Is Memory the Next Scaling Law?

Artem Shitov, FBCS, SMIEEE

***Abstract*—Modern AI industry heavily relies on rapid AI capabilities scaling with unsaturated industry benchmarks trending towards exponential growth, which in part offsets enterprises caution towards AI even as most AI pilots fail. This expectation that the scaling law of AI capabilities growing rapidly with model size, data size, and training compute time will hold is one of the key things making at-scale venture investing possible even at current AI pilots success rate. But we're seeing AI progress achievable in a practical way via pure compute and data scaling slowing down. This suggests that the next breakthrough may be not in doing even more of the same, but in opening yet another scaling dimension, same as when RL and test-time compute diverged from model size growth. This article explores one of such novel dimension that may be memory. We outline a simple token-budget model for measuring memory's value, discuss design choices for memorization, retrieval, and reintegration, and compare available mechanisms, including retrieval inference integration.**

I. Introduction

Modern AI industry heavily relies on rapid AI capabilities scaling with unsaturated industry benchmarks trending towards exponential growth (e.g., METR [1]). This expectation that the scaling law of AI capabilities growing rapidly with model size, data size, and training compute time will hold is one of the key things making at-scale venture investing possible even at current AI pilots success rate.

But we're seeing AI progress achievable in a practical way via pure compute and data scaling slowing down. Model parameter size for mainstream models peaked with GPT-4 at 1.8T parameters [2].

Llama 4 Behemoth with 2T parameters, announced 2 years after GPT-4 was never released, GPT-4.5, estimated at 5-7T parameter size [3], was only released as a research preview, did not manage to provide a significant performance leap and was discontinued after GPT-5 release.

The new mainstream model post-GPT 4 even shrunk in its size with GPT-4o estimated to be "just" 100-400B [4], not far away from GPT-3, best open source models are also sub-1T like DeepSeek R1 at 671B, Qwen-3-Max-Preview at 1T, or GPT OSS at 120B max.

There are multiple problems inhibiting direct scaling. One is data availability, another is that as scaling intensifies, model size grows at a much faster pace than hardware capabilities, both computation- and memory-wise, while seemingly providing diminishing returns. As an example, while GPT-4 estimated parameter size increased ten-fold vs. GPT-3 over 3 years, accelerator memory capacity only doubled over the same time (see Fig. 1).

To continue evolving the models frontier model developers shifted their scaling from model size and pre-training-time compute to RL- and test-time scaling with CoT [5] where the researchers observed similar scaling law, but at a frontier which was yet unsaturated [6]. DeepSeek needed only $1 mn for a RL training run compared to $6 mn for pre-training of its base DeepSeek V3 model.

Yet reasoning and RL gains are becoming marginal as well with this scaling dimension risking to follow the same path or diminishing returns and compute resource constraints with GPT-5 thinking having much more limited generational progress compared to GPT o3 than GPT o3 had when compared to GPT o1, and even less than GPT o1 had compared to GPT 4o. E.g. by METR long running tasks benchmark, o1 was a 4x increase against 4o, while o3 was a 2.4x increase over o1, and GPT-5 was only a 1.5x gain over o3.

This suggests that the next breakthrough may be not in doing even more of the same, but in opening yet another scaling dimension, same as when RL and test-time compute diverged from model size growth.

This article explores one of such novel dimension that may be memory.

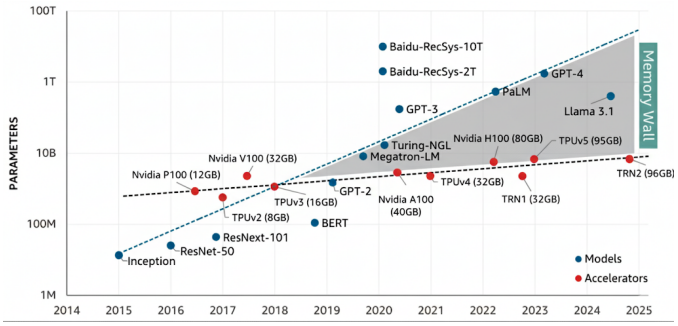

Fig. 1. Model size growth trends compared with accelerator memory growth trend shows that models memory need scale at much higher and accelerating pace requiring more accelerators, FLOPs, and more complex / expensive infrastructure configurations with high-bandwidth networking [7].

Current large-scale foundation models show limited ability to transfer their achievements across challenging benchmarks or even multiple runs of the same benchmark. As companies pushed against ARC-AGI improving their scores, even the best models underperformed as novel ARC-AGI 2 was released, and even more so with ARC-AGI 3 even as both are easily completed by average humans.

The same is true for other domains, with GPQA being all but saturated due to increased optimization efforts with models claiming "graduate-level knowledge" and high Humanity's Last Exam results putting claim onto PhD-level thinking. Yet this graduate or PhD-level dissipates in the new "Radiology's Last Exam" [8], for which the models were not optimized, and where they perform much worse than radiologist trainees.

And while frontier models are now capable of running longer and longer tasks [1], they still cap out with GPT-5 leading with a median task duration of just 24 minutes at 80% success rate, with a very wide 95% CI of 9 minutes to 1 hour.

The problem is that models don't learn until trained by the frontier researchers or fine-tuned via advanced practitioners with tailor-made pipelines.

Recent AI evolution mirrors a student who completely forgets material from previous courses when starting a new one or even when re-doing the same course. Despite extensive training, these models effectively erase their working memory with each new conversation or task, treating every interaction as if it were their first exposure to the topic. No matter how often you ask Claude Sonnet about details on recent RL developments, it will still take it's time to research it from scratch, unless you're continuing the previous context window.

This fundamental limitation, where knowledge remains in static weights rather than being dynamically updated, severely restricts meaningful adaptation.

Currently when you call GPT via API or in a Web UI, a new cloned session is born that perceives your question as if it sees it for the first time model weighs static across runs. KV cache is the closest thing to this session's transient essence. At best, the RAG system contextualizes the agent with snippets of previous conversation or potentially relevant factoids before it starts to process the input, giving it some additional pretext and semblance of cross-session memory.

As a scaling dimension memory is currently mostly untapped as we will see below with even current research implementations showing substantial double digits improvement potential (e.g., [9]).

II. Scaling

Memory can be considered a mechanism that allows to reuse already expended tokens between different inferences thus introducing test-time compute scaling without incurring test-time cost repeatedly (Fig. 2). Potentially, it allows to even pre-process and memorize necessary tokens ahead of time (sleep-time compute [10]) and pre-compute optimal processing patterns [11].

It can be directly observed in a long LLM chat on a given subject matter, where the model could have uncovered a lot of interesting observations using CoT and tools (like search) and given user steering could contextualize them in important insights relevant to user's field of interest. In the same chat thread these tokens are re-used allowing to deepen topic understanding rather than doing every inference from scratch and re-explaining user's interest to the model. Good memory system would allow to reuse these same tokens across multiple threads or tasks rather than doing the same CoT over and over and running overlapping tool requests across sessions.

If in the prior inference the system spent T_i tokens and $m_i T_i$ of them were about inferring domain-general knowledge (like searching for SMR data and reasoning on key comparison dimensions between different projects), the next inference within the same domain or topic may either reuse these tokens for free improving latency as it won't need to re-invent the prior observations it inferred, or spend additional $m_{i+1}T_{i+1}$ tokens to deepen the reasoning thus increasing output quality.

In other words, if we assume that for each inference we allocate a specific "juice" budget T_i, and $m_i T_i$ of this budget is actually spend on domain-general reasoning, whereas the remainder is task-specific reasoning tokens and output, in the perfect world we would have:

$$E(x_n, S_n) = \sum_{i=1}^{n-1} m_i T_i + m_n T_n + (1 - m_n) T_n \tag{1}$$

$$B(x_n, S_n) = m_n T_n + (1 - m_n) T_n = T_n \tag{2}$$

, where $E(x, S_n)$ is token-equivalent cost of reaching the same output quality without memory (as if we had to do all reasoning from scratch), given S_n state of existing memory accumulated over n inferences, and $B(x, S_n)$ is the actual tokens generated within budget, i.e. actual inference token cost.

In practice, however, we will manage to recall only part of relevant data $\mathcal{R}(m_i T_i, S_i, x)$, where $m_i T_i$ is the memorized result of ith inference in domain-general tokens and S_i is the current state.

And we will be only able to utilize $\mathcal{U}(m_i T_i, S_i, x)$ efficiency of result quality impact resulting in diminishing cross-domain token usage as we spend more domain-tokens on insight generation (e.g., if we re-insert recovered tokens verbatim, overpopulated context window could reduce inference efficiency, reducing $\mathcal{U}$; if we re-insert extracted summaries / insights, our summarization process may mis-represent details important for given prompt, even as we correctly surface the memory, thus reducing $\mathcal{U}$).

Furthermore, even with T_n inference budget we will not necessarily be able to use it to the full extent given previous insights reuse resulting in only $\mathcal{M}(T_n, S_n, x_n)$ generated cross-task tokens suitable for memorization and $\mathcal{N}(T_n, S_n, x_n)$ generated task-specific tokens. For example, if we already have comprehensive memory bank on a given topic, we could not effectively spend large reasoning topic budget since we do not have sufficient additional research vectors.

So, the actual impact would be:

$$\begin{aligned} E(x_n, S_n) = \sum_{i=1}^{n-1} \Big[(\mathcal{U} \circ_1 \mathcal{R})(\mathcal{M}(T_i, S_i, x_i), S_n, x_n) \Big] \\ + \mathcal{M}(T_n, S_n, x_n) + \mathcal{N}(T_n, S_n, x_n) \end{aligned} \tag{3}$$

$$B(x_n, S_n) = \mathcal{M}(T_n, S_n, x_n) + \mathcal{N}(T_n, S_n, x_n)) \tag{4}$$

$\mathcal{R}$ can be measured as a recall rate, for example via evals injecting relevant memories with synthetic data and measuring share of surfaced memories given target prompt. $\mathcal{U}$ is a deviation of actual output quality from the quality suggested by the test-time compute scaling as if we had spent the actual token budget. $\mathcal{M}$ and $\mathcal{N}$ are factual parameters of share of

generated tokens out of the allowed token budget for a given inference response.

The practical memory system should focus on optimizing the $\mathcal{U}$ and $\mathcal{R}$ function shapes, while an additional memorized token budget is the basic scaling parameter, as this budget gets reused for future inferences rather than being discarded.

To give a specific example, let us assume in previous inferences we discovered (e.g., using search, assuming we did not have prior explicit knowledge) that a) Paris is the capital of France, b) Eiffel Tower is in Paris, c) Eiffel Tower is made out of puddled iron, d) Eiffel Tower has elevators, e) these elevators are made out of steel with all of these facts being part of S_n with B_{n-1} tokens spent to uncover them over multiple inferences.

Given the prompt "What materials does French capital tower contain?", an imperfect memory could surface only facts a–c considering d–e irrelevant (hence $\mathcal{R}$ coefficient being < 1), then the actual inference could miss the part about iron being "puddled iron" as being unimportant further reducing efficiency of re-surfaced memories (hence $\mathcal{U}$ coefficient being < 1).

We still saved some of the token budget of our final inference as we re-used a–c memories, even if in imperfect form, and can now use this freed up tokens that we reused rather than generated to either improve latency or generate additional reasoning tokens improving answer quality.

Thinking of memory in tokens re-used allows for simpler concept to evaluate with direct scaling implications, thus making the concept more practical and allowing for direct optimization levers in memory-enabled systems.

III. Memory structure

Memory mechanics can be sub-split into 3 parts: memorization, retrieval (recognition & recall), and reintegration (see Fig. 3) (e.g., synthesizing [12]). This text focuses on memorization and recall components.

Memorization is the process of *encoding*, *consolidating*, and *storing* memories, making sure vital parts of knowledge are retained and correctly cross-referenced while non-essential parts are evicted or forgot to avoid over-burdening the memory store and overcrowding knowledge graph with minute details.

Retrieval is the process of *recognizing* the availability of relevant knowledge on trigger and *recalling* said knowledge from memory in accurate, precise, and timely manner making stored information accessible and contextualized.

Reconsolidation is the process of *reorganizing* existing memories based on new data and *forgetting* irrelevant details or facts.

A. Memorization

Human memory is tiered, with a lot of components not having direct representation in modern production-grade LLM systems in dynamic form. Even as they do have a vague analogous presence even in much more basic forms of compute, like your typical computer.

There are clear gaps in intermediate and long-term memory with static substrates available to serve the function, creating substantial limitations.

While external memory, such as vector DB-based RAG systems, provides a partial alleviation to an extent, it also is lossy as it focuses on saving verbatim data, mostly external to the model thinking process, which is discarded in its entirety with modern LLM systems, creating an uncanny phenomenon of models not really knowing how they arrived at the conclusions they stated to the user.

That's not how humans work though. Humans do recall the high-level understanding of their reasoning chain, even if non-verbatim, and can exchange it.

ARC-AGI 3 is a good illustration of where models typically fail completely right now. ARC claim that humans pass 100% of the games, yet they don't mention pass@n, while at least some people actually have to try a few times first to understand the gist of the game, the "no explicit rules stated" aspect, and learn to direct their effort properly before they try to solve it. And once they solve the first game, they get the second one faster, exactly because they retain the knowledge they gained while solving the games instead of discarding it.

Models, on the contrary, rediscover their thinking process every turn or quickly pollute their context window running out of memory.

Some tools, like Manus, try to circumvent this intermediate and long-term memory gap by creating intermediate external memory artifacts like files with to-do lists or current research topic synthesis, but these have limited effect as these artifacts are not integrated, still have to be ingested into the context window verbatim and on their entirety while their organization and recall follows static workflows.

A more robust and fine-grained approach would be to make the model state less static with KV- or weights-editing, where multiple approaches currently exist (see Table II):

Systems, like MemOS [23] attempting to address memory limitations by implementing a hierarchical memory system that mirrors the aforementioned human memory structures and has separate memory management processes responsible for refined knowledge graph updates and even parametric tuning to some extent. This architecture maintains episodic records while extracting semantic knowledge, allowing models to build persistent knowledge bases across sessions.

B. Retrieval

Storing knowledge alone is only a part of the process with knowledge *recognition* (i.e. understanding we have some knowledge) and *recall* (i.e. extracting specific knowledge) components which should be a) accurate, or including knowledge representative enough of real data (similar to ISO/IEC 25012), b) precise, or including knowledge relevant to the given task while avoiding excessive unrelated knowledge, and c) timely, or available when needed without excessive delays.

Available memory sources limit accuracy and have substantial impact on precision potential (see Table III).

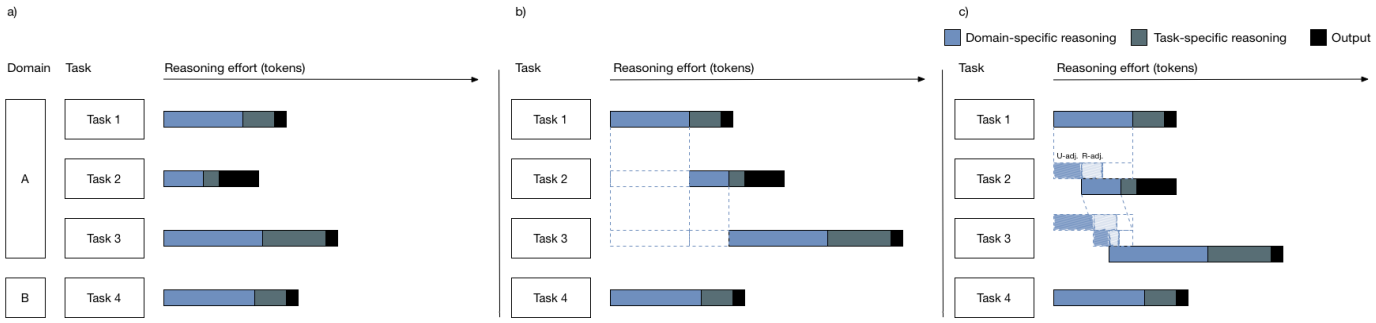

Fig. 2. a) no memory, all tasks execute independently, b) perfect memory implementation, task can reuse insights extracted from previously generated tokens for cross-task knowledge, c) imperfect memory implementation when some of the pre-generated tokens are not reused due to recall-issues (R-adjustment) or utilization issues (U-adjustment)

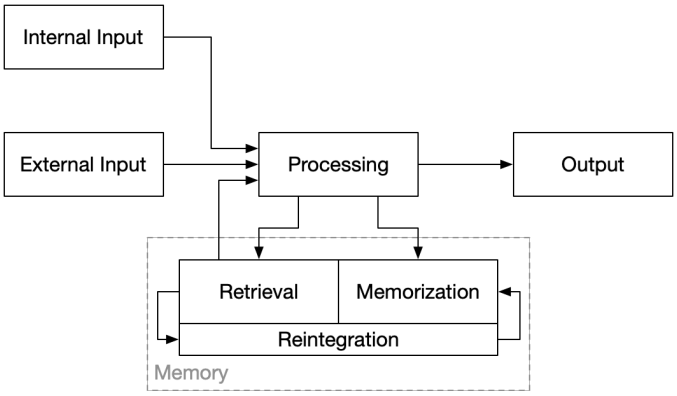

Fig. 3. Memorization and retrieval are two distinct processes where the former is responsible for persisting and organizing knowledge whereas the latter is focused on efficient and timely retrieval; reintegration focuses on background processing of the memory pool and integrating new memories with existing links and concepts adjusting prior knowledge if needed.

Once the target sources are defined, the recall process has to be integrated into overall inference chain, which can happen in one of 4 modes (see Fig. 4):

Ahead-of-time: before the main inference starts there is a separate retrieval process, most classic RAG applications follow this pattern, incl. early ChatGPT or Grok implementations prepending search call results and extracted memory factoids

This approach is the easiest to implement, but has poor timeliness as it can only surface data relevant to the original query, but as inference reasoning goes on, it fails to capture additional data points which become relevant as the process evolves.

Just-in-time: explicit direct blocking tool calls during inference orchestrated by the model, most modern tool-calling LLMs follow this approach, especially prominent for frontier reasoning models (GPT o3, GPT-5 Thinking)

Such integration allows to extract additional data points, but is limited in the model having to pre-empt the existence of target data to consider query potential and in the amount of queries as each query introduces latency impact.

Asynchronously: explicit async tool calls during inference orchestrated by the model, some frontier models follow this approach fanning out multiple async calls, like GPT-5 Thinking, or doing delayed asynchronous recall like GPT Real-Time

This approach softens latency impact from multiple queries and reduces pressure from long-running queries, but it still requires for the model to have some pre-existing assumption about availability of some data or a very vivid knowledge gap that could prompt attempt of additional retrieval and requires the model to have good planning strategy first task front-loading.

Continuously: background recall without need for direct tool calls with knowledge injected pro-actively into context window (e.g., appended to avoid invalidating KV-cache), recall is orchestrated by external system, currently absent in production models to the best of my knowledge.

This approach allows for potentially better timeliness as recall happens in parallel constantly trying to check for new relevant data points even if model does not know about their existence of its knowledge gap.

IV. Conclusion

Memory is a widely untapped dimension as of now with substantial gaps across some of the major components allowing for resource-effective scaling, while providing practical benefits that make scaling investment worthwhile and potentially allow to solve some of the major structural problems of current models, like attuning to specific tasks and domains if exposed to them repeatedly and collecting more complex knowledge associations.

We will likely see efforts at implementing more comprehensive tiered memory systems with per-user parametric augmentations or tuning as well as more continuous background recall process allowing for surfacing data without model having to pollute context window with detailed data structure and semantics details. As an additional evolution point, this continuous recall coupled with in-context memorization may unlock much longer virtual context windows pushing older context items into tiered memory and recalling relevant excerpts as needed by the current process steps.

Memory has a potential to be the next breakthrough and it's important to pay close attention to latest research trends and frontier developers motions in this direction. For startups

TABLE I
MEMORY LAYERS COMPARISON

Memory Layer	Human	Computer	Foundation models mechanism
Sensory memory [13]	Ultra-short and ultra-wide context of sensory inputs, lasts for up to a second. Works via residual activity in early sensory cortices	I/O buffers (e.g., frame buffer)	Early encoder activation
Short-term memory [13]	Contains 3-6 active concepts at a time, lasts for 20-30 seconds unless rehearsed. Works via persistent pre-frontal context activity and short-term synaptic facilitation	CPU registers	Top-attended tokens
Working memory [14]	Keeps active context and reasoning thread that defines the current activity, lasts seconds to minutes. Works via prefrontal cortex-basal ganglia-thalamus loops and oscillatory coding	L1/L2 cache	Context window, KV-cache
Intermediate memory [15]	Bridge from working to long-term memory saving important information for hours for further processing. Works via early-phase LTP	RAM	N/A
Declarative long-term memory [13]	Long-term facts and events storage and retrieval. Works via late-phase LTP with cortical consolidation via hippocampal indexing	HDD/SSD (data)	N/A. Pre-trained weights serve this role, but they are static
Non-declarative long-term memory [13]	Skills, habits, biases, and overall procedural approaches to thinking. Works via dopamine-gated corticostriatal plasticity and cerebellar LTP	HDD/SSD (compiled code, configuration)	N/A. RLHF/RLAIF-shaped policies serve this role, but they are static
External memory	Allows for keeping verbatim data for further referencing and processing avoiding lossy biological memory processing. Duration: depends on the medium, up to lifetime	External resources (e.g., web)	Vector database RAG, tool use (search, web access, database queries)

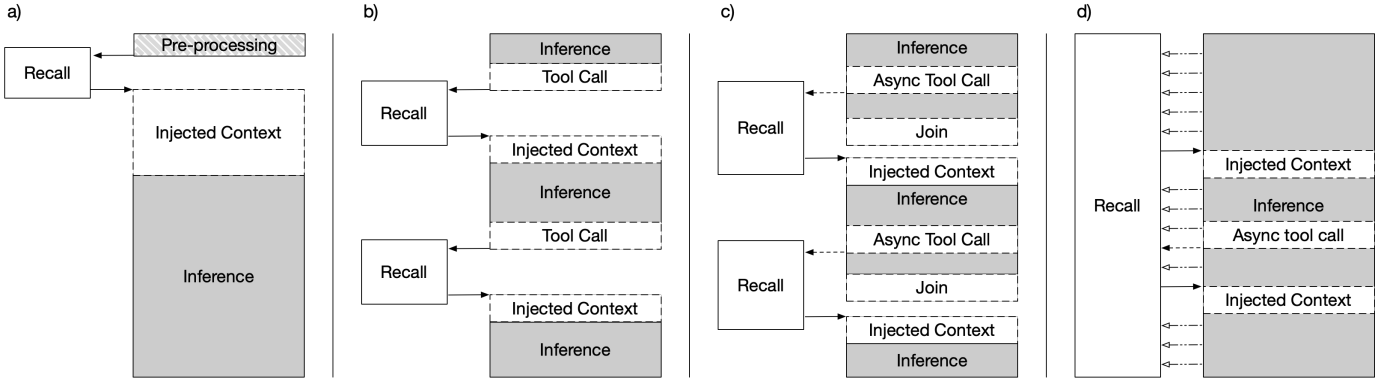

Fig. 4. a) ahead-of-time recall; b) just-in-time recall; c) asynchronous recall; d) continuous recall. Solid arrow — blocking call; dashed arrow — non-blocking call; dot-dashed white arrow — background mirroring.

this opens an opportunity to innovate and disrupt same way DeepSeek R1 disrupted the market with quick adoption of frontier RL and CoT approach allowing to gain massive new dimension unlock impact.

By addressing the current memory limitations, we can move closer to AI systems that truly understand and adapt to the world around them, rather than simply processing isolated inputs. The journey toward more human-like memory in AI is not just about mimicking biological processes but extracting the core concepts and translating them to model specifics creating systems that can maintain context, learn continuously, and develop a persistent understanding of the world - ultimately making AI more useful, reliable, and aligned with industry practitioners' needs.

TABLE II
EDITING TYPES FOR MODEL STATE

Editing type	Examples	Scope	Limitations	Memory Layer
Architecture-level Memory Integration	LongMem, Recurrent/Associative Memory Transformers [16]–[18]	Extending context window into transient memory integrated into model architecture without adjusting base weights	Provides limited capacity and does not allow for generalization and late linking	Intermediate Memory
Narrow-Scope Knowledge Injection	ROME, MEMIT, MEND [19], [20]	Injecting narrow-scope knowledge into an existing model	Can only edit specific knowledge nuggets, does not scale well to training across wider concepts	Declarative Long-Term Memory
Wide-Scope Knowledge Injection	DAPT, TAPT; EWC for avoiding catastrophic forgetting [21], [22]	Injecting domain- or task-specific data set into an existing model without affecting its architecture with EWC used to avoid forgetting at other domains / tasks	Does not allow for complex knowledge storing (e.g., novel API with ability to write in it, not just be competent in domain specific terminology)	Declarative Long-Term Memory
Procedural Training	TTT, PEFT (via LoRA, QLoRA, DoRA, etc)	You want to adjust model behavior and biases without injecting new knowledge	Does not allow to inject net new knowledge aside from adjusting behavioral patterns	Non-Declarative Long-Term Memory
Non-Parametric	RAG, REALM, RETRO	You want to inject specific knowledge into the model's context window	Does not integrate disparate knowledge entries allowing only for surfacing direct associations	External Memory

REFERENCES

[1] METR, "Measuring ai ability to complete long tasks," https://metr.org/blog/2025-03-19-measuring-ai-ability-to-complete-long-tasks/, March 2025.

[2] D. Patel and G. Wong, "Gpt-4 architecture, infrastructure, training dataset, costs, vision, moe," SemiAnalysis, July 2023. [Online]. Available: https://semianalysis.com/2023/07/10/gpt-4-architecture-infrastructure/

[3] N. Lambert, "GPT-4.5: 'Not a frontier model'?" Interconnects, February 2025.

[4] Epoch AI, "Frontier language models have become much smaller," December 2024.

[5] OpenAI, "Learning to reason with LLMs," https://openai.com/index/learning-to-reason-with-llms/, September 2024.

[6] R. Shao, J. He, A. Asai, W. Shi, T. Dettmers, S. Min, L. Zettlemoyer, and P. W. Koh, "Scaling retrieval-based language models with a trillion-token datastore," 2024. [Online]. Available: https://arxiv.org/abs/2407.12854

[7] R. Singh, A. Rathi, and G. McShane, "A close look at how amazon built the nova foundational models using sagemaker hyperpod," 2024.

[8] D. Datta, "Radiology's Last Exam," https://x.com/drdatta_aiims/status/1954586822849523789, 2025.

[9] J. Hu, Z. Zhang, G. Chen, X. Wen, C. Shuai, W. Luo, B. Xiao, Y. Li, and M. Tan, "Test-time learning for large language models," 2025. [Online]. Available: https://arxiv.org/abs/2505.20633

[10] K. Lin, C. Snell, Y. Wang, C. Packer, S. Wooders, I. Stoica, and J. E. Gonzalez, "Sleep-time compute: Beyond inference scaling at test-time," *arXiv preprint arXiv:2504.13171*, 2025. [Online]. Available: https://arxiv.org/abs/2504.13171

[11] A. Didolkar, N. Ballas, S. Arora, and A. Goyal, "Metacognitive reuse: Turning recurring llm reasoning into concise behaviors," *arXiv preprint arXiv:2509.13237*, 2025, arXiv:2509.13237. [Online]. Available: https://arxiv.org/abs/2509.13237

[12] R. Bisaz, A. Travaglia, and C. M. Alberini, "The neurobiological bases of memory formation," *Frontiers in Behavioral Neuroscience*, vol. 8, 2014.

[13] R. C. Atkinson and R. M. Shiffrin, "Human memory: A proposed system and its control processes," in *The psychology of learning and motivation: Advances in research and theory*, K. W. Spence and J. T. Spence, Eds. Academic Press, 1968, vol. 2, pp. 89–195.

[14] A. Baddeley, "Working memory: Theories, models, and controversies," *Annual Review of Psychology*, vol. 63, pp. 1–29, 2012.

[15] J. Kamiński, "Intermediate-term memory as a bridge between working and long-term memory," *Journal of Neuroscience*, vol. 37, no. 20, pp. 5045–5047, 2017.

[16] D. Wang, L. Cheng, M. Yan, J. Gao, and F. Wei, "Longmem: Long-term memory for large language models," in *NeurIPS*, 2023.

[17] A. Bulatov, Y. Kuratov, and M. Burtsev, "Recurrent-associative-memory-transformers," in *NeurIPS*, 2022.

[18] A. Rodkin, Y. Kuratov, A. Bulatov, and M. Burtsev, "Associative memory in transformers," in *ICML workshop*, 2024.

[19] K. Meng, A. Sen Sharma, A. Andonian, Y. Belinkov, and D. Bau, "Mass-editing memory in a transformer," in *ICLR*, 2023.

[20] E. Mitchell, C. Lin, A. Bosselut, C. Finn, and C. D. Manning, "Fast model editing at scale (mend)," in *ICLR*, 2022.

[21] S. Gururangan *et al.*, "Don't stop pretraining," in *ACL 2020*, 2020.

[22] J. Kirkpatrick *et al.*, "Overcoming catastrophic forgetting in neural networks," *PNAS*, 2017.

[23] Z. Li, S. Song, C. Xi, H. Wang, C. Tang, S. Niu, D. Chen, J. Yang, C. Li, Q. Yu, J. Zhao, Y. Wang, P. Liu, Z. Lin, P. Wang, J. Huo, T. Chen, K. Li, Z. Tao, H. Lai, H. Wu, B. Tang, Z. Wang, Z. Fan, N. Zhang, L. Zhang, J. Yan, M. Yang, T. Xu, W. Xu, H. Chen, H. Wang, H. Yang, W. Zhang, Z.-Q. J. Xu, S. Chen, and F. Xiong, "Memos: A memory os for ai system," *arXiv*, 2025. [Online]. Available: https://arxiv.org/abs/2507.03724

TABLE III
MEMORY SOURCES COMPARISON

Memory Source Type	Accuracy	First-Order Precision	Next-Order Precision	Prevalence
Parametric weights: direct associations inferred from learned weights of the model	Low to Medium: hallucination prone, hard to store verbatim data and control exact recall	High	High	Dynamic parametrization is absent in most implementations
Vector database: searching for data items semantically relevant to input, it's similar in essence to external data source query, but relies on vector distance for data retrieval	High: verbatim recall	Medium to High: potential issues with vector embedding may lead to missing some relevant data	Low to Medium: focuses on surfacing direct connections, but fuzzy nature of the search may find more disparate connections	Widely used in modern RAG systems
External database: using tools to query an external system (e.g., SQL database or an API) to extract specific data items verbatim	High: verbatim recall	Low to High: depends on model knowledge of data structure and data points and ability to build effective query	Low: responds to an exact query thus not affecting related queries	Increasing propensity with tool integrations and MCP
Knowledge graph: retrieving data directly or indirectly associated with input and other recall fragments based on multiple criteria (such as related terms, recency, etc)	Medium to High: incorrect knowledge links may lead to false associations resulting in incorrect data assumptions	High: explicit knowledge pieces are linked to the graph nodes	Medium to High: graph format allows to overlay and lookup indirectly connected concepts	Almost absent in modern systems