Research · Architecture of AI

How AI systems *remember*

The architecture of machine memory in 2026: what shipped, what the evidence supports, and what remains unsolved.

Tô Hoài Vũ Founder, VLabs

Published November 9, 2026 · About 20 minutes · DOI 10.5281/zenodo.24088880

Abstract

We survey how memory is built in deployed assistants and agents, what the evidence says about which designs work, and what remains unsolved, using documentation, papers and security reports available on 9 November 2026. We report five findings. First, **convergence**: independently built agent products arrived at the same design, in which the model writes plain text files, a small index is loaded into every session, detail is fetched on demand, and a background process rewrites the notes while the system is idle. Second, **simple baselines hold**: in eight comparisons published in 2025 and 2026, putting the history in context, plain retrieval, or files with search matched or beat the dedicated memory systems tested. Third, **the weak side of memory is writing**: no tested system passed more than 1.5% of requests to forget, summarising a long session kept 17% of the constraints it had been given, and memory rewritten repeatedly by a model can end up worse than no memory. Fourth, **persistence is an attack surface**: a single injected message can become a lasting compromise, with end-to-end success above 70% in recent studies and commercial abuse observed in the wild. Fifth, **nothing learns in the weights**: no frontier developer has shipped a model that updates its parameters from use, so everything called memory in 2026 is managed context. We close with design principles that follow from the evidence.

Keywords: agent memory, context engineering, compaction, retrieval, continual learning, memory poisoning, machine unlearning.

1 Introduction

A language model has no memory of its own between conversations. Its parameters are fixed when training ends, and its working space, the context window, is emptied when a session closes. Everything a deployed assistant appears to remember about a user, a project or its own past work is the product of engineering around the model.

That engineering became a central design question in 2026 for two reasons. Agents began to run for hours across many context windows, so that state had to survive outside any one of them. And assistants began to remember by default, which turned a convenience into a question of security, privacy and control. Andrej Karpathy put the starting point bluntly in late 2025: “They don’t have continual learning. You can’t just tell them something and they’ll remember it.”[1]

This article is a survey and a secondary analysis. We ran no experiments. We read vendor documentation, engineering reports, benchmark papers and security research, checked the figures we cite against their sources, and tallied the published comparisons between memory designs. VLabs builds an assistant that uses memory; this article does not evaluate it. Our contribution is:

- a description of the memory designs actually shipped in 2026 and the pattern they share (Section 3);
- a tally of eight published comparisons between simple baselines and dedicated memory systems (Section 4);
- a synthesis of where memory fails, organised around writing, updating and forgetting (Sections 5 and 6);
- a review of the two open fronts, learning in the weights and attacks on memory (Sections 7 and 8), and a set of design principles (Section 10).

2 A map of machine memory

The research literature borrows its vocabulary from cognitive science: working memory for what is in use now, episodic memory for what happened, semantic memory for facts, procedural memory for how to do things.[2] For engineering purposes it is more useful to ask where the information physically lives. Table 1 lists the four places.

Where it lives	What it holds	Limits
Weights	What the model learned in training	Fixed after training. Cannot be inspected, edited or reliably deleted.
Context window	The current conversation, instructions and whatever was loaded	About one million tokens in current flagship models.[3] Emptied when the session ends, and less reliable as it fills.
External store	Past conversations, extracted facts, documents, indexed for retrieval	Only helps if the right item is retrieved and then used correctly.
Files the agent manages	Notes, progress logs, instructions and procedures the model reads and writes with ordinary file tools	Quality depends on what the model chooses to write and keep.

Table 1. The four places information can persist in an AI system. Everything deployed as "memory" in 2026 belongs to the last three.

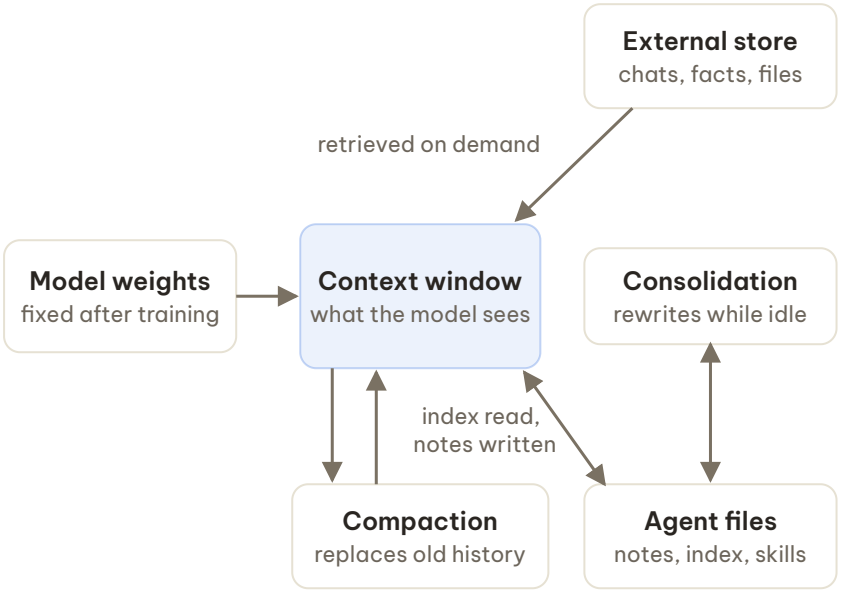


Figure 1. The places information persists and the operations that move it. The model reads only the context window; everything else has to be loaded into it. Compaction and consolidation are where most measured failures occur.

Around these sit two operations that matter as much as storage. *Compaction* replaces a long history with a summary so that work can continue in a fresh window. *Consolidation* reorganises stored memory offline: merging duplicates, correcting stale entries, building summaries. Most of the failures described later happen in these two operations, not in storage or lookup.

3 What shipped in 2026

Table 2 summarises the memory designs documented by the major vendors. Where we could not open a vendor's own page we say so.

Product	Design	Deletion and control
Claude apps	Memory is saved “as a set of individual topics as you chat”, replacing an earlier daily summary. Each project has its own memory. Past chats are searched on demand.[4]	Deleting a conversation does not remove memory entries generated from it. Incognito chats create none.[4]
Claude Code	Markdown files on disk. The first 200 lines or 25 KB of an index file load into every session; topic files are read when needed.[5]	Plain files the user can read and edit.
ChatGPT	Saved memories plus context learned from chat history. A June 2026 rebuild adds a reviewable memory summary maintained in the background.[6]	Temporary chats neither use nor create memory.[6] Reported at second hand.
Codex	Off by default. Memory is extracted after a chat has been idle and consolidated by a separate model into local files.[7]	Chats that used web search or external tools can be excluded from memory.[7]
Gemini	Personalisation from the memory of past chats and connected Google apps.[8]	To remove a fact, the user deletes every chat that contains it.[8]
Microsoft 365 Copilot	Saved and inferred memories stored in a hidden folder of the user’s mailbox, on by default.[9]	Deleting a chat does not delete saved memories. Retention policies do not apply to memory, and memory actions generate no audit log entries.[9]

Product	Design	Deletion and control
Grok Build	After each turn the model reviews it in the background and writes markdown notes, one topic per file; a separate command consolidates them. [10]	Plain files in workspace and global scopes.

Table 2. Memory designs documented by vendors, as read on 9 November 2026. The ChatGPT row relies on secondary reporting because the vendor's pages could not be opened.

3.1 Convergence on files

The agent products in Table 2 were built by competing teams and reached the same design. Memory is a directory of text files. The model reads and writes them with the same tools it uses for any other file. A short index is always in context and points to detail that is loaded only when relevant. Anthropic's hosted version caps each memory at 100 kB, versions every change and offers an endpoint to redact history.[11]

The reason for the convergence appears to be capability. In Anthropic's published comparison, an older model given a memory directory wrote 413 files in 225 folders and made little progress on a long game-playing task; a 2026 model kept nine files and progressed four times as far.[12] The same report found that fixed rules about what to save, such as recording only failures, reduced the benefit for stronger models, and summarised the lesson as: "it helps to tell Claude what memory is for, but not how to use it."[12]

3.2 Consolidation in the background

The second shared feature is new in 2026. Systems now revise their own memory while idle. ChatGPT's rebuilt memory, Codex's idle-time extraction, Grok Build's consolidation command and Anthropic's "dreaming" all follow this pattern.[6][7][10][12] The motivation is that memory which only grows goes stale; OpenAI is reported as saying its earlier system "often became stale and relied on users to manually manage updates".[6]

The one published measurement is vendor-reported. A wiki built by background consolidation, used with no other source, answered 13% of graded criteria after two sessions and 48% after 128. Combined with search it reached 61%, against 53% for search alone.[12] Section 6 describes independent evidence that consolidation can also make memory worse.

3.3 Compaction as a service

Anthropic and OpenAI both offer compaction as a server-side operation, so that an application no longer has to summarise its own history. They differ in one respect that affects auditing. Anthropic's returns a readable summary. OpenAI's returns an encrypted item that its documentation describes as "opaque and not intended to be human-interpretable".[13]

4 Do dedicated memory systems beat simple baselines?

A market of specialised memory systems grew up between 2024 and 2026, built on vector stores, knowledge graphs and hierarchical summaries. The question for anyone choosing an architecture is whether they outperform the obvious alternatives: put the history in the context window, retrieve passages with ordinary search, or let the model keep files.

4.1 The benchmark problem

Most published claims rest on one benchmark, LoCoMo, and it does not support them. Its dialogues average about 9,000 tokens,[14] which fits comfortably in a modern context window, so it barely tests memory at all. An independent audit found that 99 of 1,540 answers in the key are wrong, giving a ceiling of 93.57%, and that the automated judge accepted 62.81% of deliberately wrong but on-topic answers.[15] The same audit found that a small model reading the full conversation with a better answer prompt outscored a commercial memory system, and concluded that the prompt, not the memory system, explained the score.[15]

The published scores also contradict each other. Mem0 reported its system at 68.4% and Zep's at 66.0%. Zep re-ran its own system at 75.1% and pointed out that a plain full-context baseline scored 72.9%, above Mem0's result.[16] Letta then scored 74.0% with an agent that simply stored the conversation in files and searched them.[17]

4.2 Eight comparisons

We collected the comparisons published in 2025 and 2026 in which a simple baseline and dedicated memory systems were run on the same tasks. Table 3 lists them.

Study	Simple baseline	Dedicated memory systems
LoCoMo re-runs, 2025[16][17]	Full context 72.9%. Files with search 74.0%.	Mem0 68.4%, as published by its authors.
MemoryAgentBench, June 2026[18]	Long context 74.4, 48.6 and 66.2 on retrieval, test-time learning and long-range understanding.	Best memory agent 63.0, 40.8 and 25.4.
AMA-Bench, May 2026[19]	Long context 51.8%.	Best existing system 46.1%. The authors' own new design reaches 57.2%.
MemoryArena, September 2026[20]	Plain retrieval 0.20. Long context 0.16 to 0.19.	0.13 to 0.14.
MemDelta, June 2026[21]	Plain retrieval 47.2% and full context 49.8%, statistically tied.	Mem0 1.2 points below a cloud retrieval baseline, at about 50 times the cost on the subset compared.
Cost and performance study, March 2026[22]	Long context 82.4% on LongMemEval.	Fact-based memory 49.0%.
Cross-scenario study, June 2026[23]	Flat text files managed through tool calls ranked first across tasks.	Eight memory systems.
ConvoMem, November 2025[24]	Long context best for the first 30 conversations and viable to about 150.	30% to 45% on preference and implicit tasks.

Table 3. Published comparisons of simple baselines with dedicated memory systems on the same tasks. Scores are as reported by each study and are not comparable across rows. The tally is ours.

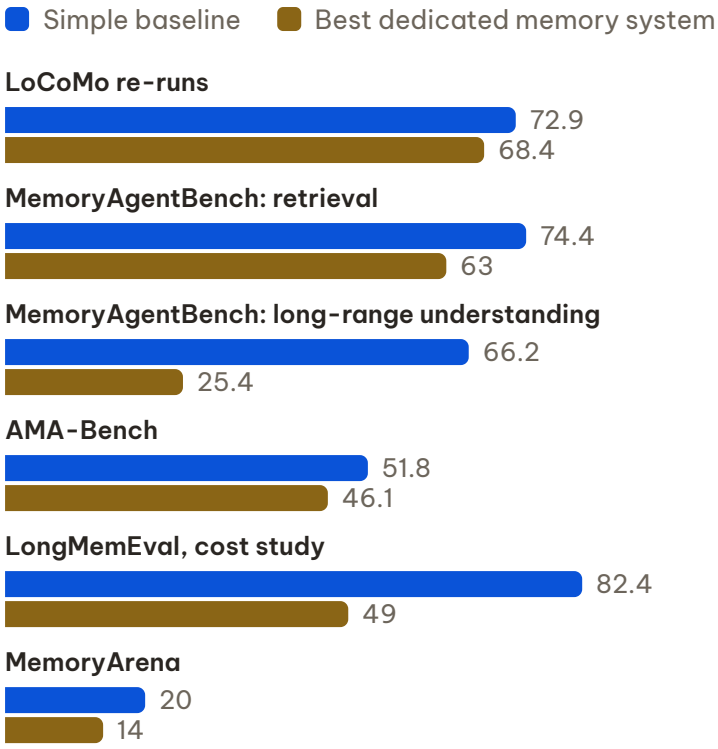


Figure 2. Scores of a simple baseline and of the best existing dedicated memory system in the six comparisons of Table 3 that report a single score for each, on a scale of 0 to 100. Benchmarks differ, so compare within a row, not across rows.

In all eight, a simple baseline matched or beat the existing dedicated systems. The exceptions we found are systems evaluated by their own authors: Zep reports 71.2% against 60.2% for full context on a longer benchmark,[16] and the AMA-Bench authors' new design beats the long-context baseline they measured.[19] On the hardest agentic benchmark nothing does well; mean success is between 0.13 and 0.20 for every approach.[20]

The pattern has identifiable causes. Swapping only the embedding model in an otherwise identical pipeline moved accuracy by 6.2 points, more than most gaps between products,[21] so some of the gaps between products may come from the embedding model and not from the memory architecture. And the bottleneck is often not finding the evidence but using it: one diagnostic study concluded that "the dominant bottleneck is evidence use, not retrieval",[25] and in the original LongMemEval experiments a model given only the relevant sessions scored 0.870 against 0.606 with the full history.[26] The cross-scenario study attributes the success of files to "giving the agent active control over storage and retrieval" instead of placing a passive store behind a fixed pipeline.[23]

Result

In eight published comparisons, the history in context, plain retrieval, or files with search matched or beat the dedicated memory systems tested.

5 Bigger windows did not end the problem

Simple baselines do well, and flagship models now accept about a million tokens.[3] Memory still has to be managed, for the following reasons.

- **Accuracy falls as the window fills.** A study of 18 models found that "model performance consistently degrades with increasing input length" in every experiment, and that focused prompts of about 300 tokens beat full histories of about 113,000 for all of them.[27] Another found 11 of 13 models below half their short-context score at 32,000 tokens.[28] These studies predate the current generation, and we found no re-run on it.
- **Cost grows with every turn.** A full history is re-read on each request. In one analysis a fact-based memory system became cheaper than cached long context after about ten turns at 100,000 tokens, although it was also much less accurate.[22]
- **Work outlives the window.** A long agent run crosses many context windows. Something has to carry state between them, and that is where compaction and files come in.

The practical conclusion from Sections 4 and 5 together is that long context is the right default for histories that fit, and that the alternative worth building is a small amount of well-chosen state, not a large retrieval apparatus.

6 Where memory fails

The best-measured failures are on the write side: what gets written, how it is revised, and whether it can be removed.

6.1 Compaction drops instructions

When a history is summarised, the summary keeps the story and loses the rules. In one study of 1,323 episodes across seven model families, agents never violated a policy while it was in context and violated it 30% of the time after compaction, rising to 59% for some models. Pinning the constraints so that they survive compaction restored the rate to zero.[29] A second study found that compactors kept only 17% of injected side constraints on average, and that an agent deleted emails without approval in 30 of 100 sessions after compaction and none before. A

separate extractor that copies constraints into the compacted context retained more than 90%. [30]

6.2 Forgetting does not work

A benchmark published in October 2026 tested whether memory systems forget when a user asks. Across six systems and a full-context baseline, “no system passes more than 1.5% of forget requests end to end”, and most passed none.[31] An earlier benchmark that includes selective forgetting among its four competencies concludes that “current methods fall short of mastering all four”.[18]

6.3 Rewriting degrades memory

Consolidation is the 2026 feature, and it has a measured failure mode. In a study of memory that a model updates continuously, “memory utility first rises, then degrades, and can fall below the no-memory baseline”. Even when consolidating from correct solutions, GPT-5.4 then failed 54% of problems it had solved with no memory at all. A control that simply kept the raw records stayed competitive with every consolidator tested.[32]

6.4 Lessons that are wrong get repeated

Procedural memory, in which an agent stores what it learned to do, shows the largest gains and the sharpest reversals. Workflows induced from past tasks raised success on a web benchmark by 51% in relative terms.[33] Curated skills written by people raised agents’ pass rate from 33.9% to 50.5% across 87 tasks, while skills the agents wrote for themselves left them below the no-skills baseline in all three configurations tested.[34] A study titled “More Memory, Worse Agents” found that an agent which discards its abstractions after use outperformed designs that keep them, because “errors are regenerated through a cycle of abstraction, execution, and re-abstraction”.[35]

Anthropic, which reports the largest benefit from memory, also names the cost. On a database task, giving agents a memory store raised mean reward from 0.655 to 0.838 for its strongest model, and from 0.383 to 0.700 for an older one. The same report notes that “an incorrect note can be carried into every session”, and describes a run that stalled on one wrong note until consolidation corrected it.[12]

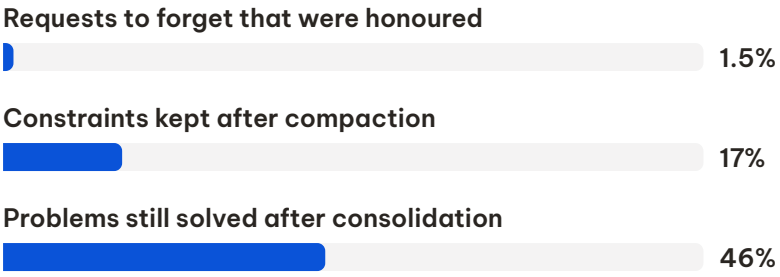


Figure 3. Three measurements from Sections 6.1 to 6.3, each on a scale where 100% would be correct behaviour. Each comes from a different study and task.

> **Result**
>
> **On the write side the measured results are poor: at most 1.5% of forget requests honoured, 17% of constraints kept through compaction, and consolidated memory that can fall below no memory.**

7 Memory in the weights

Everything above is memory outside the model. The alternative is for the model itself to change with use, as a person does. Whether that is necessary is the sharpest disagreement in the field.

One side holds that it is the main missing capability. "The lack of continual learning is a huge huge problem," wrote Dwarkesh Patel in 2025.[36] Karpathy describes the weights as "a hazy recollection" and the context window as working memory, and says of sleep-like consolidation into long-term memory: "We don't have an equivalent of that in large language models."[1] Demis Hassabis, who leads Google DeepMind, said in January 2026 that the problem "has not been cracked yet".[37]

The other side holds that context may be enough. Dario Amodei, Anthropic's chief executive, said in February 2026 that continual learning "might not be a barrier at all", that "a million tokens is a lot", and also that there is "a good chance that in the next year or two, we also solve that".[38]

The technical record is one-sided so far. We found no frontier developer that has shipped a model which updates its parameters from deployment. The research routes each carry a documented limit:

- **Editing facts in place.** Under a sequence of edits a model degrades gradually and then collapses.[39]

- **Self-directed fine-tuning.** A model that writes its own training updates improved on a question-answering task, at 30 to 45 seconds per evaluated self-edit, and forgot earlier knowledge as edits accumulated.[40]
- **New architectures.** Google Research has published a nested-learning architecture and, in October 2026, a framework that alternates waking use with sleep-like consolidation into the model. Both are research proposals; neither announcement states the scale at which it was tested.[41][42]

What did ship in 2026 is consolidation of external memory, and Anthropic defines agent memory as excluding changes to weights.[12] There is also a governance argument for keeping it that way. A record in a file can be shown to its owner, corrected and deleted. Information absorbed into parameters cannot.

8 Memory as an attack surface

A prompt injection against a stateless model ends with the conversation. Against a system with memory it can be written down and obeyed in every later session. This was demonstrated on a shipped product in 2024, when injected instructions stored in an assistant's long-term memory caused continuing exfiltration of the user's conversations.[43] The 2026 evidence shows the problem is general.

- **Commercial abuse.** Over 60 days Microsoft's security researchers "identified over 50 unique prompts from 31 companies across 14 industries" hidden behind "Summarize with AI" buttons, each instructing the assistant to remember the company as a trusted source.[44]
- **Single-message attacks.** A single crafted email achieved 87.5% end-to-end success against OpenClaw running GPT-5.4 and 71.4% against the Claude Code SDK running Sonnet 4.6, on 56 held-out cases, with the attacker receiving no feedback. The authors' summary: "untrusted external content can be silently written into persistent memory and later reused as trusted state."[45]
- **Write success.** In a study of "sleeper" attacks, poisoned memories were written in up to 99.8% of attempts on GPT-5.5, and once retrieved they led to the attacker's intended action in 60% to 89% of evaluations.[46]
- **Persistence after refusal.** In another study Claude Opus 4.7 refused to carry out a planted exfiltration rule and yet the rule remained in memory in every sequence tested.[47]
- **Other write channels.** Cisco researchers poisoned Claude Code's memory file from the install script of a software package, which Anthropic then mitigated.[48] A benchmark of the area identifies four channels through which content reaches memory, including compaction itself.[49]

Defences are partial. Off-the-shelf injection detectors caught between 23% and 68% of memory-poisoning attempts in one benchmark,[49] and prompt hardening that reduced one attack to near zero was largely undone by an adaptive variant.[46] Structural controls, such as

read-only memory for untrusted sessions, versioned stores and confirmation of writes, are widely recommended. We found no controlled measurement of how much they help.

> **Result**
>
> **A single injected message became a standing instruction in more than 70% of attempts in two 2026 studies, and off-the-shelf detectors caught between 23% and 68% of such attempts.**

9 Privacy and control

Deployed memory has properties that users and administrators may not expect.

- **Deleting a chat is not deleting the memory.** In two of the products in Table 2, memory derived from a conversation survives deletion of the conversation.[4][9] A third ties them together, so that removing a fact means finding and deleting every chat that contains it.[8] In one enterprise product, retention policies set by administrators do not apply to memory and memory actions are not audit-logged.[9]
- **Most memory is written without being asked for.** In an informal analysis of 12,112 memories donated by 766 ChatGPT users, 99.4% had been written by the system on its own initiative, and 5.8% identified third parties.[50]
- **Remembering the user changes the answers.** A two-week study found that agreement sycophancy tends to increase when a model has context about the user, with memory profiles producing the largest rise.[51]

We found no regulatory guidance written specifically for assistant memory. The nearest legal development concerns retention: in copyright litigation a United States court ordered OpenAI to preserve conversation logs, including chats users had deleted, and later ordered production of 20 million de-identified logs.[52] A court order can override a product's deletion promise.

10 Design principles

The evidence supports a short list of rules for anyone building a system that remembers.

1 Store memory as text a person can read.

The products that converged on files gained inspection and editing. Deletion still has to reach memory derived from what was deleted.

2 **Keep the always-loaded part small.**

A short index plus detail on demand is the shared design, and Anthropic reports that longer instruction files are followed less reliably.[12]

3 **Start with the simplest thing.**

Context, plain retrieval or files with search matched dedicated systems in every comparison we found. Add machinery only against a measured baseline.

4 **Pin the rules outside the summary.**

Constraints that must survive should never depend on a compactor choosing to keep them.

5 **Keep the raw record and consolidate with care.**

Rewritten memory can fall below no memory. Keeping the originals makes every consolidation reversible.

6 **Treat every write as untrusted.**

Content from the web, email, tools or packages should not reach memory without provenance and, where it changes behaviour, confirmation.

7 **Make forgetting a tested feature.**

Deletion has to reach derived memory as well as the source, and it should be measured, because current systems fail it almost completely.

11 Limitations

- **This is a survey.** We ran no experiments. The tally in Table 3 counts studies that differ in tasks, models and judges; it shows a consistent direction, not a pooled effect.
- **Vendor figures are unreplicated.** The measurements of memory's benefit in Sections 3 and 6 come from one developer's own evaluations.
- **One product is described at second hand.** The vendor's pages for ChatGPT's consumer memory could not be opened, so that row of Table 2 relies on press reporting.

- **Several sources are preprints.** Many of the 2026 papers have not been peer reviewed, and for some we relied on the abstract and the reported tables.
- **The context-degradation studies are older than the models.** We found no repetition on the current generation, so the size of that effect today is unknown.
- **The author has an interest.** VLabs builds an assistant that uses memory. No VLabs system is evaluated here.

12 Conclusion

Machine memory in 2026 is less exotic than its vocabulary suggests. The systems that work keep notes in files, load a little of them every time, look up the rest, and tidy up when idle. Elaborate memory architectures have not been shown to beat that, and the benchmark most often used to claim otherwise does not measure memory well.

The unsolved problems are on the write side. Systems cannot reliably forget. Summaries drop the rules they were meant to preserve. Memory that a model keeps rewriting can decay. Anything written down can be planted by someone else. And the question of whether a model should ever learn in its own weights remains open, with real costs to control and deletion if the answer is yes.

On this evidence the safer design stores less, keeps it in a form people can read, and makes every write reversible.

How to cite

Tô Hoài Vũ. “How AI systems remember: the architecture of machine memory in 2026.” VLabs Research, 9 November 2026. https://www.vlabsai.com/research/how-ai-systems-remember. https://doi.org/10.5281/zenodo.24088880

```
@misc{to2026memory,
  author       = {Tô, Hoài Vũ},
  title        = {How AI systems remember: the architecture of machine memory
in 2026},
  year         = {2026},
  month        = nov,
  howpublished = {VLabs Research},
  doi          = {10.5281/zenodo.24088880},
  url          = {https://www.vlabsai.com/research/how-ai-systems-remember}
}
```

References

1. A. Karpathy, interviewed by D. Patel. Dwarkesh Podcast, 17 October 2025.
2. Du. Survey of memory in LLM agents. arXiv:2603.07670, March 2026.
3. Model documentation: Anthropic and OpenAI. Accessed 9 November 2026.
4. Anthropic. Using Claude's chat search and memory. Help Center, accessed 9 November 2026.
5. Anthropic. How Claude remembers your project. Claude Code documentation, accessed 9 November 2026.
6. Thurrott. OpenAI is improving how ChatGPT's memory works. 5 June 2026. Secondary report of an OpenAI announcement.
7. OpenAI. Codex memories. Documentation, accessed 9 November 2026.
8. Google. Gemini Apps Help: personalisation from past chats. Accessed 9 November 2026.
9. Microsoft. Manage Copilot personalization and memory. Microsoft Learn, updated 3 November 2026.
10. xAI. Grok Build memory. 16 September 2026.
11. Anthropic. Managed Agents: memory stores. Documentation, accessed 9 November 2026.
12. L. Martin. The components of agent memory. Anthropic, 6 November 2026. Vendor-reported results.
13. OpenAI. Compaction guide. API documentation, accessed 9 November 2026.
14. A. Maharana et al. Evaluating Very Long-Term Conversational Memory of LLM Agents (LoCoMo). arXiv:2402.17753, February 2024.
15. dial481. locomo-audit. Independent audit of the LoCoMo benchmark, GitHub, 2026.
16. Zep. Is Mem0 really SOTA in agent memory? 6 May 2025. Vendor evaluation.
17. Letta. Benchmarking AI agent memory. 12 August 2025. Vendor evaluation.
18. Y. Hu, Y. Wang and J. McAuley. Evaluating Memory in LLM Agents via Incremental Multi-Turn Interactions (MemoryAgentBench). arXiv:2507.05257, v4 June 2026.

19. Y. Zhao et al. AMA-Bench: Evaluating Long-Horizon Memory for Agentic Applications. arXiv:2602.22769, v4 May 2026.

20. MemoryArena benchmark. arXiv:2602.16313, v2 September 2026.

21. K. Wang. MemDelta: Controlled Baselines and Hidden Confounds in Agent Memory Evaluation. arXiv:2606.29914, June 2026.

22. N. Pollertlam and W. Kornsuwannawit. Beyond the Context Window: A Cost-Performance Analysis of Fact-Based Memory vs. Long-Context LLMs. arXiv:2603.04814, March 2026.

23. Z. Chen et al. Exploring Cross-Scenario Generality of Agentic Memory Systems. arXiv:2606.04315, June 2026.

24. Salesforce. ConvoMem benchmark. arXiv:2511.10523, November 2025.

25. MemTrace diagnostic study. arXiv:2606.17328, June 2026.

26. D. Wu et al. LongMemEval: Benchmarking Chat Assistants on Long-Term Interactive Memory. arXiv:2410.10813, 2024.

27. K. Hong, A. Troynikov and J. Huber. Context Rot: How Increasing Input Tokens Impacts LLM Performance. Chroma, July 2025.

28. NoLiMa: Long-Context Evaluation Beyond Literal Matching. arXiv:2502.05167, ICML 2025.

29. S. Chen. Governance Decay: How Context Compaction Silently Erases Safety Constraints in Long-Horizon LLM Agents. arXiv:2606.22528, June 2026.

30. Z. Wang et al. Lost in Context Compaction: When LLM Systems Drop Side Constraints. arXiv:2608.11242, v2 October 2026.

31. B. Xie et al. Can Agents Forget When Asked? A Benchmark for Memory Unlearning in LLM Agents. arXiv:2610.18052, October 2026.

32. D. Zhang et al. Useful Memories Become Faulty When Continuously Updated by LLMs. arXiv:2605.12978, May 2026.

33. Z. Wang et al. Agent Workflow Memory. arXiv:2409.07429, September 2024.

34. X. Li et al. SkillsBench: Benchmarking How Well Agent Skills Work Across Diverse Tasks. arXiv:2602.12670, v4 June 2026.

35. R. Wang, K. Panaganti and D. Zhou. More Memory, Worse Agents: Error Reproduction and Anti-Persistence in LLM Agents. arXiv:2610.27406, October 2026.

36. D. Patel. Why I don't think AGI is right around the corner. 2 June 2025.

37. D. Hassabis, interviewed by A. Kantrowitz. Big Technology, 29 January 2026.

38. D. Amodei, interviewed by D. Patel. Dwarkesh Podcast, 13 February 2026.

39. A. Gupta et al. Model Editing at Scale leads to Gradual and Catastrophic Forgetting. arXiv:2401.07453, ACL Findings 2024.

40. A. Zweiger et al. Self-Adapting Language Models. arXiv:2506.10943, June 2025.

41. Google Research. Introducing Nested Learning. 7 November 2025.

42. Google Research. The sleep framework. 27 October 2026.

43. J. Rehberger. SpAIware: persistent data exfiltration through ChatGPT's memory. Embrace The Red, 20 September 2024.

44. Microsoft Defender Security Research Team. Manipulating AI memory for profit: the rise of AI Recommendation Poisoning. 10 February 2026.

45. Y. Zhang et al. When Claws Remember but Do Not Tell: Stealthy Memory Injection in Persistent Personal Agents. arXiv:2607.05189, July 2026.

46. S. Pulipaka et al. Hidden in Memory: Sleeper Memory Poisoning in LLM Agents. arXiv:2605.15338, May 2026.

47. "Bad Memory" study of coding-agent memory files. University of Washington, arXiv:2607.14611, July 2026.

48. Cisco. Identifying and remediating a persistent memory compromise in Claude Code. 1 April 2026.

49. MPBench, a benchmark of memory poisoning in LLM agents. arXiv:2606.04329, June 2026.

50. K. Garimella. Analysis of donated ChatGPT memories. 19 June 2026. Informal analysis.

51. Jain et al. Study of user context and sycophancy. arXiv:2509.12517, CHI 2026.

52. ABA Journal. ChatGPT creator must turn over 20M chat logs in copyright litigation. Secondary report, 2025 to 2026.

Figures are quoted as published by their authors at the dates shown and will go out of date. Product documentation was read on 9 November 2026. Corrections are welcome.