AGENTIC MEMORY SHOULD LOCALIZE COMPRESSION

Izaaz Inhar
KAIST
mizaazir2@kaist.ac.kr

ABSTRACT

Long-horizon LLM agents require memory, but unbounded storage is unusable at inference time, making compression unavoidable. In continual deployment, compression becomes repeated updates to accessible state and can induce behavioural drift on previously supported queries. We formalize this as *interference*: expected divergence between the agent's policies before and after an update. Our position is that stability is governed by retrieval–update overlap; modular designs minimize overlap and thus localize update effects. Under routing stability, expected interference is controlled by the probability that updated modules are retrieved.

1 INTRODUCTION

Agentic systems that operate over long time horizons inevitably require memory to store new factual knowledge and experiences. Unlike a model's static parametric memory (its weights), this takes the form of explicit working memory and long-term storage acting as a context management tool (Hu et al., 2026). However, naive append-only memory is untenable: storing everything grows without bound, while repeated summarization accumulates omissions and errors over rewrites (Hong et al., 2025). This necessitates compression as a main requirement for agentic memory.

Recent work has explored powerful compression mechanisms (e.g., KV-cache compression, token-latent memory, etc.) (Eyuboglu et al., 2025; Song et al., 2025; Xu et al., 2025; Sun et al., 2025), but these methods are typically optimized for efficiency and are rarely designed or evaluated for the continual-update regime, where an agent's memory is updated repeatedly over time. In this setting, compression and consolidation can cause *interference*, which we define as measurable changes in behaviour on previously solved queries after memory updates, manifesting as regression in performance, drift, or even catastrophic forgetting (Luo et al., 2025).

We argue that the key design question is not whether to compress, but where compression lives. Our position is a modularity-first memory layer: represent memory independently as updatable modules (each with its own storage, compression/forgetting policy, and access boundary) with explicit retrieval and composition interfaces, and apply compression locally within modules rather than as a single global summary. Local updates then affect behaviour primarily through the subset of queries that retrieve updated modules, reducing interference while still enabling aggressive compression where needed.

2 MODULAR COMPRESSION

2.1 THE CASE FOR COMPRESSION

Compression is a structural requirement for long-horizon agents because the binding constraints are not raw storage but *usable* memory at inference: bounded context length, bounded retrieval/ranking budget, and bounded capacity to select and compose evidence (Kang et al., 2025; Zhang et al., 2025). As the memory corpus grows, the marginal utility of storing another raw item decreases while the cost of finding and packaging the right context increases. Even with perfect storage, uncompressed memory enlarges the search space, increases near-duplicate clutter, and raises the probability that retrieval returns weakly relevant or redundant context (Liu et al., 2026). Thus, without compression, agents face an inevitable degradation in latency and reliability due to retrieval and context-construction overhead (Bini et al., 2025).

Compression should therefore be viewed as part of the memory *lifecycle*, not a one-time optimization. In continual deployment, new experiences arrive continuously and must be integrated into a bounded memory interface through operations such as consolidation (merging and abstraction), eviction/forgetting (budget enforcement), and schema alignment (bringing heterogeneous traces into retrievable forms) (Bousetouane, 2026; Logan, 2026). Each of these operations is an update to the agent's accessible state: it changes what can be retrieved and how it is presented to the policy. This perspective turns memory compression into a continual-update problem, where the central question becomes how to schedule and structure updates so that memory remains compact and behaviour remains stable under repeated revisions, setting up our modular interference view in the next subsection.

2.2 MODULARITY

Interference Control. In continual deployment, memory is repeatedly modified by consolidation, removal, and addition of information (Latimer et al., 2025). Despite fixed model weights, these memory updates can introduce behaviour changes by changing the effective context the policy is conditioned on. Therefore we defined interference as update-induced behavioural drift as

$$\Delta_t(Q) = \mathbb{E}_{q\sim Q}[D(\pi_t(\cdot \mid q) \parallel \pi_{t+1}(\cdot \mid q))]$$

for any divergence D and a question distribution Q that has already been consolidated in time step t. This definition utilizes divergence as it is a task-agnostic stability metric. We argue that the main driver for interference is coupling: when the memory representation is monolithic, any compression or rewrite potentially changes the evidence available to many unrelated queries. Modularity reduces this coupling by localizing updates.

Consider a memory module set M_t and an update event that modifies only a subset of memory modules U_t. For a query q, let $R(q, M_t)$ denote the set of modules retrieved to answer q. Define the retrieval-update overlap probability $\rho_t = \Pr_{q\sim Q}(U_t \cap R(q, M_t) \neq \emptyset)$. Under a standard locality assumption, which requires stable routing, and a bounded-change assumption for queries that do overlap, Proposition 1 shows that expected interference is controlled by overlap: $\Delta_t(Q) \leq \rho_t \epsilon_t$ (proof in Appendix A). It follows that with a single memory store ($K = 1$) that is consulted for most queries, any non-trivial update yields $\rho_t \approx 1$, making interference difficult to localize.

Implications for modular compression design. This framing yields concrete design requirements for memory compression in continual agents. Here we define a *module* as an independently updatable memory store with a scoped access distribution and an explicit retrieval/composition boundary.

1. **Local Compression with Update Isolation.** Compression should be performed within a module using objectives tied to that module's supported query set. Modules should have independent lifecycle policies (retain/merge/forget) and update schedules, enabling targeted maintenance. As each module is self-contained, self-distillation is allowed to be aggressive. This avoids global rewrites that entangle unrelated behaviours and makes it possible to update "active" modules frequently while leaving stable modules untouched, limiting drift on older tasks (Jiang et al., 2026).
2. **Sparse Routing.** Retrieval should be designed so that typical queries will only depend on a small subset of modules. Updating a small number of modules affects a correspondingly small portion of the query distribution. Practically, this can be implemented via task- or topic-conditioned routing, gating networks, clustering by usage, or explicit module ownership rules. To reduce silent misrouting failures, use confidence gating with a broad fallback retrieval path when the router is uncertain.
3. **Explicit Composition Interface.** Modularity only helps if cross-module composition is treated as a first-class interface: multi-module queries should retrieve and compose the relevant modules, and derived connecting information can be stored as its own module or as explicitly referenced links rather than being silently blended into a global summary.

This suggests evaluating memory systems not only on compression rate and task success, but also on stability under repeated updates by measuring overlap ρ_t, interference $\Delta_t(Q)$ alongside task-level regression and action/tool-call drift, and cross-module composability on multi-module queries. A strong modular design should concentrate behaviour changes in the small subset of queries that retrieve updated modules, while leaving the remainder of the suite stable.

REFERENCES

Massimo Bini, Ondrej Bohdal, Umberto Michieli, Zeynep Akata, Mete Ozay, and Taha Ceritli. MemLoRA: Distilling expert adapters for on-device memory systems, 2025. URL https://arxiv.org/abs/2512.04763.

Fouad Bousetouane. AI agents need memory control over more context, 2026. URL https://arxiv.org/abs/2601.11653.

Sabri Eyuboglu, Ryan Ehrlich, Simran Arora, Neel Guha, Dylan Zinsley, Emily Liu, Will Tennien, Atri Rudra, James Zou, Azalia Mirhoseini, et al. Cartridges: Lightweight and general-purpose long context representations via self-study. *arXiv preprint arXiv:2506.06266*, 2025.

Kelly Hong, Anton Troynikov, and Jeff Huber. Context rot: How increasing input tokens impacts LLM performance. Research report, Chroma, July 2025. URL https://research.trychroma.com/context-rot.

Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, Senjie Jin, Jiejun Tan, Yanbin Yin, Jiongnan Liu, Zeyu Zhang, Zhongxiang Sun, Yutao Zhu, Hao Sun, Boci Peng, Zhenrong Cheng, Xuanbo Fan, Jiaxin Guo, Xinlei Yu, Zhenhong Zhou, Zewen Hu, Jiahao Huo, Junhao Wang, Yuwei Niu, Yu Wang, Zhenfei Yin, Xiaobin Hu, Yue Liao, Qiankun Li, Kun Wang, Wangchunshu Zhou, Yixin Liu, Dawei Cheng, Qi Zhang, Tao Gui, Shirui Pan, Yan Zhang, Philip Torr, Zhicheng Dou, Ji-Rong Wen, Xuanjing Huang, Yu-Gang Jiang, and Shuicheng Yan. Memory in the age of AI agents, 2026. URL https://arxiv.org/abs/2512.13564.

Dongming Jiang, Yi Li, Guanpeng Li, and Bingzhe Li. MAGMA: A multi-graph based agentic memory architecture for AI agents, 2026. URL https://arxiv.org/abs/2601.03236.

Minki Kang, Wei-Ning Chen, Dongge Han, Huseyin A. Inan, Lukas Wutschitz, Yanzhi Chen, Robert Sim, and Saravan Rajmohan. ACON: Optimizing context compression for long-horizon LLM agents, 2025. URL https://arxiv.org/abs/2510.00615.

Chris Latimer, Nicoló Boschi, Andrew Neeser, Chris Bartholomew, Gaurav Srivastava, Xuan Wang, and Naren Ramakrishnan. Hindsight is 20/20: Building agent memory that retains, recalls, and reflects, 2025. URL https://arxiv.org/abs/2512.12818.

Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. SimpleMem: Efficient lifelong memory for LLM agents, 2026. URL https://arxiv.org/abs/2601.02553.

Joe Logan. Continuum memory architectures for long-horizon LLM agents, 2026. URL https://arxiv.org/abs/2601.09913.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *IEEE Transactions on Audio, Speech and Language Processing*, 33:3776–3786, 2025. doi: 10.1109/TASLPRO.2025.3606231.

Charlie Victor Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context, 2023. URL https://openreview.net/forum?id=am22IukDiKf.

Xinshuai Song, Weixing Chen, Yang Liu, Weikai Chen, Guanbin Li, and Liang Lin. Towards long-horizon vision-language navigation: Platform, benchmark and method, 2025. URL https://arxiv.org/abs/2412.09082.

Weiwei Sun, Miao Lu, Zhan Ling, Kang Liu, Xuesong Yao, Yiming Yang, and Jiecao Chen. Scaling long-horizon LLM agent via context-folding, 2025. URL https://arxiv.org/abs/2510.11967.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-Mem: Agentic memory for LLM agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=FiM0M8gcct.

Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, Jian Xie, Yuxuan Sun, Boyu Gou, Qi Qi, Zihang Meng, Jianwei Yang, Ning Zhang, Xian Li, Ashish Shah, Dat Huynh, Hengduo Li, Zi Yang, Sara Cao, Lawrence Jang, Shuyan Zhou, Jiacheng Zhu, Huan Sun, Jason Weston, Yu Su, and Yifan Wu. Agent learning via early experience, 2025. URL https://arxiv.org/abs/2510.08558.

A FORMALIZATION OF THE MODULARIZED OBJECTIVE

Let $\Pi(\cdot \mid C,\ q)$ denote the base model's next token distribution given full context C and a query q. Let Z be a compressed memory object (latent memory, compressed KV cache, etc) and $\Pi_z(\cdot \mid Z,\ q)$ be the model augmented with Z. We can use context distillation (Snell et al., 2023) as a simple explicit objective to align these behaviours

$$\min_{Z:size(Z)<B} \mathbb{E}_{q\sim Q}[D(\Pi(\cdot \mid C,\ q)) \parallel \Pi_z(\cdot \mid Z,\ q)]$$

with a maximum context size B. Now, we can introduce modularity as a library of memory modules $\{Z_k\}_{k=1}^K$ and a sparse retrieval policy $R(q) \subseteq [K]$ to modify our objective to

$$\min_{Z:size(Z)<B} \mathbb{E}_{q\sim Q}[D(\Pi(\cdot \mid C,\ q)) \parallel \Pi_z(\cdot \mid \ \oplus_{i\in R(q)} Z_i,\ q)] + \lambda \mathbb{E}_{q\sim Q}[|R(q)|]$$

Proposition 1 *Let M_t denote the memory set of an agent Π at time t , e.g.*

$$M_t := \{Z_1^t, \ldots, Z_K^t\}$$

Given a query q and a retrieval scheme $R(q, M_t) \subseteq [K]$ that returns a set of indices of M_t, we can model the agent distribution as

$$\pi_t(\cdot \mid q) = \Pi_z(\cdot \mid Z^t_{R(q,M_t)},\ q)$$

where $Z^t_{R(q,M_t)}$ represents the composed version of the retrieved memory modules for some composition function $\oplus$, i.e. $Z^t_{R(q,M_t)} := \bigoplus_{i\in R(q,M_t)} Z_i^t$. Then for an update event that updates only a subset of modules with indices U_t, we can define the overlap event as

$$A_t(q) := U_t \cap R(q, M_t) \subseteq [K]$$

Assume:

1. ***(A1) No behaviour change under zero overlap.*** *If $A_t(q)$ is empty then the behaviour of the model in every query is the same: $A_t(q) = \emptyset \implies \pi_t(\cdot \mid q) = \pi_{t+1}(\cdot \mid q)$*
2. ***(A2) Updated behaviour is bounded.*** *The updates are contained such that for every q, there exists $\epsilon_t > 0$ such that $D(\pi_t(\cdot \mid q) \parallel \pi_{t+1}(\cdot \mid q)) < \epsilon_t$ for some probability divergence D*

Then for any query distribution Q, we have

$$\mathbb{E}_{q\sim Q}[D(\pi_t(\cdot \mid q) \parallel \pi_{t+1}(\cdot \mid q))] < \rho_t \epsilon_t$$

where $\rho_t = \Pr_{q\sim Q}(A_t(q) \neq \emptyset)$

Proof. If $A_t(q) = \emptyset$, by **(A1)** $\pi_t(\cdot \mid q) = \pi_{t+1}(\cdot \mid q)$ resulting in $D(\pi_t(\cdot \mid q) \parallel \pi_{t+1}(\cdot \mid q)) = 0$. If $A_t(q) \neq \emptyset$ then it holds that $D(\pi_t(\cdot \mid q) \parallel \pi_{t+1}(\cdot \mid q)) < \epsilon_t$. Thus we have

$$D(\pi_t(\cdot \mid q) \parallel \pi_{t+1}(\cdot \mid q)) \leq \epsilon_t \mathbf{1}_{[A_t(q)\neq\emptyset]}$$

Taking the expectation over Q, we have

$$\mathbb{E}_{q\sim Q}[D(\pi_t(\cdot \mid q) \parallel \pi_{t+1}(\cdot \mid q))] \ \leq \ \mathbb{E}_{q\sim Q}[\epsilon_t \mathbf{1}_{[A_t(q)\neq\emptyset]}] \ = \ \rho_t \epsilon_t$$

Corollary. For a monolithic memory, $K = 1$ and any non-trivial parametric changes will result in $\Pr(A_t(q) \neq \emptyset) = 1$ and the divergence is bounded above by $\mathbb{E}_{q\sim Q}[D(\pi_t(\cdot \mid q) \parallel \pi_{t+1}(\cdot \mid q))] < \epsilon_t$. By modularizing the memory, we can force ρ_t to be $\ll 1$ and interference can be reduced with better retrieval overlap.