMGYM-CODEQA scores between the no-memory (0.53) and full-context (0.86) references at every budget. LightMem scores highest at each budget, rising from 0.75 at 64 tokens to 0.82 at 512. Its lead over MemoryBank and A-Mem is largest at the tightest budget (3.0 and 4.7 points) and narrows as the budget grows, until the three lie within 2.2 points at 512 tokens. Keeping only the most recent part of the episode trails LightMem by 13 to 15 points, which suggests that selecting task-relevant facts helps more than recency in the tested configurations.

On MEMGYM-DR every strategy loses ground as chains grow longer. BM25 drops from 0.808 at 3 hops to 0.425 at 5/6 hops, and the no-memory reference falls to 0.009. Longer chains require facts to

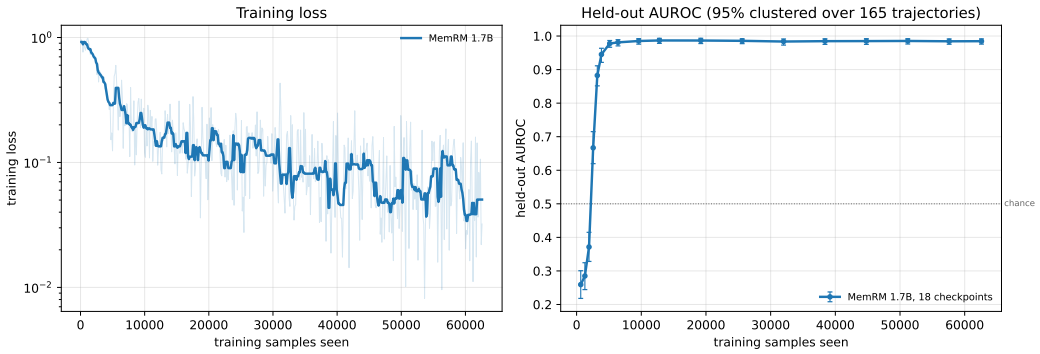


Figure 5: MEMRM training on SWE-Gym compression examples. **(a)** Training loss over 489 steps (62,592 examples seen). **(b)** AUROC of 18 checkpoints on the 3,007 held-out examples, with 95% intervals from resampling the 165 held-out trajectories.

be retained longer but also demand more reasoning steps, and we do not separate the two. A-Mem has the highest point estimate at 5/6 hops, followed by MemoryBank. The leader among the three systems evaluated on both tracks thus changes with the track, from LightMem on code to A-Mem on the deepest research chains. We cannot establish a domain-general ordering from two benchmarks. The pilot without fictionalization and per-hop intervals are in Section B.4.

4.4 PREDICTING REFERENCE-OUTPUT AGREEMENT

A lightweight compression scorer should be inexpensive to query and predict reference-output agreement on unseen events. We fine-tune Qwen3-1.7B-Base (Qwen Team, 2025), which is small enough to score an event in 0.6 seconds at batch size 1 on one A100 GPU (excluding the reasoner step that proposes the action), whereas a rollout takes about 4 minutes. We train LoRA adapters (rank 16) in bf16 at a 32K-token context because a compression event spans a long stretch of history. Matching examples make up only 12.8% of the training split, so the class-weighted loss of Equation 2 is computed on the single label token with class-balanced weights, 3.0 for matching examples (3.9 before capping) and 0.57 for mismatching ones. Training details are in Appendix E.

For the second requirement we hold out 3,007 examples from 165 trajectories on repositories absent from training, labeled as in Section 3.3. Accuracy says little because mismatching examples dominate. We therefore report AUROC (0.5 is chance) and expected calibration error (ECE) over 15 equal-mass bins, computed on the raw probabilities of the two label tokens without any calibration map. On this split MEMRM reaches an AUROC of 0.985 (95% interval 0.976–0.992, resampling trajectories) and an ECE of 0.017. We report the final checkpoint (step 489). AUROC stays between 0.981 and 0.987 from step 50 on while the training loss continues to fall (Figure 5), and ECE varies between 0.008 and 0.019. A string-similarity baseline between the proposed and recorded actions reaches an AUROC of 0.747, compared with 0.985 for MEMRM. On 1,832 paired SWE-Gym episodes, the score of the first compression predicts which of the 606 episodes resolved without memory become unresolved with it at an AUROC of 0.81 (95% interval 0.78–0.84). MEMRM is therefore a local check of compression events, not a substitute for rollouts.

5 CONCLUSION

MEMGYM provides a shared interface for evaluating memory configurations across interactive tasks and constructed retention benchmarks. Rolling summaries do not improve SWE-Gym performance in the tested configurations. Summary-based context management yields higher mean scores on tool-use dialogue, and structured summaries do so on web tasks. On ProgramBench, structured summarization raises the mean fraction of hidden tests passed from 9.3% to 48.7%. The leading strategies on MEMGYM-CODEQA close most of the gap to the full-context reference. MEMRM predicts reference-output agreement on perturbed events from held-out SWE-Gym repositories. Its first-compression score also predicts loss of baseline success in the paired-episode analysis (AUROC 0.81). MEMRM is a local check that complements full rollouts rather than replacing them.

AI USE STATEMENT

We used AI assistants to help search for relevant literature, synthesize data, implement code, and polish the writing of this paper. The authors supervised the entire process, reviewed and verified all outputs, and take full responsibility for the content of this paper.

REPRODUCIBILITY STATEMENT

The anonymous repository linked in the abstract hosts all code and data. It includes the benchmark construction pipelines and prompts, benchmark instances, model weights, and recorded trajectories. The paper and appendix describe the full experimental details needed to reproduce our results.

REFERENCES

Qingyao Ai, Yichen Tang, Changyue Wang, Jianming Long, Weihang Su, and Yiqun Liu. MemoryBench: A benchmark for memory and continual learning in LLM systems. *arXiv preprint arXiv:2510.17281*, 2025.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *International Conference on Learning Representations (ICLR)*, 2024.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 3119–3137, 2024.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, et al. LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3639–3664, 2025.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-Bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*, 2025.

Jiayang Cheng, Dongyu Ru, Lin Qiu, Yiyang Li, Xuezhi Cao, Yangqiu Song, and Xunliang Cai. AMemGym: Interactive memory benchmarking for assistants in long-horizon conversations. *arXiv preprint arXiv:2603.01966*, 2026.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3829–3846, 2023.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

Payel Das, Subhajit Chaudhury, Elliot Nelson, Igor Melnyk, Sarath Swaminathan, Sihui Dai, Aurelie Lozano, Georgios Kollias, Vijil Chenthamarakshan, Soham Dan, et al. Larimar: Large language models with episodic memory control. In *International Conference on Machine Learning (ICML)*, 2024.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. From local to global: A graph RAG approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

Jizhan Fang, Xinle Deng, Haoming Xu, Ziyan Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, et al. LightMem: Lightweight and efficient memory-augmented generation. *arXiv preprint arXiv:2510.18866*, 2025.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. HippoRAG: Neurobiologically inspired long-term memory for large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Zexue He, Yu Wang, Churan Zhi, et al. MemoryArena: Benchmarking agent memory in interdependent multi-session agentic tasks. *arXiv preprint arXiv:2602.16313*, 2026.

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*, pp. 6609–6625, 2020.

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? *arXiv preprint arXiv:2404.06654*, 2024.

Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating memory in LLM agents via incremental multi-turn interactions. *arXiv preprint arXiv:2507.05257*, 2025.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 13358–13376, 2023.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *The Twelfth International Conference on Learning Representations*, 2024.

Greg Kamradt. Needle in a haystack – pressure testing LLMs. https://github.com/gkamradt/LLMTest_NeedleInAHaystack, 2023. GitHub repository.

Rodney Kinney, Chloe Anastasiades, Russell Authur, Iz Beltagy, Jonathan Bragg, Alexandra Buraczynski, Isabel Cachola, Stefan Candra, Yoganand Chandrasekhar, Arman Cohan, et al. The Semantic Scholar open data platform. *arXiv preprint arXiv:2301.10140*, 2023.

Dong-Ho Lee et al. REALTALK: A 21-day real-world dataset for long-term conversation. *arXiv preprint arXiv:2502.13270*, 2025.

Kuang-Huei Lee, Xinyun Chen, Hiroki Furuta, John Canny, and Ian Fischer. A human-inspired reading agent with gist memory of very long contexts. *arXiv preprint arXiv:2402.09727*, 2024.

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

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024.

Tobias Lindenbauer, Igor Slinko, Ludwig Felder, Egor Bogomolov, and Yaroslav Zharov. The complexity trap: Simple observation masking is as efficient as LLM summarization for agent context management. *arXiv preprint arXiv:2508.21433*, 2025.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

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

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with SWE-Gym. *arXiv preprint arXiv:2412.21139*, 2024.

Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, et al. QuALITY: Question answering with long input texts, yes! In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5336–5358, 2022.

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 *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST)*, 2023.

Jason Priem, Heather Piwowar, and Richard Orr. OpenAlex: A fully-open index of scholarly works, authors, venues, institutions, and concepts. *arXiv preprint arXiv:2205.01833*, 2022.

Zengyi Qin, Jinyuan Chen, Yunze Man, Shengcao Cao, Ziqi Pang, Zhuoyuan Wang, Xin Sun, Gen Lin, Han Fang, Ling Zhu, et al. OSGym: Super-scalable distributed data engine for generalizable computer agents. *arXiv preprint arXiv:2511.11672*, 2025.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Stephen Robertson and Hugo Zaragoza. *The probabilistic relevance framework: BM25 and beyond*, volume 4. Now Publishers Inc, 2009.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. RAPTOR: Recursive abstractive processing for tree-organized retrieval. In *International Conference on Learning Representations (ICLR)*, 2024.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 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.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023a.

Qingyue Wang, Liang Ding, Yanan Cao, Zhiliang Tian, Shi Wang, Dacheng Tao, and Li Guo. Recursively summarizing enables long-term dialogue memory in large language models. *arXiv preprint arXiv:2308.15022*, 2023b.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, et al. OpenHands: An open platform for AI software developers as generalist agents. *arXiv preprint arXiv:2407.16741*, 2024a.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*, 2023c.

Yu Wang, Yifan Gao, Xiusi Chen, Haoming Jiang, Shiyang Li, Jingfeng Yang, Qingyu Yin, Zheng Li, Xian Li, Bing Yin, Jingbo Shang, and Julian McAuley. MemoryLLM: Towards self-updatable large language models. In *International Conference on Machine Learning (ICML)*, 2024b.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. *arXiv preprint arXiv:2410.10813*, 2024.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-Mem: Agentic memory for LLM agents. *arXiv preprint arXiv:2502.12110*, 2025.

John Yang, Kilian Lieret, Carlos E Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. SWE-smith: Scaling data for software engineering agents. *arXiv preprint arXiv:2504.21798*, 2025.

John Yang, Kilian Lieret, Jeffrey Ma, Parth Thakkar, Dmitrii Pedchenko, Sten Sootla, Emily McMilin, Pengcheng Yin, Rui Hou, Gabriel Synnaeve, Diyi Yang, and Ofir Press. ProgramBench: Can language models rebuild programs from scratch? *arXiv preprint arXiv:2605.03546*, 2026.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William 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*, pp. 2369–2380, 2018.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023b.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

Yujie Zhao, Boqin Yuan, Junbo Huang, Haocheng Yuan, Zhongming Yu, Haozhou Xu, Lanxiang Hu, Abhilash Shankarampeta, Zimeng Huang, Wentao Ni, Yuandong Tian, and Jishen Zhao. AMA-Bench: Evaluating long-horizon memory for agentic applications. *arXiv preprint arXiv:2602.22769*, 2026.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. MemoryBank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 19724–19731, 2024.

Shuyan Zhou. WebArena-Infinity: Generating browser environments with verifiable tasks at scale, March 2026. URL https://webarena.dev/webarena-infinity/.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. WebArena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.

CONTENTS

1 Introduction **1**

2 Related Work **2**

3 The MEMGYM Evaluation Framework **3**

3.1 Where the Memory Module Has to Sit 4

3.2 Constructing Tracks That Target Retention 5

3.3 A Learned Scorer for Compression Events 5

4 Evaluating Memory Strategies Across Tracks **6**

4.1 Experimental Setup 6

4.2 Memory Across Four Wrapped Gyms 7

4.3 Memory Strategies Under Controlled Pressure 8

4.4 Predicting Reference-Output Agreement 9

5 Conclusion **9**

A Extended Related Work **15**

A.1 Agentic Memory Systems 15

A.2 Context Compression and Long-Context Evaluation 15

A.3 Long-Horizon Agentic Benchmarks 15

A.4 Reward Models and Learned Evaluators 15

B Extended Experimental Details **16**

B.1 Per-Track Experimental Setup 16

B.2 Asset Licenses 16

B.3 SWE-Gym Pairing 17

B.4 MEMGYM-DR Fictionalization Pilot and Per-Hop Results 17

B.5 MEMRM Data Augmentation 17

C Synthetic Pipeline Details **17**

C.1 MEMGYM-DR 18

C.2 MEMGYM-CODEQA 18

D Memory Strategy Implementation Details **19**

E MEMRM Training Details **19**

A Extended Related Work

A.1 Agentic Memory Systems

Memory for LLMs was first used to extend finite context windows in long-running assistants. MemoryBank (Zhong et al., 2024) stores and updates user memories, MemGPT (Packer et al., 2023) pages between working and archival context, and ReadAgent (Lee et al., 2024) keeps gist memories of long documents. Prompting methods reuse earlier outputs as in-context memory through self-reflection and revision (Shinn et al., 2023; Madaan et al., 2023), sampling and search over reasoning paths (Wang et al., 2023c; Yao et al., 2023a), or interleaved reasoning and action traces (Yao et al., 2023b; Schick et al., 2023). Other work organizes experience into memory streams (Park et al., 2023), skill libraries (Wang et al., 2023a; Zhao et al., 2024), hierarchical or graph indexes (Sarthi et al., 2024; Edge et al., 2024; Gutiérrez et al., 2024), or recursive summaries (Wang et al., 2023b). Some systems instead use managed memory services (Chhikara et al., 2025), parametric memory (Wang et al., 2024b; Das et al., 2024), or retrieval with self-critique (Asai et al., 2024). These mechanisms are usually evaluated in one domain. MemGym offers a common interface across domains.

A-Mem (Xu et al., 2025) organizes interactions into linked, evolving notes and is evaluated on LoCoMo (Maharana et al., 2024), a long-term conversational benchmark. LongMemEval (Wu et al., 2024) covers several memory abilities over long chat histories, and MemoryAgentBench (Hu et al., 2025) tests incremental memory updates. MemoryBench (Ai et al., 2025) studies continual learning from user feedback, and REALTALK (Lee et al., 2025) studies memory in real long-term conversations. MemoryArena (He et al., 2026) evaluates interdependent multi-session agentic tasks, AMA-Bench (Zhao et al., 2026) memory over agentic trajectories, and AMemGym (Cheng et al., 2026) on-policy interaction with simulated users. MemGym complements these benchmarks by placing memory inside coding, search, dialogue, and web environments. Compared runs within a track hold the environment, scaffold, and reasoner fixed while the memory module varies.

A.2 Context Compression and Long-Context Evaluation

Prompt compression methods such as LLMLingua (Jiang et al., 2023) and AutoCompressors (Chevalier et al., 2023) shorten a static prompt for efficient inference, whereas MemRM evaluates compressions made during an interaction. Long-context benchmarks (Kamradt, 2023; Hsieh et al., 2024; Bai et al., 2024; 2025), multi-hop QA (Yang et al., 2018; Ho et al., 2020; Trivedi et al., 2022), and long-document QA (Pang et al., 2022) measure recall and reasoning over a given input. An agent's memory must also decide what to keep while the context grows turn by turn.

A.3 Long-Horizon Agentic Benchmarks

SWE-bench (Jimenez et al., 2024) and SWE-Gym (Pan et al., 2024) evaluate issue resolution in real Python repositories with executable tests, and ProgramBench (Yang et al., 2026) asks agents to rebuild a program from its compiled executable. τ-bench and τ^2-bench (Yao et al., 2024; Barres et al., 2025) evaluate tool-agent-user dialogue, WebArena (Zhou et al., 2023) and WebArena-Infinity (Zhou, 2026) web tasks, and OSWorld (Xie et al., 2024) and OSGym (Qin et al., 2025) desktop computer use. Memory contributes to outcomes in all of them. MemGym adds a memory boundary that follows the condenser interface of OpenHands (Wang et al., 2024a). This boundary lets us compare memory modules while the environment, reasoner, and scaffold stay fixed.

A.4 Reward Models and Learned Evaluators

Reward models (Ouyang et al., 2022) and process reward models (Lightman et al., 2024) provide outcome- and step-level supervision for LLMs. MemRM applies this idea to memory by scoring individual compression events for reference-output agreement. We do not use or evaluate it as a reward for training memory policies.

B Extended Experimental Details

B.1 Per-Track Experimental Setup

On SWE-Gym we evaluate the pooled set of 1,054 instances from 11 Python repositories. It combines a 500-instance subset spanning all repositories with a 554-instance subset of issues that have long Claude Opus 4.8 trajectories. Between 1,003 and 1,041 instances are scored per reasoner because the rest failed to start in their Docker containers. The scaffold is mini-swe-agent with a 250-step limit, and patches are scored with the official SWE-bench harness. Compaction fires once the history exceeds 100 messages. It keeps the first three messages, and Claude Haiku 4.5 summarizes 75% of the rest while the most recent messages stay intact. The dollar figures in Section 3.3 are the input-token cost of one no-memory episode with Claude Sonnet 4.5 at $3 per million input tokens.

On τ^2-bench we use all 288 tasks with Claude Haiku 4.5, and both strategies are compared with the same no-memory runs on the same tasks. Compaction fires at 10 messages and keeps the last 2 messages under summary memory and the last 4 under structured memory, so the two strategies also differ in the retained tail. On WebArena-Infinity the policy is Claude Haiku 4.5 with text accessibility-tree observations and a 50-step limit. To separate memory effects from early exploration, we inject the first half of a recorded trajectory as a fixed prefix and run the rest live (140 hard tasks with compaction at 20 messages). A no-memory run on the same prefix serves as the reference.

On MemGym-DR, Claude Sonnet 4.5 answers and Claude Haiku 4.5 judges. A-Mem and LightMem write their notes with Claude Sonnet 4.5. The judge receives only the question, the reference facts, and the agent's final answer. It sees none of the agent's notes, reasoning, or memory state. Claude Sonnet 4.5 also answers on MemGym-CodeQA, where A-Mem and LightMem write their memories with it. The 670 evaluated instances are the first 670 of the 4,289 released instances ordered by a seeded hash of their identifiers, so every strategy and budget is scored on the same 2,324 QA pairs. A QA pair is correct when at least half of its linked facts are recalled in the answer, judged by overlap of key terms. The no-memory reference receives the task description, a repository overview, and the question but no retained debugging-session context. It scores 0.53 under this evaluation.

Table 4: Evaluated combinations of memory strategy and track, with the reasoner of each track. The memory boundary admits every strategy on every track. We evaluate each track with the strategy family that matches how it uses memory. The wrapped gyms use compaction because the agent keeps acting on its own history. The constructed tracks use external memory because a fixed session is ingested and then queried, and truncation serves as a recency reference on MemGym-CodeQA. Every track also runs the no-memory condition $\varnothing$.

Strategy	SWE-Gym	ProgramBench	τ^2-bench	WebArena-Inf.	MemGym-CodeQA	MemGym-DR
Rolling summary	✓	–	✓	✓	–	–
Structured summary	–	✓	✓	✓	–	–
Truncation	–	–	–	–	✓	–
BM25, Naive RAG	–	–	–	–	–	✓
A-Mem, MemoryBank, LightMem	–	–	–	–	✓	✓
Reasoner	Sonnet 4.5, Haiku 4.5, GPT-OSS-120B	Sonnet 4.6	Haiku 4.5	Haiku 4.5	Sonnet 4.5	Sonnet 4.5

On ProgramBench we use the official harness with Claude Sonnet 4.6 on 53 randomly sampled tasks and cap requests at 154,944 input tokens, the 200K-token window less a reserve for output. Without context management an episode ends once the next request would exceed this cap. The 264K figure in Section 3.2 is measured on the official trajectories, which were recorded without this cap. The structured summary acts whenever the history exceeds the cap. It replaces the oldest turns with a state of at most 2,048 tokens written by Claude Haiku 4.5 and keeps the most recent turns up to 75% of the remaining window. It drops each tool call together with its result. Both conditions share a 1,000-step budget, and tasks are scored by the fraction of hidden tests passed, with zero for an episode that ends without a graded submission.

B.2 Asset Licenses

Third-party assets are used under their respective open licenses. SWE-Gym (Pan et al., 2024), SWE-bench (Jimenez et al., 2024), and SWE-smith (Yang et al., 2025) are released under the MIT

Table 5: Memory strategies on the MEMGYM-DR corpus by number of reasoning hops. Scores are judge scores in $[0, 1]$. The $\pm$ values are 95% confidence intervals (1.96 standard errors). A run that fails to produce an answer scores zero.

Strategy	3-hop ($n = 398$)	4-hop ($n = 412$)	5/6-hop ($n = 384$)
None (no memory)	0.330 ± 0.049	0.290 ± 0.014	0.009 ± 0.011
BM25 over notes (Robertson & Zaragoza, 2009)	$\mathbf{0.808 \pm 0.037}$	$\mathbf{0.555 \pm 0.017}$	0.425 ± 0.048
Naive RAG (Lewis et al., 2020)	0.753 ± 0.039	0.538 ± 0.016	0.442 ± 0.045
A-Mem (Xu et al., 2025)	0.709 ± 0.043	0.540 ± 0.016	$\mathbf{0.518 \pm 0.037}$
LightMem (Fang et al., 2025)	0.610 ± 0.050	0.467 ± 0.036	0.400 ± 0.044
MemoryBank (Zhong et al., 2024)	0.699 ± 0.044	0.537 ± 0.035	0.482 ± 0.044

license. The same license covers τ^2-bench (Barres et al., 2025), ProgramBench (Yang et al., 2026), and the OpenHands condenser interface (Wang et al., 2024a). WebArena-Infinity (Zhou, 2026) is used under the terms of its public release. WebArena (Zhou et al., 2023), the Qwen3-1.7B base model used for MEMRM, and GPT-OSS-120B (one of the reasoners used to collect trajectories) are released under Apache-2.0. The closed-source models (Claude Sonnet 4.5, Claude Sonnet 4.6, Claude Haiku 4.5, and Claude Opus 4.8) are accessed through the AWS Bedrock API under Anthropic's commercial terms of use. The MEMGYM wrappers, the paired-trajectory corpus, and the synthetic MEMGYM-CODEQA instances are released under MIT. The MEMGYM-DR instances are released for research use, and their source documents remain under their original licenses. The MEMRM weights inherit the Apache-2.0 license of the Qwen3-1.7B base.

B.3 SWE-GYM PAIRING

On SWE-Gym a strategy acts only on instances whose baseline trajectory crosses the compaction trigger (65.1% of instances for Sonnet 4.5, 63.7% for Haiku 4.5, 55.4% for GPT-OSS-120B). The memory run equals the baseline on all other instances. The full-set resolve rates of Table 3 therefore add the baseline outcomes of these instances, and the full-set change is the paired change scaled by the share of instances that compact.

B.4 MEMGYM-DR FICTIONALIZATION PILOT AND PER-HOP RESULTS

Without fictionalization, a model can answer MEMGYM-DR questions from parametric knowledge. The no-memory condition scored 0.70 to 0.85 in a 100K-token pilot built without it. We replace real entities and numbers with invented ones consistently across each instance (Appendix C) to reduce this shortcut. The no-memory reference then falls to 0.330 at 3 hops, 0.290 at 4, and 0.009 at 5/6 hops. Table 5 gives the full results behind Figure 4(b).

B.5 MEMRM DATA AUGMENTATION

Each MEMRM example applies one perturbation of Table 6 to a recorded SWE-Gym compression event. We run the agent's next action under the perturbed view in a snapshot of the environment. The example is labeled a match when this action returns the recorded output and exit status after normalization, and a mismatch otherwise. The perturbations model a compressor that loses information, inserts incorrect content, or drops recent context. Every event receives all six perturbations and some receive a second draw. The resulting 18,637 examples come from 2,690 events in 570 trajectories, and 12.2% are matches. The split is grouped by repository, with 15,630 training examples (12.8% matching) and 3,007 held-out examples (9.2% matching) from seven unseen repositories.

C SYNTHETIC PIPELINE DETAILS

Both synthetic pipelines start from a source, which is a research topic with retrieved documents for MEMGYM-DR and a SWE-smith bug with its hidden patch for MEMGYM-CODEQA. They add distractors, check for answer leakage, and filter for fact recovery and answer support.

Table 6: Perturbations applied to recorded compression events to build MEMRM examples. Every recorded event receives all six perturbations.

Perturbation	Construction
Aggressive masking (50%)	Replace 50% of summary tokens with noise or blanks, modeling an aggressive compression that ignores content.
Random sentence drop (20%)	Randomly drop 20% of summary sentences, modeling a lossy summarizer that omits content uniformly at random.
Path deletion	Remove file paths from the summary, modeling a summarizer that under-records where things are.
Redaction	Redact key entities and values that are referenced later in the trajectory, modeling over-compression that loses task-critical specifics.
Noise injection	Insert plausible but incorrect facts (wrong file names, wrong test outcomes), modeling hallucinated summarizer output.
Recent-context removal	Drop the last 10 messages from the agent's view, a stress test for losing recent observations not yet folded into the summary.

C.1 MEMGYM-DR

The pipeline searches academic sources on a topic and builds a chain of dependent bridge facts. Each fact has a retention span, namely the number of hops between its first appearance and its use. A four-hop instance holds spans of 3, 2, and 1 plus a terminal fact. The chain is expanded into a multi-turn research session and filled to the target length (100K tokens here) with real search results and generated academic prose. We then fictionalize it. An LLM proposes invented substitutes for entities and numbers, and a deterministic substitution applies them across the whole instance. Four tiers of distractors target different shortcuts. Natural distractors from the same searches target topic-based shortcuts, and near-miss rewrites of gold facts target surface matching. Contradictions target majority-based shortcuts, and filler adds volume without signal.

Construction-time filtering uses gold bridge facts, the full document set, and individual documents as three fixed reference inputs. An instance is retained when the notes score is at least 0.85 and the full-context score at most 0.90. Each single-document score must be at most 0.5, and the notes score must be no lower than the full-context score. None of these inputs is produced by an evaluated memory strategy. All strategies receive the same selected instances at evaluation time and are scored by the Haiku 4.5 judge. The filters select instances on which curated notes support strong answers under the construction-time evaluation. They do not establish that every bridge fact is individually necessary or remove selection bias.

C.2 MEMGYM-CODEQA

MEMGYM-CODEQA keeps SWE-smith bugs with nontrivial, localized patches. Each kept bug has 5 to 500 changed lines, at least two hunks and two tests that fail before the fix, and a problem statement of at least 100 characters. An LLM prescreen then rejects candidates whose target facts it recovers from the problem statement and the repository. These checks retain 4,289 of 4,999 candidates (85.8%) and yield 14,899 QA pairs over 32 repositories.

The pipeline targets facts observed during debugging that its recovery checks do not recover from the repository. We propose questions with the gold patch visible, asking what a developer would need to know rather than what to change. We then extract and label facts with the patch hidden and check them against the full repository. We keep an instance only if at least two of its facts are not recovered by this check. The facts are turned into single- and multi-fact questions, which are checked for solvability and leakage and deduplicated. Distractors from other bug reports, same-repository docstrings, near-misses, and contradictions are interleaved with the facts in natural documents such as incident reports and review comments. The question is asked only after the turns carrying the facts have left the context.

D MEMORY STRATEGY IMPLEMENTATION DETAILS

The rolling summary follows the OpenHands condenser (Wang et al., 2024a). It keeps a pinned head and a recent tail and replaces the middle with a summary of the actions, findings, and decisions it contains once the history exceeds the trigger. The structured summary fills an environment-specific schema instead, so the state a developer would otherwise reconstruct is kept explicitly. Truncation keeps the most recent messages that fit the budget. Every strategy on MEMGYM-CODEQA gets the same memory-context budget, the pressure factor we vary. On MEMGYM-DR, BM25 (Robertson & Zaragoza, 2009) and Naive RAG (Lewis et al., 2020) store observations as independent notes and retrieve them by lexical or embedding similarity. A-Mem (Xu et al., 2025) adds LLM-written metadata and links between notes, so a note can be reached through a related one. MemoryBank (Zhong et al., 2024) and LightMem (Fang et al., 2025) consolidate memories over time, MemoryBank by updating stored memories and LightMem through staged short- and long-term stores.

E MEMRM TRAINING DETAILS

MEMRM reads a compression event and predicts reference-output agreement. The output tokens `safe` and `harmful` encode the match and mismatch labels. The examples (Appendix B.5) come from SWE-Gym rollouts of Sonnet 4.5, Haiku 4.5, and GPT-OSS-120B and are split by repository into 15,630 training examples from 405 trajectories and 3,007 held-out examples from 165 trajectories. We fine-tune Qwen3-1.7B-Base (Qwen Team, 2025) with LoRA (rank 16, $\alpha = 32$) in bf16 on the attention projections at a maximum length of 32,768 tokens, with the class-weighted loss of Equation 2 on the label token. The class-balanced weights $N/(2N_y)$ come from the training split and give 0.57 for the `harmful` token and 3.9 for the `safe` token, capped at 3.0. Training runs for 489 steps (about four epochs) at a learning rate of 5×10^{-5} with an effective batch size of 128 and a cosine schedule with 5% warmup, and we report the final checkpoint. Training takes about one hour on 128 A100 GPUs. The predicted match probability is the softmax probability of the `safe` token over the two label tokens. We apply no calibration map and tune no threshold on held-out data.