GLOVE: Global Verifier for LLM Memory-Environment Realignment

Xingkun Yin, Hongyang Du
Department of Electrical and Computer Engineering
The University of Hong Kong
Hong Kong SAR, China
{yinxingkun@connect., duyu@}hku.hk

Abstract

Most existing memory-enhanced Large Language Model (LLM) approaches implicitly assume that memory validity can be established either through external evaluators that provide task-specific success signals or through internal model cognition, such as reflection, for editing memory entries. However, these assumptions often break down in practical environments with dynamic drifts. We propose the Global Verifier (GLOVE), a plug-and-play module for LLM memory systems that establishes a relative notion of truth to achieve memory-environment realignment in the face of environmental drifts. Through active probing to detect inconsistencies between retrieved memories and fresh observations, GLOVE enables memory-environment realignment by verifying and updating memory without access to task-specific ground-truth supervision or strong reliance on model introspection. We evaluate GLOVE across a spectrum of tasks ranging from web navigation and discrete planning to continuous and embodied control, covering both simulated benchmarks and real-world robotic deployment. Across all domains, GLOVE improves adaptation across various LLM memory designs under both explicit and implicit drift, often recovering performance from near-failure to high success rates. These results suggest a practical pathway to memory-environment realignment for self-evolving cognitive agents.

Code and Experiment Settings: https://github.com/NICE-HKU/GLOVE.

1 Introduction

Large Language Models (LLMs) have evolved from static conversational interfaces to autonomous agents that execute long-horizon tasks, including web navigation and embodied control [25, 4]. This shift toward general-purpose autonomy increases the importance of experience accumulation, with recent evidence suggesting that future capability gains will rely on agents learning from their own interaction histories [16]. External memory is therefore becoming a central component of LLM agents, as it allows them to reuse past observations, action traces, and task outcomes across long time horizons. However, this paradigm also introduces a fundamental weakness, as stored experience is often treated as reliable once it has been written into memory. Without continuous validation against the current environment, memory can grow into a mixture of useful experience, outdated facts, and internally plausible but invalid assumptions. This problem becomes especially critical in dynamic settings, where an experience may be correct when it is stored but false when it is later retrieved. For example, a web interface may change its button layout, a navigation route may become blocked, or a user preference may shift over time. In these cases, stale memory can persist even after it no longer reflects the current environment, causing the agent to reuse obsolete decisions with high confidence.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Existing LLM memory systems provide useful mechanisms for storing, organizing, and retrieving experience, but they do not fully solve this deployment-time validation problem. We characterize current approaches to maintaining memory validity along two broad axes. The first axis is memory-internal adaptation. MemoryBank uses passive decay [28], while Generative Agents synthesize high-level insights through reflection [14]. Another line reduces the cognitive burden on the LLM by reorganizing memory into structured forms, such as graph-based or hierarchical representations, as in MemGPT and G-Memory [12, 26]. These approaches improve memory usability, retrieval quality, and reasoning coherence. However, they still rely on the agent to judge memory validity mainly from internal consistency. This is insufficient under environmental drift, because a stale memory can be logically consistent with prior experience while being inconsistent with the current environment. Passive decay, reflection, or organization may make such memories easier to forget, retrieve, or justify, but they do not by themselves test whether the stored experience remains valid. The second axis is feedback-based validation, where memory or behavior is checked using explicit rewards, verifiers, task-success feedback, or other external signals [15, 2]. Representative examples include Memory-R1 and Voyager with an evaluator [23, 19]. Such feedback can be effective when it is dense, timely, and aligned with the memory entries that need to be edited. In open-ended deployments, however, supervision is often sparse, delayed, or absent. Even when partial feedback exists, it usually evaluates terminal task outcomes rather than the intermediate state-action experiences that populate memory. This creates a localization gap. For example, in navigation, a previously safe route may become blocked due to environmental changes [3]. A reflection-based agent may repeatedly justify a plan using an obsolete map stored in memory, while a feedback-based approach may only observe failure at the end of a long trajectory. In both cases, the agent lacks a direct mechanism for identifying which intermediate memory entries became invalid. As a result, it can retrieve and reuse stale steps indiscriminately, and may treat hallucinated or outdated premises as reliable evidence [8].

Dynamic environments further expose this gap by changing the response pattern associated with a familiar context. Under such changes, the same retrieved memory can lead to different consequences at different deployment times. The key issue is therefore not merely whether an agent can store useful experience, but whether it can detect when live interaction contradicts retrieved experience and then realign its memory store before the stale experience continues to affect future decisions. A practical memory system should treat memory as an editable hypothesis about current environment behavior, rather than as a static record of past truth. This requires an online realignment mechanism that monitors inconsistencies between stored experience and fresh interaction evidence, suppresses or revises contradicted memories, and updates the memory store as new evidence accumulates. To address this issue, we introduce a relative-truth-based approach for memory-environment realignment. Relative truth refers to the current validity of memory as supported by observable interaction evidence, including immediate state transitions, reward or score changes, user responses, and delayed task feedback. This formulation covers both explicit drift, where misalignment appears through changed state-action-outcome transitions, and implicit drift, where the visible transition structure may remain similar while downstream feedback changes. Thus, memory validity is determined by its consistency with current interaction evidence, rather than by internal coherence alone.

We instantiate this design with Global Verifier (GLOVE), a plug-and-play module that augments existing memory systems without modifying their underlying architecture. GLOVE monitors retrieved memory during online interaction, detects persistent inconsistencies between stored experience and fresh evidence, and realigns the memory store by removing obsolete experience and incorporating verified outcomes. This design enables memory systems to maintain alignment with dynamic environments while remaining compatible with a wide range of existing agent frameworks. Our contributions are summarized as follows:

- **Memory-Environment Misalignment Analysis.** We formalize memory-environment misalignment in dynamic environments, identifying an "epistemic gap" in which LLM agents rely on static historical experience despite unobserved changes in environment dynamics and the absence of immediate ground-truth feedback.

- **Technical Solution.** We propose **GLOVE**, a plug-and-play memory-validation module that closes this epistemic gap through active probing. GLOVE detects cognitive dissonance between retrieved memories and fresh interaction evidence, constructs relative truth from observable consequences of current interaction, and edits the memory store by replacing obsolete experience.

- **Empirical Validation.** We evaluate GLOVE across web navigation, discrete planning, continuous control, and real-world robot navigation under both explicit and implicit environmental drift. Experiments across diverse memory architectures and LLM backbones show that GLOVE improves adaptation after drift, remains effective when exact replay is unavailable, and concentrates interaction overhead around drift events.

2 Related Works

Long-Term Agentic Memory. Recent LLM agents have improved long-term memory through scalable storage, structured retrieval, and reflective summarization, including external memory expansion [20], graph- or hierarchy-based organization [6, 26], and reflection-based consolidation of episodic experience [14, 15, 27]. However, most of these systems still operate in a RAG-style regime that treats retrieved memory as valid once stored [10]. Under environmental drift, this assumption can turn stale experience into a persistent source of error. GLOVE targets this gap by introducing explicit verification between retrieved memory and current environment feedback.

Self-Evolving Agent and Environment Adaptation. A related line of work studies self-evolving agents through introspective self-correction and interaction-driven adaptation. Semantic critics and self-refinement loops can improve reasoning when reliable feedback or reference criteria are available [15, 11, 9]. Other approaches interleave reasoning and acting or improve behavior from ongoing interaction [25, 19], while lifelong adaptation methods address changing environments mainly through failure feedback or stationary-update assumptions [5, 13, 22]. Yet these methods typically update behavior additively and do not explicitly verify whether previously stored memories remain valid after drift. GLOVE focuses on this missing step by detecting memory-environment misalignment and actively realigning stale memory.

3 Problem Setup and Global Verifier

3.1 Memory-Environment Misalignment Objective

We consider a memory-augmented LLM agent interacting with a dynamic environment over discrete time steps. At time t, the agent receives a raw observation o_t and maps it to a task-relevant state representation $s_t = \phi(o_t)$, where $s_t \in \mathcal{S}$ and ϕ is an environment-specific state formation procedure. Examples are provided in Appendix C.3. Given s_t, the agent retrieves a memory context C_t from an external experience bank $\mathcal{D}_t$ and selects an action $a_t \in \mathcal{A}$. After executing a_t, the environment returns an observable outcome s'_t. We characterize the local behavior of the environment at time t by a response distribution $\mathcal{Q}_t(\cdot \mid s, a)$ over observable outcomes for a decision context (s, a). The agent stores past interaction records in an experience bank

$$\mathcal{D}_t = \biguplus_{i=1}^{N_t} \{e_i\}, \qquad e_i = (s_i, a_i, s'_i), \tag{1}$$

where $\mathcal{D}_t$ is an indexed multiset of records collected before time t, duplicate outcomes are preserved as separate samples, while auxiliary metadata is omitted for clarity.

An *environmental drift* is relevant to memory validation when the response distribution that generated a stored experience at e_i's collection time $t_i < t$ differs from the current one:

$$\exists e_i = (s_i, a_i, s'_i) \in \mathcal{D}_t \quad \text{s.t.} \quad \mathcal{Q}_{t_i}(\cdot \mid s_i, a_i) \neq \mathcal{Q}_t(\cdot \mid s_i, a_i), \tag{2}$$

This definition covers both explicit and implicit drifts. In explicit drift, the observable transition outcome changes, such as after an interface update, topology change, or dynamics shift. In implicit drift, the visible transition may remain similar while the downstream consequence changes, e.g., through reward reversal, score change, or shifted user preference. Thus, a memory entry can become obsolete either because the environment now produces a different outcome or because the same apparent outcome has a different task consequence.

We define *memory-environment misalignment* as the resulting discrepancy between the response pattern implied by retrieved memory and the current environment response. For a decision context (s, a), let $\widehat{\mathcal{Q}}_{\mathcal{D}_t}(\cdot \mid s, a)$ denote the empirical outcome distribution induced by relevant entries in the

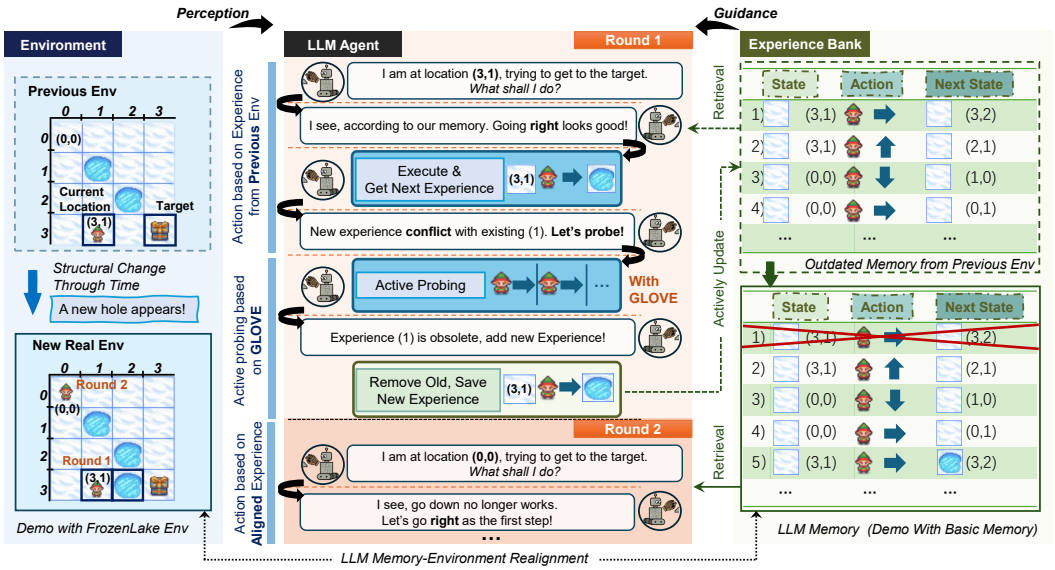


Figure 1: The overview of the GLOVE-augmented LLM workflow.

current experience bank. The objective of *Global Verification* is to realign the memory bank when current interaction evidence indicates that this memory-induced distribution no longer matches the environment:

$$\mathcal{D}_{t+1} = \text{Realign}(\mathcal{D}_t, \mathcal{E}_t), \qquad \widehat{\mathcal{Q}}_{\mathcal{D}_{t+1}}(\cdot \mid s, a) \approx \mathcal{Q}_t(\cdot \mid s, a), \tag{3}$$

where $\mathcal{E}_t$ denotes fresh evidence collected from current or future interactions. The challenge is to decide when such realignment is needed and how much evidence is sufficient to revise memory reliably. GLOVE addresses these two questions through cognitive dissonance detection and relative truth construction, described next.

3.2 GLOVE Framework Overview

Unlike retrieval-augmented agents that treat retrieved experience as fixed ground truth [12], GLOVE treats memory as a **hypothesis** about environment behavior and verifies it through interaction. As illustrated in Fig. 1, GLOVE is integrated into the agent–environment loop. At each step, the agent retrieves relevant experiences from the experience bank $\mathcal{D}$ to guide action selection. After the action is executed, GLOVE compares the observed outcome with stored outcomes associated with the same state-action precondition. When a discrepancy is detected, GLOVE triggers an **active verification** procedure. In *synchronous verification*, the agent re-executes the queried action under the same state context to collect fresh outcomes immediately. When exact replay is unavailable, GLOVE uses *replay-relaxed asynchronous verification*: it suppresses the contradicted memory from future retrieval, marks the queried precondition as pending verification, and accumulates evidence when counterpart states are re-encountered through natural interaction. These two modes differ operationally, but both construct relative truth from observable interaction evidence, without relying on memory-validity labels or internal introspection. Algorithm 1 summarizes the complete procedure, with the three phases detailed below.

3.3 When to Realign: Cognitive Dissonance Detection

Cognitive dissonance occurs when a newly observed outcome is inconsistent with historical experience under the same effective state-action precondition. Since the environment response $\mathcal{Q}_t(\cdot \mid s, a)$ may be stochastic, a single unseen outcome does not necessarily imply environmental drift. GLOVE therefore detects dissonance through distributional consistency. Let the current interaction at time t be $e_t = (s_t, a_t, s'_t)$. GLOVE retrieves counterpart experiences by matching the current precondition (s_t, a_t) against historical records in the experience bank:

$$\mathcal{N}(e_t) = \{e_k \in \mathcal{D} \mid s_k \sim s_t \wedge a_k = a_t\}, \tag{4}$$

Algorithm 1 Memory-enhanced LLM Agent with GLOVE

Input: Task t, LLM agent π, environment Env, experience bank $\mathcal{D}$, probing budget α, persistence threshold p_{th}
Output: Task trajectory τ, updated experience bank $\mathcal{D}$

Initialize pending verification queue $\mathcal{P} \leftarrow \emptyset$ and track consecutive surprise counts for each (s, a)
 # Main Execution Loop
while Task t not finished **do**
  Observe current state s_t and retrieve related information C_t from $\mathcal{D}$
  Form action $a_t \sim \text{LLM}(s_t, C_t)$
  Execute a_t in Env, observe outcome s'_t
  Record transition e_t, (s_t, a_t, s'_t), in trajectory τ
  # GLOVE Process
  # Phase I: Cognitive Dissonance Detection
  Retrieve local counterpart experiences $\mathcal{N}$ according to (4)
  if $\mathcal{N} \neq \emptyset$ **and** $\Phi_{\text{surp}}(e_t)$ holds for p_{th} consecutive encounters of (s_t, a_t) **then**
    # Phase II: Relative Truth Formation
    if sync mode **then**
      Recollect (s_t, a_t) data points for α trials
      Collect fresh outcomes $\mathcal{V} = \{s'_{t,1}, \ldots, s'_{t,\alpha}\}$
      Construct verified transition summary $\hat{Q}_t(\cdot \mid s_t, a_t)$ from $\mathcal{V}$
      # Phase III: Memory–Environment Realignment
      Remove obsolete counterparts: $\mathcal{D} \leftarrow \mathcal{D} \setminus \mathcal{N}$
      Insert verified outcomes: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, s') \mid s' \in (\hat{Q}_t(\cdot \mid s_t, a_t))\}$
    else
      Suppress contradicted counterparts and add e_t to pending verification queue $\mathcal{P}$
    end if
  end if
end while
Return $\tau, \mathcal{D}$

where $s_k \sim s_t$ denotes that two state representations are considered equivalent under a task-dependent matching rule. This relation may be exact in deterministic settings or approximate in open-ended environments, such as nearest-neighbor matching in a representation space. It only defines local comparability for verification and does not constrain the memory format used for action generation. Environment-specific examples are provided in Appendix C.3. The set $\mathcal{N}(e_t)$ can be viewed as historical samples from the response distribution under the same effective precondition (s_t, a_t). If $\mathcal{N}(e_t) = \emptyset$, the transition is treated as novel exploration. Otherwise, GLOVE constructs an empirical distribution $\hat{\mathcal{Q}}_{\text{hist}}(\cdot \mid s_t, a_t)$, where $\hat{\mathcal{Q}}_{\text{hist}}(s')$ is the empirical frequency of outcome s' among experiences in $\mathcal{N}(e_t)$. Cognitive dissonance is detected when the new outcome is unlikely under this distribution. We define the *surprise predicate* as

$$\Phi_{\text{surp}}(e_t) \iff \hat{\mathcal{Q}}_{\text{hist}}(s'_t \mid s_t, a_t) < \epsilon, \tag{5}$$

where ϵ controls the tolerated probability mass of rare outcomes. Intuitively, Φ_{surp} triggers when the observed transition falls outside the typical behavior of the environment under the same precondition. To reduce sensitivity to transient noise, GLOVE maintains a counter for each state-action pair and initiates verification when Φ_{surp} holds for p_{th} consecutive encounters of the same (s_t, a_t). A non-surprising observation for that pair resets the counter.

Detection Guarantee. Assume the environment remains locally stationary over a time window and the agent repeatedly encounters the same (s, a). The historical outcomes can be treated as independent samples from a fixed response distribution $\mathcal{Q}_t(\cdot \mid s, a)$, and $\hat{\mathcal{Q}}_{\text{hist}}(\cdot \mid s, a)$ estimates the typical outcomes under this precondition. We bound the finite-sample deviation of this estimate as:

Theorem 3.1 (Finite-Sample Detection Bound)**.** *For any outcome s' and confidence level $\delta \in (0, 1)$, with probability at least $1 - \delta$,*

$$\left| \hat{\mathcal{Q}}_{\text{hist}}(s' \mid s, a) - \mathcal{Q}_t(s' \mid s, a) \right| \leq \sqrt{\frac{\ln(2/\delta)}{2n}}. \tag{6}$$

Interpretation. Theorem 3.1 characterizes the finite-sample uncertainty of the empirical response estimate under locally stationary dynamics. Let $r_n(\delta) = \sqrt{\ln(2/\delta)/(2n)}$. For any outcome s' whose true stationary probability satisfies $\mathcal{Q}_t(s' \mid s, a) \geq \epsilon + r_n(\delta)$, the empirical estimate can fall below the surprise threshold ϵ only when the estimation error exceeds the concentration margin. Thus, such a typical outcome is falsely marked as surprising with probability at most δ. The persistence threshold p_{th} further reduces transient false alarms by requiring repeated surprise events under the same (s, a) before verification is triggered, trading detection sensitivity for robustness to noise. The proof is provided in Appendix B.1.

Deterministic Special Case. If the environment is deterministic, $\mathcal{Q}_t(\cdot \mid s, a)$ is supported on a single outcome $s^\star$, with $\mathcal{Q}_t(s^\star \mid s, a) = 1$. Under the local stationarity assumption, once at least one historical counterpart sample is observed, $\hat{\mathcal{Q}}_{\text{hist}}(\cdot \mid s, a)$ also concentrates on $s^\star$, so the actual empirical deviation is zero. The distribution-free Hoeffding bound in (6) remains positive for finite n, but the surprise predicate in (5) becomes an exact check of whether the newly observed outcome s'_t equals $s^\star$. Thus, exact-match detection is a deterministic special case of the distributional criterion.

3.4 What to Trust: Relative Truth Construction

When cognitive dissonance is detected, the agent must resolve uncertainty about the current environment response distribution $\mathcal{Q}_t(\cdot \mid s_t, a_t)$. GLOVE addresses this by active verification, which re-estimates the local response distribution under the current environment. Specifically, GLOVE re-executes action a_t from state s_t for α trials, where α is the verification budget for this state-action pair. This produces fresh outcomes

$$\mathcal{V} = \{s'_{t,1}, \ldots, s'_{t,\alpha}\}, \tag{7}$$

which are treated as i.i.d. samples from $\mathcal{Q}_t(\cdot \mid s_t, a_t)$. GLOVE aggregates these outcomes into an empirical distribution $\hat{\mathcal{Q}}_t(\cdot \mid s_t, a_t)$, referred to as the *relative truth*. This distribution is the agent's current estimate of $\mathcal{Q}_t$ from observable interaction evidence, without requiring oracle labels of transition correctness or memory validity. The term relative indicates that correctness is defined with respect to current environment behavior.

Theoretical Guarantee. We next characterize the verification budget needed to construct a reliable relative truth. Consider a discrete outcome setting where, for any state-action pair (s, a) at time t, the response distribution $\mathcal{Q}_t(\cdot \mid s, a)$ is supported on K outcome modes. Re-executing (s_t, a_t) for α trials gives α i.i.d. samples from $\mathcal{Q}_t(\cdot \mid s_t, a_t)$, and GLOVE forms $\hat{\mathcal{Q}}_t$ as their empirical distribution. Using the concentration inequality for discrete distributions [21], we obtain:

Theorem 3.2 (Verification Budget Requirement)**.** *Let $\mathcal{Q}_t$ be the environment response distribution supported on K outcome modes, and let $\hat{\mathcal{Q}}_t$ be the relative truth estimated from α verification samples. For any target accuracy $\varepsilon > 0$ and confidence level $\delta \in (0, 1)$, if*

$$\alpha \;\geq\; \frac{2\big(K \ln 2 + \ln(1/\delta)\big)}{\varepsilon^2}, \tag{8}$$

then with probability at least $1 - \delta$,

$$\|\hat{\mathcal{Q}}_t - \mathcal{Q}_t\|_1 \leq \varepsilon, \tag{9}$$

where $\|\cdot\|_1$ denotes the L_1 distance.

Interpretation. Theorem 3.2 quantifies how α should scale with the outcome space and the desired reliability. Larger outcome spaces K or stricter requirements ε, δ require more verification samples, while stable environments can use smaller budgets. Together with the detection mechanism in Section 3.3, this gives a closed-loop design: detection decides *when* verification is needed, and Eq. 8 specifies *how much* interaction is sufficient to estimate the current response distribution. The proof is provided in Appendix B.2.

Deterministic Special Case. If the queried precondition (s, a) has a deterministic response, then $\mathcal{Q}_t(\cdot \mid s, a)$ is supported on a single outcome $s^\star$, with $\mathcal{Q}_t(s^\star \mid s, a) = 1$. A single re-execution identifies this outcome, so the empirical distribution $\hat{\mathcal{Q}}_t$ constructed with $\alpha = 1$ satisfies $\hat{\mathcal{Q}}_t = \mathcal{Q}_t$ almost surely. Equivalently, $\|\hat{\mathcal{Q}}_t - \mathcal{Q}_t\|_1 = 0$ in (9), so the accuracy requirement holds for any $\varepsilon > 0$ with probability 1. Thus, one probe is sufficient in deterministic settings, and the verification budget can be set to $\alpha = 1$. In practice, α controls the trade-off between verification cost and reliability, so GLOVE starts with a small α and increases it only when inconsistency persists.

Table 1: **GLOVE's robustness to environmental drift (Explicit).** Success rate in percent. The shaded rows show the performance of GLOVE-augmented agents. The LLM is GPT-4o. We annotate the performance gap relative to the baseline (e.g., (↑56.3) indicates improvement).

VERIFIER-VISIBLE STRUCTURAL DRIFT									
	WebShop (Semantic Drift)			**FrozenLake** (Topology Drift)			**MountainCar** (Dynamics Drift)		
Method	**Source**	**→ Drift I**	**→ Drift II**	**Source**	**→ Drift I**	**→ Drift II**	**Source**	**→ Drift I**	**→ Drift II**
No Memory (Plain)	15	5	0	0	5	0	85	75	55
Vanilla	85	0	0	85	0	0	100	100	90
+GLOVE	85	85 (↑85)	95 (↑95)	80	75 (↑75)	80 (↑80)	100	100	100 (↑10)
MemoryBank	85	20	20	80	0	45	95	100	100
+GLOVE	100	90 (↑70)	85 (↑65)	80	70 (↑70)	65 (↑20)	95	100	100
Voyager	85	0	0	85	0	0	100	90	100
+GLOVE	80	90 (↑90)	95 (↑95)	85	85 (↑85)	75 (↑75)	100	100 (↑10)	100
Generative Agent	80	0	0	80	15	0	100	95	70
+GLOVE	75	95 (↑95)	95 (↑95)	80	80 (↑65)	85 (↑85)	95	100 (↑5)	100 (↑30)

3.5 How to Update: Memory–Environment Realignment

After relative truth construction, GLOVE updates the experience bank so that future retrieval no longer depends on obsolete counterparts. Let $\mathcal{N}(e_t)$ be the counterpart collection in Eq. (4), and let $\mathcal{V} = \{s'_{t,1}, \ldots, s'_{t,\alpha}\}$ be the fresh outcomes collected for (s_t, a_t). In synchronous verification, GLOVE removes contradicted counterparts and inserts verified outcomes:

$$\mathcal{D} \leftarrow \big(\mathcal{D} \setminus_{\mathrm{m}} \mathcal{N}(e_t)\big) \uplus \biguplus_{j=1}^{\alpha} \{(s_t, a_t, s'_{t,j})\}. \tag{10}$$

Here $\setminus_{\mathrm{m}}$ and $\uplus$ denote multiset removal and union, preserving duplicate outcomes as separate samples that induce $\hat{\mathcal{Q}}_t(\cdot \mid s_t, a_t)$ without changing the agent's memory interface. When exact replay is unavailable, GLOVE applies replay-relaxed asynchronous realignment. It suppresses contradicted counterparts from retrieval, adds the queried precondition to a pending verification queue $\mathcal{P}$, and accumulates evidence when comparable contexts are later encountered through natural interaction. Once sufficient evidence is collected, GLOVE applies the same replacement rule as in Eq. (10). This fallback avoids repeated execution of irreversible, costly, or safety-critical actions while preventing contradicted memories from continuing to guide future decisions.

4 Experiments

4.1 Experiment Setup

Environments. We evaluate GLOVE on three benchmarks and one real-world robotic deployment. The benchmarks include **WebShop** [24] for web navigation, **FrozenLake** [1] for discrete planning, and **MountainCar** [1] for continuous control. To test deployment under physical noise, we also evaluate GLOVE on a **JetRover** mobile robot, where the agent must navigate a real maze and update stale target-location memory after semantic drift. These environments cover semantic reasoning, discrete planning, continuous dynamics, and embodied memory realignment. We introduce controlled environmental drifts as difficulty-enhanced evaluation settings. Agents first build memories in the source environment and then continue solving the same task after drift, allowing us to measure adaptation under stale memory. Specifically, *Explicit Drift* refers to observable structural changes, including altered layouts, transition dynamics, or semantic mappings. We report success rate over twenty rounds. *Implicit Drift* refers to changes in hidden task logic, such as reward reversals or hidden transition rules, that are not directly observable from immediate state descriptions alone. We report the score obtained over twenty rounds. All drift settings are implemented as systematic benchmark-level modifications and applied uniformly across methods without algorithm-specific tuning. Detailed specifications are provided in Appendix C.4 and Appendix C.5.

Baselines. We evaluate GLOVE across representative memory architectures to test whether it functions as a general augmentation module. The baselines include (1) **No Memory**, a zero-shot agent without external context; (2) **Vanilla**, a basic RAG agent; and (3) agentic memory architectures, including **Voyager** [19], **MemoryBank** [28], and **Generative Agents** [14]. Each architecture is evaluated in its original form and with GLOVE, isolating the effect of memory verification and realignment. These baselines cover plain retrieval, passive decay, reflective memory, and iterative-refinement memory backends. Details are provided in Appendix C.1. We focus on memory-augmented LLM agents as the main comparison class, while Appendix D.4 provides an additional positioning check against adaptive model-based RL methods.

LLM Backbones. We evaluate GLOVE across diverse LLM backbones to test architecture-agnostic performance, including open-weights models including **Llama-3.1-8B**, **Llama-3.3-70B**, **Qwen2.5-7B**, **Qwen3-30B**, **DeepSeek-V3.2**, as well as proprietary models, such as **GPT-4o** and **Grok-3**.

4.2 Memory Realignment under Environmental Drift

Verifier-visible drift. We evaluate agents under explicit structural drift by transferring memories collected in a source environment to drifted variants with altered semantics, topology, or dynamics. Table 1 reports success rates before and after drift using GPT-4o as the backbone. Static memory retrieval leads to severe degradation once the environment changes: under WebShop semantic drift (Drift I), Voyager drops from 85% to 0%, and under FrozenLake topology drift (Drift II), Generative Agent drops from 80% to 0%. These failures show that memory systems performing well in the source environment can still lack mechanisms to realign stored experience after structural change. MemoryBank provides limited robustness through time-based forgetting, but this passive adaptation is slow and can retain high-confidence invalid memories; for example, it still reaches 0% success in FrozenLake Drift I. Adding GLOVE yields consistent improvements across explicit drift settings by detecting conflicts between stored memories and observed outcomes and then realigning the experience bank. In WebShop, GLOVE-augmented agents recover to around 90% success after semantic drift, compared with 20% for MemoryBank. In FrozenLake Drift II, GLOVE improves performance by an average of 65% over non-augmented agents, indicating effective correction of outdated topological memories. In MountainCar, GLOVE maintains near-perfect performance, showing adaptation to changed continuous dynamics. These gains hold across multiple LLM backbones, as summarized in Table 4 in Appendix D.2. Appendix D.3.1 further shows that broader episodic context alone does not remove stale-memory failure under drift, supporting that the observed gains come from explicit memory verification and realignment rather than a specific backbone, memory format, or context length.

Verifier-hidden drift. We evaluate GLOVE under implicit drift, including reward reversals and changes in hidden transition logic. Unlike explicit drift, implicit drift may leave the immediate observation space unchanged, causing memory-based agents to trust stale experience that remains superficially consistent with the current state. As shown in Table 2, this can make stored experience harmful after drift: MemoryBank, despite its forgetting mechanism, suffers an 18.8% score decrease under the hidden WebShop semantic change. GLOVE addresses this failure by checking observable interaction consequences, including both outcome s' and associated reward dynamics. In Table 2, adding GLOVE improves the Vanilla agent from 47.5% to 97.5% in FrozenLake hidden drift. In the Grok-3 benchmark, Voyager with GLOVE reaches 100% success, compared with 62.5% for Voyager alone, as reported in Table 14. These results show that GLOVE can realign memory even when the drift is not visible from state descriptions alone. More generally,

Table 2: GPT-4o: GLOVE's robustness to implicit environmental drift.

VERIFIER-HIDDEN DRIFT				
	WebShop (Semantic Drift)		**FrozenLake** (Reward Reversal)	
Method	**Source**	**→ Hidden drift**	**Source**	**→ Hidden drift**
No Memory	82.5	87.5	0	0
Vanilla	100	75	47.5	50
+GLOVE	97.5	93.8 (↑18.8)	35	97.5 (↑47.5)
MemoryBank	100	81.2	62.5	57.5
+GLOVE	98.8	98.8 (↑17.5)	67.5	97.5 (↑40)
Voyager	98.8	75	67.5	50
+GLOVE	98.8	98.8 (↑23.8)	40	97.5 (↑47.5)
Generative Agent	98.8	75	62.5	50
+GLOVE	98.8	75	62.5	92.5 (↑42.5)

Table 3: **Direct adaptive-memory comparison on Qwen3-30B FrozenLake explicit drift.** Natural columns use 20 episodes per drift phase; all methods achieve 20/20 in Source. The last column reports completed successes under the same $B = 310$ post-drift interaction budget.

	Natural adaptation		Fixed budget
Method	**Drift I**	**Drift II**	**Successes @ $B = 310$**
GLOVE	**16/20**	**15/20**	**31**
LatestOverwrite	16/20	0/20	16
RecentK-K3	16/20	0/20	16
SlidingWindow-W5	14/20	0/20	14
RecencyWeighted-H5	12/20	0/20	12
PageHinkleyReset	14/20	0/20	14
PeriodicRefresh-S30	5/20	7/20	<u>18</u>

explicit drift can be detected from changes in state-action-outcome transitions, while implicit drift requires observable downstream consequences such as reward, score, user response, or delayed task feedback. GLOVE uses these signals as interaction evidence rather than oracle labels of memory validity. If both the interaction process and all feedback remain unchanged, the drift is observationally indistinguishable from stationarity for any experience-based method.

4.3 Statistical Reliability

We further assess the statistical reliability of the main recovery trend through 10 paired runs on the Qwen3-30B FrozenLake explicit-drift setting. For each run and memory backend, the baseline and its GLOVE variant use byte-equivalent source-memory snapshots and identical task instances. In Drift II, GLOVE improves Vanilla from $0.0 \pm 0.0\%$ to $75.0 \pm 0.0\%$ and MemoryBank from $18.5 \pm 7.5\%$ to $75.0 \pm 0.0\%$. Both improvements are positive in all 10 paired runs and remain significant after Bonferroni correction (adjusted $p = 0.0078$ for both; Cohen's $h = 2.094$ and 1.205). For settings without repeated paired runs, we do not make statistical-significance claims. Full counts, confidence intervals, and paired tests are reported in Appendix D.2.1.

4.4 Comparison with Direct Online Memory Adaptation

To separate GLOVE's gains from simple online updating and from the extra-interaction confound, we compare against six direct adaptive rules on Qwen3-30B FrozenLake explicit drift: LATESTOVERWRITE, RECENTK-K3, SLIDINGWINDOW-W5, RECENCYWEIGHTED-H5, PAGEHINKLEYRESET, and PERIODICREFRESH-S30. In the natural protocol, each baseline updates only from its own trajectory, while GLOVE retains standard verification. We also impose a strict $B = 310$ post-drift interaction horizon: GLOVE uses 280 task interactions plus 30 verification interactions, whereas each baseline may spend all 310 on task execution.

The pair-local rules expose a *self-confirming exploration deadlock*. After Drift I, the stale `Down→Hole` record discourages the policy from selecting Down even when it becomes safe again in Drift II; LatestOverwrite selects Down zero times across 381 Drift-II actions and therefore never observes the corrective transition. PeriodicRefresh can restore exploration through blind forgetting, but also removes valid memories. GLOVE instead targets suspected stale records for verification, allowing correction even when the current policy avoids the corresponding action. Under the same $B = 310$ horizon, GLOVE achieves 31 successes versus 18 for the strongest adaptive baseline. Full protocols are in Appendix D.2.1.

4.5 Verification Assumptions and Cost

Replay-relaxed verification. We evaluate whether GLOVE remains effective when exact replay is unavailable. In the replay-feasible setting, the agent can re-execute a matched state-action pair to estimate the current environment response. This assumption can fail in continuous, irreversible, or high-stakes environments where the agent cannot safely reset and repeat an action. We therefore evaluate a replay-relaxed asynchronous variant, which treats exact replay as the strongest verification mode and asynchronous verification as a safer fallback.

Instead of re-executing a consequential action, GLOVE suppresses the contradicted memory and waits for non-destructive confirmation from future natural encounters, passive observations, sandboxed checks, or task feedback. As shown in Fig. 2, asynchronous verification recovers at Episode 32 and Episode 54 under the two drift blocks, compared with Episode 26 and Episode 45 for synchronous replay. Thus, removing exact replay delays recovery but does not prevent memory realignment, showing that GLOVE remains effective under weaker deployment assumptions.

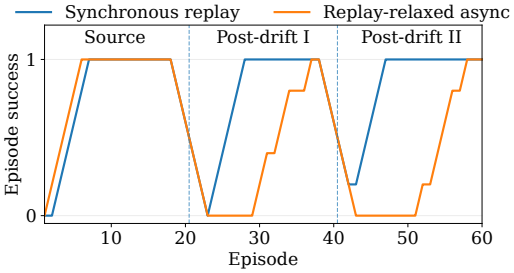


Figure 2: Replay-feasible versus replay-relaxed verification in FrozenLake.

Budget sensitivity. Active verification adds environment interactions, but the overhead is selective. Appendix D.5 shows that verification steps concentrate around drift events rather than stable periods. The same analysis shows that the probing budget α controls a cost–reliability trade-off, while moderate and strict counterpart-matching granularities achieve the same average reward with similarly low verification overhead.

4.6 Real-Robot Evaluation on JetRover

We evaluate GLOVE in a real-robot setting on a HiWonder JetRover [7] robot. Experimental details are provided in Appendix C.6. The setup includes physical nuisance variation, such as irregular cardboard walls, small target displacement, and added path obstacles. The robot searches for the blue block. After source-memory formation, we induce semantic drift by swapping the blue and green blocks, testing whether GLOVE distinguishes persistent target-location drift from incidental physical noise in deployment. As shown in Fig. 3, No Memory fails to find the target reliably, while Static Memory succeeds before drift but collapses after the swap because it keeps revisiting the stale target location. In contrast, GLOVE detects the semantic mismatch, updates the stale memory, and recovers correct search behavior while remaining stable under incidental physical variation.

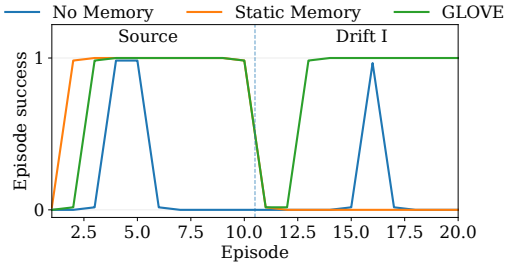


Figure 3: Real-robot evaluation on JetRover.

5 Conclusion

We introduced GLOVE, a plug-and-play memory-validation module for memory-augmented LLM agents in dynamic environments. GLOVE detects cognitive dissonance between stored experience and current interaction evidence, constructs relative truth from fresh observations, and realigns the memory bank by replacing obsolete entries. Across web navigation, discrete planning, continuous control, and real-robot navigation, GLOVE improves adaptation under both explicit and implicit drift while remaining compatible with diverse memory architectures and LLM backbones. These results identify memory-environment misalignment as one of the core reliability issues for long-horizon agents and show that deployment-time memory realignment can improve robustness.

Acknowledgments and Disclosure of Funding

We thank Yaoyutong Shen for helpful hardware robotic experimental setup during her internship at the University of Hong Kong.

Funding. This work received no specific external funding.

Competing interests. The authors declare no competing interests related to this work.

References

[1] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[2] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[3] Gregory Ditzler, Manuel Roveri, Cesare Alippi, and Robi Polikar. Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, 10(4):12–25, 2015.

[4] Danny Driess, Fei Xia, Mehdi S.M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, et al. Palm-e: An embodied multimodal language model. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

[5] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4):1–37, 2014.

[6] Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. LightRAG: Simple and fast retrieval-augmented generation. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Findings of the Association for Computational Linguistics: Empirical Methods in Natural Language Processing 2025*, pages 10746–10761, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7.

[7] Hiwonder. JetRover (Jetson Nano Version) Documentation. https://wiki.hiwonder.com/projects/JetRover/en/jetson-orin-nano/, 2025. Accessed: 2026-04-20.

[8] Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations*, 2024.

[9] Tian Lan, Wenwei Zhang, Chen Xu, Heyan Huang, Dahua Lin, Kai Chen, and Xian-Ling Mao. Criticeval: Evaluating large-scale language model as critic. *Advances in Neural Information Processing Systems*, 37:66907–66960, 2024.

[10] 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.

[11] 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*, volume 36, pages 46534–46594. Curran Associates, Inc., 2023.

[12] Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2024.

[13] Sindhu Padakandla. A survey of reinforcement learning algorithms for dynamically varying environments. *ACM Computing Surveys (CSUR)*, 54(6):1–25, 2021.

[14] Joon Sung Park, Joseph O'Brien, Carrie Jun 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*, pages 1–22, 2023.

[15] 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.

[16] David Silver and Richard S. Sutton. Welcome to the era of experience. Technical report, Google DeepMind, 2025.

[17] Richard S Sutton. Integrated modeling and control based on reinforcement learning and dynamic programming. *Advances in neural information processing systems*, 3, 1990.

[18] Richard S Sutton. Reinforcement learning architectures for animats. In *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior*, pages 288–296. MIT Press Cambridge, MA, 1991.

[19] 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. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856.

[20] Yu Wang, Dmitry Krotov, Yuanzhe Hu, Yifan Gao, Wangchunshu Zhou, Julian McAuley, Dan Gutfreund, Rogerio Feris, and Zexue He. M+: Extending memoryllm with scalable long-term memory. In *Forty-second International Conference on Machine Learning*, 2025.

[21] Tsachy Weissman, Erik Ordentlich, Gadiel Seroussi, Sergio Verdu, and Marcelo J Weinberger. Inequalities for the l1 deviation of the empirical distribution. *Hewlett-Packard Labs, Tech. Rep*, page 125, 2003.

[22] Yule Wen, Yixin Ye, Yanzhe Zhang, Diyi Yang, and Hao Zhu. Real-time reasoning agents in evolving environments. *arXiv preprint arXiv:2511.04898*, 2025.

[23] Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Kristian Kersting, Jeff Z Pan, Hinrich Schütze, et al. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *arXiv preprint arXiv:2508.19828*, 2025.

[24] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022.

[25] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022.

[26] Guibin Zhang, Muxin Fu, Kun Wang, Guancheng Wan, Miao Yu, and Shuicheng Yan. G-memory: Tracing hierarchical memory for multi-agent systems. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

[27] Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, et al. Agent learning via early experience. *arXiv preprint arXiv:2510.08558*, 2025.

[28] 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, pages 19724–19731, 2024.

Appendix Contents

A. Limitation and Broader Impact ..... Page 13

A.1 Limitation ..... Page 13
A.2 Broader Impact ..... Page 13

B. Theoretical Analysis ..... Page 14

B.1 Proof of Theorem 3.1: Finite-Sample Detection Bound ..... Page 14
B.2 Proof of Theorem 3.2: Verification Budget Requirement ..... Page 14

C. Experimental Details ..... Page 15

C.1 Baseline Setup ..... Page 15
C.2 Environment Setup ..... Page 15
C.3 State Formation and Matching Rules ..... Page 16
C.4 Explicit Environmental Drifts ..... Page 17
C.5 Implicit Environmental Drifts ..... Page 18
C.6 JetRover Real-Robot Setup ..... Page 18
C.7 Case Study: FrozenLake Implicit Environmental Drifts ..... Page 21

D. Additional Experiment Results ..... Page 24

D.1 Experiment Results: Backbone Sweep Overview ..... Page 24
D.2 Experiment Results: Explicit Environmental Drifts ..... Page 24
D.3 Experiment Results: Implicit Environmental Drifts ..... Page 28
D.4 Positioning against Adaptive Model-Based RL ..... Page 31
D.5 Cost Analysis ..... Page 32
D.6 JetRover Real-World Robotic Result ..... Page 34
D.7 Result Analysis ..... Page 34

A Limitation and Broader Impact

A.1 Limitation

GLOVE requires observable interaction evidence to detect memory-environment misalignment. Explicit drift can be detected from changed state-action-outcome transitions, while implicit drift requires downstream feedback such as reward, score, user response, or delayed task feedback. If neither the interaction process nor any feedback changes, the drift is observationally indistinguishable from stationarity for any experience-based method. GLOVE also depends on the cost and safety of evidence collection. In replay-feasible settings, active probing can directly estimate the current local response pattern. For costly, irreversible, or safety-critical actions, GLOVE uses replay-relaxed asynchronous realignment by suppressing contradicted memories and waiting for safe evidence from future encounters, proxy checks, or delayed feedback. This avoids unsafe repetition, but recovery can be slower and replacement memories may remain incomplete when safe feedback is sparse. Finally, the JetRover experiment provides a physical proof of concept under irregular walls, target displacement, and path-level obstacles, but it still uses a pre-constructed navigation map and waypoints. Extending GLOVE to fully open-world embodied agents, including 3D navigation platforms such as drones or autonomous vehicles, remains future work.

A.2 Broader Impact

GLOVE improves the reliability of memory-augmented LLM agents in dynamic environments by verifying retrieved memories against fresh interaction evidence. This may benefit long-horizon applications such as web automation, assistive agents, and embodied robotics, where stale experience can otherwise cause persistent failures. However, active probing also introduces risks where probes may trigger unwanted actions or extra interaction costs, especially in safety-critical or user-facing settings. Moreover, GLOVE constructs relative truth from observable consequences, which should not be treated as a substitute for safety, fairness, or human preference alignment. We therefore view GLOVE as a reliability layer rather than a standalone safety mechanism. Deployments should use task-level safety constraints, human approval for irreversible actions, and fixed policies that override

learned experience. GLOVE should not be used for malicious automation, surveillance, deception, or other harmful applications.

B Theoretical Analysis

This section provides a formal analysis of the two statistical components underlying GLOVE. The analysis mirrors the two-stage structure of the framework. First, we study the reliability of cognitive dissonance detection under finite historical samples, establishing a false-alarm bound based on Hoeffding's inequality. Second, we analyze the active verification process and derive a sample-complexity requirement for constructing a reliable relative truth, using concentration results for discrete distributions due to Weissman et al. [21].

B.1 Proof of Theorem 3.1: Finite-Sample Detection Bound

We prove Theorem 3.1. Consider a fixed state–action pair (s, a) and assume that, over a local time window, the environment response distribution $\mathcal{Q}_t(\cdot \mid s, a)$ remains unchanged. Let s' be any specific outcome.

Each historical execution of (s, a) produces an outcome S'_i drawn from $\mathcal{Q}_t(\cdot \mid s, a)$. Define the indicator random variables

$$X_i = \mathbf{1}\{S'_i = s'\}, \qquad i = 1, \ldots, n.$$

Under the locally stationary assumption, $\{X_i\}_{i=1}^n$ are independent and identically distributed Bernoulli random variables with

$$\mathbb{E}[X_i] = \mathcal{Q}_t(s' \mid s, a).$$

The empirical probability of observing outcome s' is

$$\hat{\mathcal{Q}}_{\text{hist}}(s' \mid s, a) = \frac{1}{n} \sum_{i=1}^{n} X_i.$$

By Hoeffding's inequality, for any $\lambda > 0$,

$$\Pr\Big(\Big| \hat{\mathcal{Q}}_{\text{hist}}(s' \mid s, a) - \mathcal{Q}_t(s' \mid s, a) \Big| \geq \lambda \Big) \;\leq\; 2\exp(-2n\lambda^2).$$

Setting the right-hand side equal to δ and solving for λ yields

$$\lambda = \sqrt{\frac{\ln(2/\delta)}{2n}}.$$

Therefore, with probability at least $1 - \delta$,

$$\Big| \hat{\mathcal{Q}}_{\text{hist}}(s' \mid s, a) - \mathcal{Q}_t(s' \mid s, a) \Big| \leq \sqrt{\frac{\ln(2/\delta)}{2n}},$$

which proves Eq. (6) and completes the proof.

□

B.2 Proof of Theorem 3.2: Verification Budget Requirement

We prove Theorem 3.2. Fix a state–action pair (s, a) at time t and assume that the environment response distribution $\mathcal{Q}_t(\cdot \mid s, a)$ is supported on a finite set of K distinct outcome modes. Let

$$\mathcal{Q}_t = (q_1, \ldots, q_K)$$

denote this categorical distribution.

Active verification re-executes (s, a) for α trials, producing outcomes

$$S'_1, \ldots, S'_\alpha \sim \mathcal{Q}_t(\cdot \mid s, a),$$

which are independent and identically distributed by construction. The relative truth $\hat{\mathcal{Q}}_t$ is defined as the empirical distribution of these samples.

The problem therefore reduces to estimating a discrete distribution from α i.i.d. samples. By Theorem 2 of Weissman et al. [21], for any $\varepsilon > 0$,

$$\Pr\Big(\|\hat{\mathcal{Q}}_t - \mathcal{Q}_t\|_1 > \varepsilon\Big) \;\le\; 2^K \exp\!\left(-\frac{\alpha\varepsilon^2}{2}\right).$$

Setting the right-hand side to δ and solving for α gives

$$\alpha \;\ge\; \frac{2\big(K\ln 2 + \ln(1/\delta)\big)}{\varepsilon^2}.$$

Under this condition, with probability at least $1-\delta$,

$$\|\hat{\mathcal{Q}}_t - \mathcal{Q}_t\|_1 \le \varepsilon,$$

which proves Eqs. (8) and (9), completing the proof.

□

C Experimental Details

C.1 Baseline Setup

In this section, we provide detailed descriptions of each baseline for comparison in our experiments:

Vanilla Agent. The Vanilla agent implements a fundamental RAG architecture. Given the current state s, the system queries the external memory to retrieve the top-k historical experiences with the same state s, utilizing them directly as in-context exemplars for action generation

Voyager. Derived from the Voyager agent [19], the Voyager memory employs an iterative prompting mechanism, continuously acquiring and refining actions. In our experiment, we utilize its feedback-based refinement capability to evaluate experience entries.

MemoryBank. MemoryBank [28] implements a memory updating mechanism inspired by the Ebbinghaus forgetting curve, which passively decays the retrieval weights of older experiences to simulate human forgetting. We include this baseline to represent passive adaptation strategies, providing a rigorous contrast to the active verification paradigm introduced in GLOVE.

Generative Agents. Generative Agents [14] synthesize high-level insights from experience entries through reflection during the retrieval phase. We evaluate whether introspection alone is sufficient to detect logical inconsistencies in dynamic environments without active probing.

C.2 Environment Setup

In this section, we provide the environments used in our experiments:

WebShop. WebShop [24] is a scalable simulated e-commerce environment that requires agents to navigate webpages and process semantic instructions to locate desired products. We adapt this environment to evaluate **semantic reasoning** under structural drifts and implicit drifts. We employ this environment to challenge the agent's ability to ground natural language instructions in an evolving website setting. Details of the drift settings are in Section C.4 and Section C.5.

FrozenLake. FrozenLake [1] is a classic grid-map environment, which we employ as a benchmark for **discrete planning** and spatial navigation. Described in detail in Section C.4 and Section C.5, we introduce topological shifts where the map layout and obstacle placement change explicitly, as well as implicit reward reversals where previously safe goals become traps, testing agents' ability to adapt to changing topological environments.

MountainCar. MountainCar [1] represents a standard continuous control problem where a vehicle in a valley must build momentum to reach a target hilltop. We introduce dynamic shifts by altering the engine force magnitude, thereby rigorously testing the agent's ability to adapt to dynamic physical feedback. Details of the drift settings are provided in Section C.4.

C.3 State Formation and Matching Rules

For all environments, the agent does not observe the underlying environment state directly. Instead, it operates on an observation-derived state representation that is passed to the LLM, stored in memory, and reused for counterpart matching during verification. Importantly, this matching interface should not be confused with the overall memory format used for action generation: in trajectory-based memory backends, matched states are linked into longer action paths and may be further summarized into higher-level natural-language memory, while GLOVE applies local matching only at the verification stage.

WebShop. In WebShop, the state is constructed from the current webpage content and navigation context. Specifically, we use denoised HTML text that removes scripts and irrelevant markup, together with the current URL. This representation captures the visible product information and page structure available to the agent at decision time.

For counterpart retrieval, we use a compact confirmation-state representation derived from decision-relevant observable attributes rather than requiring exact equality over the full raw page. In line with the plug-and-play nature of GLOVE, this matching can reuse the same representation and retrieval mechanism already used by the underlying memory system. We additionally study a controllable granularity parameter η: $\eta = 0.0$ keeps no attribute content, $\eta = 0.5$ keeps only partial attribute information such as `{"color":"red"}`, and $\eta = 1.0$ keeps the full attribute string such as `{"color":"red","size":"x-large"}`. Larger η therefore yields stricter counterpart matching, while smaller η merges more contexts and can trigger unnecessary verification. Appendix D.5 reports the corresponding sensitivity study.

```
"state": {"html": "Back to Search Page 1 (Total results: 50) Next
> ...", "url":
"http://127.0.0.1:3000/search_results/<session_id>/<query>/1"}
```

FrozenLake. In FrozenLake, the state is obtained from the environment observation and includes the agent's current grid position and the corresponding tile type. This compact representation reflects the fully observable but discrete environment used for controlled analysis. Counterpart matching uses exact equality on the observation-derived state stored in memory.

```
"state": {"cur_pos": [1,1], "tile_type": "F", "gold_collected":
1}
```

Under implicit scenarios, destinations may contain fractional reward:

```
"state": {"cur_pos": [1,1], "tile_type": "G", "gold_collected":
0.5}
```

MountainCar. In MountainCar, the state consists of the continuous position and velocity returned by the environment. This setting evaluates GLOVE under continuous control dynamics where small environment changes can invalidate prior experience. Counterpart retrieval follows the same observation-derived state representation stored in memory.

```
"state": {"position": -0.477, "velocity": 0.0068}
```

JetRover Maze. In JetRover, the agent operates on a semantic inspection state rather than raw-image equality. Each memory entry stores the queried target object together with its last verified location. During verification, a mismatch is triggered when the robot revisits the memorized target location and Qwen2.5-VL identifies a different object there. Counterpart matching is therefore performed at the level of target-conditioned inspection context, which tolerates small viewpoint changes, centimeter-scale target displacement, and path-level obstacle variation while still allowing stale target-location memory to be falsified after drift.

Full-Trajectory Ablation To test whether stale-memory failure is induced by local state-matched retrieval, we additionally evaluate a trajectory-level context baseline in WebShop explicit drift. We first collect successful trajectories in the source environment. During drifted evaluation, FULLTRAJ-STATIC selects a successful source trajectory at the beginning of each episode using only pre-drift information and keeps it fixed for the entire episode. At every LLM decision step, the prompt includes the same complete step-by-step source trajectory, together with the current observation and the current episode history. It does not include current-state matched one-step memory snippets for action generation. This protocol tests whether broader episodic context alone can make the agent discount stale source-environment experience. In the GLOVE condition, local counterpart matching is still used only for verification, as in Eq. (4), because GLOVE aims to identify which specific state-action transition has become stale rather than treating an entire trajectory as an indivisible verification unit.

C.4 Explicit Environmental Drifts

We construct explicit environmental drifts by changing the environment structures that are observable through interaction. These drifts modify the mapping from actions to outcomes, including navigation structure in WebShop, obstacle topology in FrozenLake, and physical dynamics in MountainCar. The purpose is to test whether an agent can detect that a previously valid memory no longer matches the current environment.

WebShop. As shown in Fig. 4, the WebShop task proceeds normally until the agent selects the "Buy Now" action. After this action, the environment redirects the agent to an unexpected advertisement page. The page contains multiple candidate buttons and decoy elements, and only one button leads back to successful task completion. The explicit drift is implemented by changing which button is correct across drift phases. This setting tests whether the agent can revise stale navigational memory after the page structure changes.

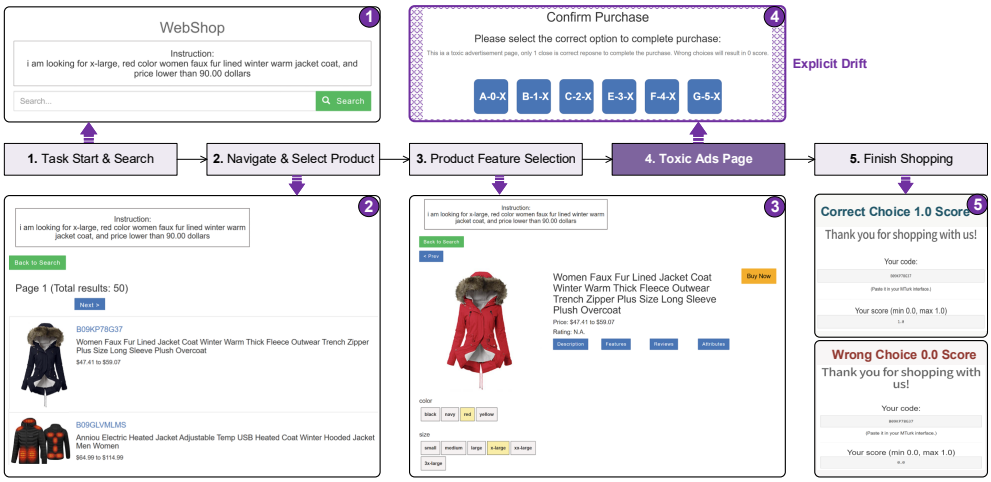


Figure 4: WebShop explicit drift.

FrozenLake. As shown in Fig. 5, the explicit drift in FrozenLake is introduced by modifying the obstacle layout. The source environment and the two drifted environments share the same task objective, but the feasible paths differ because the positions of holes change. This setting tests whether the agent can invalidate outdated topological memory and recover a valid path in the new map.

MountainCar. As shown in Fig. 6, the explicit drift in MountainCar is implemented by decreasing the engine force parameter from the source environment to the drift phases. This change alters the transition dynamics while preserving the observation and action spaces. The agent must therefore adapt its control behavior rather than reuse a policy learned under the original dynamics.



(a) Source (b) Drift I (c) Drift II

Figure 5: FrozenLake explicit drift.

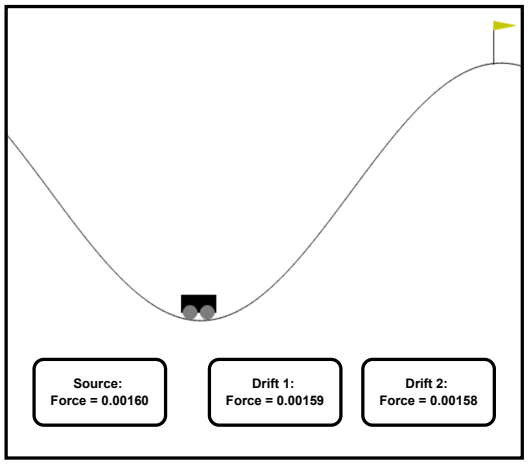


Figure 6: MountainCar explicit drift.

C.5 Implicit Environmental Drifts

We construct implicit environmental drifts by changing hidden task logic while keeping the surface observation largely unchanged. Unlike explicit drift, these changes are not directly visible from the state description or interface. The agent can identify them only through interaction feedback, such as reward or task success. These settings test whether an agent can stop trusting stale memory when the environment appears unchanged but the optimal behavior has shifted.

WebShop. As shown in Fig. 7, the implicit WebShop drift changes the semantic grounding of the attribute "warm color." In the source environment, the correct grounding is Yellow, while in the drifted environment it becomes Red. The instruction and interface remain the same, but the reward feedback changes according to the new semantic rule. This setting tests whether the agent can update semantic memory based on feedback rather than relying on the previous attribute mapping.

FrozenLake. As shown in Fig. 8, the implicit FrozenLake drift keeps the map structure fixed but changes the reward values associated with goal tiles. The goal that was optimal in the source environment becomes a suboptimal trap after drift, while another goal becomes the new global optimum. This setting tests whether the agent can suppress a retrieved policy that still appears feasible but no longer maximizes reward.

C.6 JetRover Real-Robot Setup

We provide additional details for the real-robot evaluation in Section 4.6. The goal of this experiment is not to train a new robot policy, but to test whether GLOVE can detect and correct stale memory in



Figure 7: WebShop implicit drift.

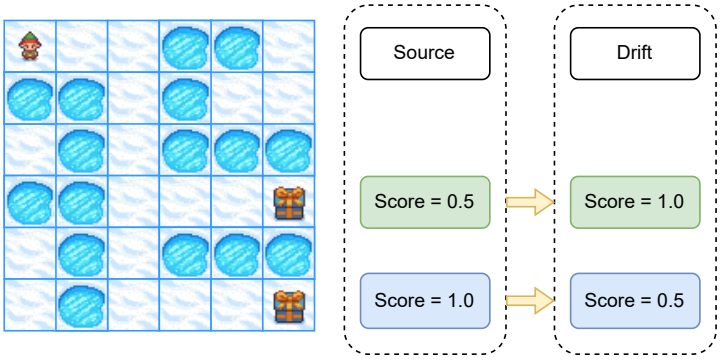


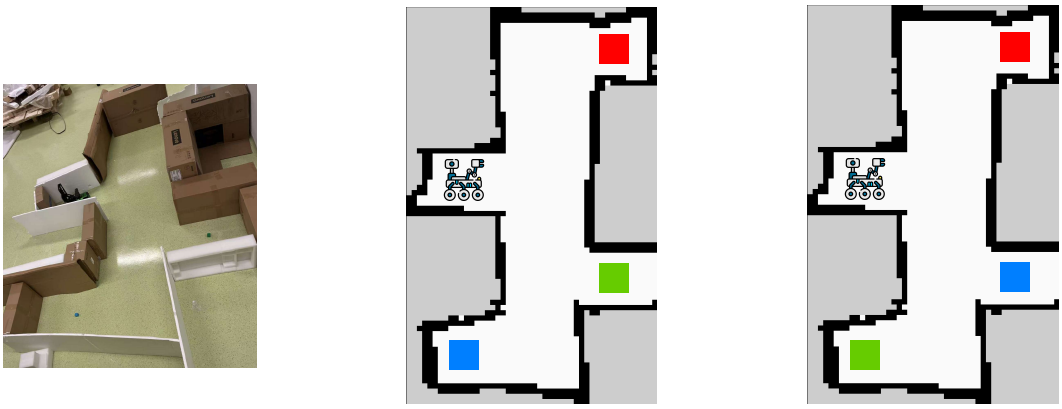
Figure 8: FrozenLake implicit drift.

a replay-relaxed physical environment. We therefore keep the task deliberately simple and use the same deployed robot stack throughout.

Platform and models. We use a JetRover mobile robot in a small indoor maze built from cardboard boxes. The language reasoning module is Qwen2.5-Max, and visual grounding is provided by Qwen2.5-VL. No task-specific training or finetuning is performed. The robot is given a simple instruction, *find the blue block*, and must navigate to the target region and verify the object identity from onboard visual observations.

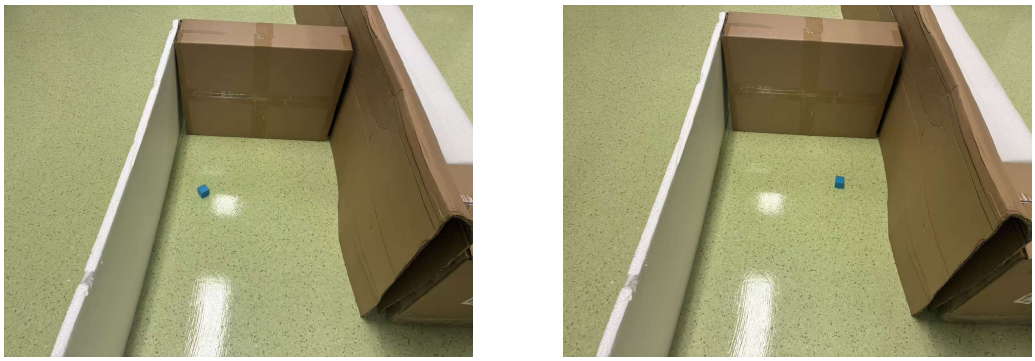
Task and drift protocol. The environment contains three semantic targets: `red`, `green`, and `blue`. In the source environment, the robot first builds memory for the location of the `blue` block. We then introduce drift by swapping the positions of the `blue` and `green` blocks, while keeping the rest of the layout unchanged. Figure 9 shows the real-world layout together with the abstract source map and the drifted map used in evaluation. This setup creates a minimal but concrete stale-memory failure mode: after drift, a policy that trusts static memory will continue navigating to the old `blue` location and encounter the `green` block instead.

Compared methods. We compare three variants built on the same platform and navigation stack. *No Memory* searches without retrieved target-location memory. *Static Memory* always revisits the previously memorized location of the `blue` block. *GLOVE* uses the same memory but verifies the retrieved target against current visual evidence and asynchronously updates stale memory after a



(a) Real-world setup (b) Source environment (c) Drift 1: Cube swap

Figure 9: **JetRover real-world layout and map drift.** Left: the physical cardboard-maze setup used in the real-robot experiment. Middle: the source environment with three colored targets. Right: Drift 1, where the `blue` and `green` blocks exchange positions while the rest of the layout remains unchanged.



(a) Blue block shifted left (b) Blue block shifted right

Figure 10: **Robustness to small target displacement.** Small position changes of the `blue` block within the same local region do not change the semantic target and do not prevent recognition.

mismatch. This comparison isolates the effect of memory verification and realignment, rather than changes in low-level navigation or perception.

Physical noise and robustness. The maze is intentionally imperfect: the cardboard walls are irregular, local paths may be partially blocked, and the target block may shift by a few centimeters between runs. These changes do not alter the semantic target identity, but they introduce realistic deployment-time noise. Figure 10 shows that small left-right target displacement does not prevent the robot from recognizing the `blue` block. Figure 11 shows that the robot can still pass through the corridor and reach the target region even when an additional obstacle is inserted.

Interpretation. This setup produces a clean separation between true semantic drift and incidental physical noise. The swap between the `blue` and `green` blocks changes the correct target location and therefore invalidates previously stored memory. By contrast, centimeter-scale target displacement or temporary path obstruction changes only the local geometry of the run, not the semantic identity of the target. In our experiments, *Static Memory* fails under the former case because it continues to trust stale target-location memory, whereas *GLOVE* uses asynchronous verification to revise memory and recover the correct target.



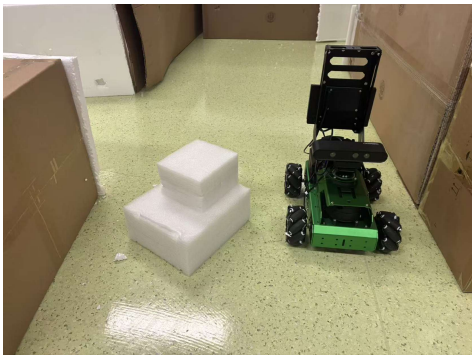
(a) Corridor without extra obstacle



(b) Successful passing without obstacle



(c) Corridor with an added obstacle

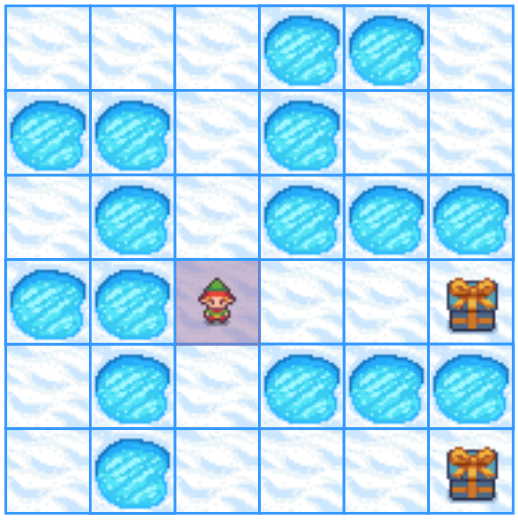


(d) Successful passing with obstacle

Figure 11: **Robustness to path-level noise.** We add an extra obstacle in the corridor to perturb the path geometry. The robot can still traverse the corridor and reach the inspection region, indicating that local navigation noise does not by itself induce stale-memory failure.

C.7 Case Study: FrozenLake Implicit Environmental Drifts

When the agent enters block (3,2):



Right after the drift, the agent still has experience from the source. The agent's prompt at the time looks like:

```
<|begin_of_sentence|>
You are an intelligent exploration agent navigating a frozen lake.
Your goal is to reach the destination with highest possible score while
avoiding holes. Analyze the current position and decide the next move.
Respond with only the action number (0, 1, 2, or 3) without any additional
explanation or formatting.

Map Size: 6 rows x 6 columns
Current Position: [3, 2]

There are a total of 2 destination(s).
Destinations: (3, 5), (5, 5)

......

--- Historical Experience from Similar States ---
You have been at this position before. Here are 2 previous experience(s):

  Action taken: 2
  Result Position: [3, 3]
  Result Tile: F
  Max score achievable from this path: 0.5

  Action taken: 1
  Result Position: [4, 2]
  Result Tile: F
  Max score achievable from this path: 1.0

......

Which action effectively moves you closer to the HIGHEST SCORE goal?
Respond with only the action number (0, 1, 2, or 3).
```

Through concatenating experiences' s' with other experiences whose s is the same, the agent constructs its internal world model and therefore realizes that the max score action 2 and action 1 can lead to.

The agent will proceed to move to the destination (3,5).GLOVE starts probing when a memory inconsistency is detected.

After the agent arrives at the (3,2) again with updated experience:

```
<|begin_of_sentence|>
You are an intelligent exploration agent navigating a frozen lake.
Your goal is to reach the destination with highest possible score while
avoiding holes. Analyze the current position and decide the next move.
Respond with only the action number (0, 1, 2, or 3) without any additional
explanation or formatting.

Map Size: 6 rows x 6 columns
Current Position: [3, 2]

There are a total of 2 destination(s).
Destinations: (3, 5), (5, 5)

......

--- Historical Experience from Similar States ---
You have been at this position before. Here are 2 previous experience(s):

  Action taken: 2
```

```
  Result Position: [3, 3]
  Result Tile: F
  Max score achievable from this path: 0.5

  Action taken: 1
  Result Position: [4, 2]
  Result Tile: F
  Max score achievable from this path: 0.5

......

Which action effectively moves you closer to the HIGHEST SCORE goal?
Respond with only the action number (0, 1, 2, or 3).
```

The agent now notices that neither path leads to a full score and starts exploring again.

After some time, it reaches (5,5). GLOVE realizes the memory conflicts and starts probing again, re-align the internal world model in the experience base with the environment.

```
<|begin_of_sentence|>
You are an intelligent exploration agent navigating a frozen lake.
Your goal is to reach the destination with highest possible score while
avoiding holes. Analyze the current position and decide the next move.
Respond with only the action number (0, 1, 2, or 3) without any additional
explanation or formatting.

Map Size: 6 rows x 6 columns
Current Position: [3, 2]

There are a total of 2 destination(s).
Destinations: (3, 5), (5, 5)

......

--- Historical Experience from Similar States ---
You have been at this position before. Here are 2 previous experience(s):

  Action taken: 2
  Result Position: [3, 3]
  Result Tile: F
  Max score achievable from this path: 0.5

  Action taken: 1
  Result Position: [4, 2]
  Result Tile: F
  Max score achievable from this path: 1.0

......

Which action effectively moves you closer to the HIGHEST SCORE goal?
Respond with only the action number (0, 1, 2, or 3).
```

D Additional Experiment Results

D.1 Backbone Sweep Overview

Table 4: Backbone sweep overview. Each cell reports the change in percentage caused by a drift. The shaded rows show the performance of GLOVE-augmented agents. We annotate the performance gap relative to the baseline (e.g., (↑56.3) indicates improvement).

	Explicit Structural Drift (Drift II)			Implicit Hidden Drift	
Backbone	**WebShop**	**FrozenLake**	**MountainCar**	**WebShop**	**FrozenLake**
Llama3.1-8B	7.5 → 5	0 → 1.2	93.8 → 65	34.4 → 36.2	13.1 → 52.5
+GLOVE	67.5 → 75 (↑70)	36.2 → 67.5 (↑66.3)	93.8 → 68.8 (↑3.8)	42.5 → 61.6 (↑25.4)	12.5 → 87.5 (↑35)
Llama3.3-70B	8.8 → 7.5	83.7 → 0	0 → 0	95.3 → 76.6	39.4 → 96.2
+GLOVE	95 → 88.8 (↑81.3)	82.5 → 73.8 (↑73.8)	0 → 0	100 → 83.4 (↑6.8)	40 → 97.5 (↑1.3)
Qwen2.5-7B	7.5 → 3.8	6.2 → 0	0 → 0	35.6 → 69.7	40.6 → 97.5
+GLOVE	63.7 → 71.2 (↑67.4)	78.8 → 68.8 (↑68.8)	0 → 0	46.9 → 75 (↑5.3)	41.9 → 100 (↑2.5)
Qwen3-30B	8.8 → 7.5	80 → 13.8	100 → 100	75 → 100	88.8 → 46.9
+GLOVE	95 → 86.2 (↑78.7)	80 → 75 (↑61.2)	100 → 100	75 → 100	85 → 47.5 (↑0.6)
GPT-4o	5 → 5	3.8 → 11.2	96.2 → 90	99.4 → 76.6	60 → 51.9
+GLOVE	90 → 92.5 (↑87.5)	77.5 → 76.2 (↑65)	100 → 100 (↑10)	98.4 → 91.6 (↑15)	51.2 → 96.2 (↑44.3)
Grok-3	8.8 → 7.5	60 → 11.2	100 → 100	96.2 → 78.4	35.6 → 68.1
+GLOVE	87.5 → 93.8 (↑86.3)	75 → 71.2 (↑60)	100 → 100	95 → 98.4 (↑20)	33.8 → 98.8 (↑30.7)
DeepSeek-V3.2	5 → 6.2	0 → 0	0 → 0	95.3 → 76.9	26.9 → 75.6
+GLOVE	85 → 93.8 (↑87.6)	68.8 → 70 (↑70)	0 → 0	94.4 → 97.5 (↑20.6)	27.5 → 95.6 (↑20)

D.2 Experiment Results: Explicit Environmental Drifts

Table 5: **GPT-4o: GLOVE's robustness to environmental drift (Explicit).** Success rate in percent. The shaded rows show the performance of GLOVE-augmented agents. We annotate the performance gap relative to the baseline (e.g., (↑56.3) indicates improvement).

	VERIFIER-VISIBLE STRUCTURAL DRIFT								
	WebShop (Semantic Shift)			**FrozenLake** (topology shift)			**MountainCar** (dynamics shift)		
Method	**Source**	**→ Drift I**	**→ Drift II**	**Source**	**→ Drift I**	**→ Drift II**	**Source**	**→ Drift I**	**→ Drift II**
No Memory (Plain)	15	5	0	0	5	0	85	75	55
Vanilla	85	0	0	85	0	0	100	100	90
+GLOVE	85	85 (↑85)	95 (↑95)	80	75 (↑75)	80 (↑80)	100	100	100 (↑10)
MemoryBank	85	20	20	80	0	45	95	100	100
+GLOVE	100	90 (↑70)	85 (↑65)	80	70 (↑70)	65 (↑20)	95	100	100
Voyager	85	0	0	85	0	0	100	90	100
+GLOVE	80	90 (↑90)	95 (↑95)	85	85 (↑85)	75 (↑75)	100	100 (↑10)	100
Generative Agent	80	0	0	80	15	0	100	95	70
+GLOVE	75	95 (↑95)	95 (↑95)	80	80 (↑65)	85 (↑85)	95	100 (↑5)	100 (↑30)

Table 6: **Grok-3: GLOVE's robustness to environmental drift (Explicit).** Success rate in percent. The shaded rows show the performance of GLOVE-augmented agents. We annotate the performance gap relative to the baseline (e.g., (↑56.3) indicates improvement).

VERIFIER-VISIBLE STRUCTURAL DRIFT									
	WebShop (Semantic Shift)			**FrozenLake** (topology shift)			**MountainCar** (dynamics shift)		
Method	**Source**	**→ Drift I**	**→ Drift II**	**Source**	**→ Drift I**	**→ Drift II**	**Source**	**→ Drift I**	**→ Drift II**
No Memory (Plain)	20	5	15	25	0	0	100	90	95
Vanilla	85	0	0	100	75	0	100	100	100
+GLOVE	90	80 (↑80)	95 (↑95)	90	80 (↑5)	70 (↑70)	100	100	100
MemoryBank	90	35	30	95	70	0	100	100	100
+GLOVE	85	90 (↑55)	95 (↑65)	95	65 (↓5)	80 (↑80)	100	100	100
Voyager	85	0	0	95	95	0	100	100	100
+GLOVE	85	90 (↑90)	95 (↑95)	85	70 (↓25)	65 (↑65)	100	100	100
Generative Agent	85	0	0	95	0	45	100	100	100
+GLOVE	85	90 (↑90)	90 (↑90)	95	85 (↑85)	70 (↑25)	100	100	100

Table 7: **DeepSeek-V3.2: GLOVE's robustness to environmental drift (Explicit).** Success rate in percent. The shaded rows show the performance of GLOVE-augmented agents. We annotate the performance gap relative to the baseline (e.g., (↑56.3) indicates improvement).

VERIFIER-VISIBLE STRUCTURAL DRIFT									
	WebShop (Semantic Shift)			**FrozenLake** (topology shift)			**MountainCar** (dynamics shift)		
Method	**Source**	**→ Drift I**	**→ Drift II**	**Source**	**→ Drift I**	**→ Drift II**	**Source**	**→ Drift I**	**→ Drift II**
No Memory (Plain)	10	20	5	0	0	0	0	0	0
Vanilla	85	0	0	95	0	0	0	0	0
+GLOVE	100	70 (↑70)	95 (↑95)	85	75 (↑75)	75 (↑75)	0	0	0
MemoryBank	85	20	25	80	0	0	0	0	0
+GLOVE	85	90 (↑70)	95 (↑70)	80	70 (↑70)	65 (↑65)	0	0	0
Voyager	85	0	0	95	0	0	0	0	0
+GLOVE	75	90 (↑90)	95 (↑95)	90	65 (↑65)	70 (↑70)	0	0	0
Generative Agent	85	0	0	80	0	0	0	0	0
+GLOVE	75	90 (↑90)	90 (↑90)	100	65 (↑65)	70 (↑70)	0	0	0

Table 8: **Llama3.3-70B: GLOVE's robustness to environmental drift (Explicit).** Success rate in percent. The shaded rows show the performance of GLOVE-augmented agents. We annotate the performance gap relative to the baseline (e.g., (↑56.3) indicates improvement).

VERIFIER-VISIBLE STRUCTURAL DRIFT									
	WebShop (Semantic Shift)			**FrozenLake** (topology shift)			**MountainCar** (dynamics shift)		
Method	**Source**	**→ Drift I**	**→ Drift II**	**Source**	**→ Drift I**	**→ Drift II**	**Source**	**→ Drift I**	**→ Drift II**
No Memory (Plain)	50	50	0	0	0	0	0	0	0
Vanilla	90	0	0	85	85	0	0	0	0
+GLOVE	95	95 (↑95)	85 (↑85)	85	80 (↓5)	75 (↑75)	0	0	0
MemoryBank	100	35	30	90	85	0	0	0	0
+GLOVE	100	95 (↑60)	85 (↑55)	85	85	75 (↑75)	0	0	0
Voyager	90	0	0	85	85	0	0	0	0
+GLOVE	90	95 (↑95)	95 (↑95)	85	85	75 (↑75)	0	0	0
Generative Agent	95	0	0	85	80	0	0	0	0
+GLOVE	90	95 (↑95)	90 (↑90)	85	80	70 (↑70)	0	0	0

Table 9: **Llama3.1-8B: GLOVE's robustness to environmental drift (Explicit).** Success rate in percent. The shaded rows show the performance of GLOVE-augmented agents. We annotate the performance gap relative to the baseline (e.g., (↑56.3) indicates improvement).

VERIFIER-VISIBLE STRUCTURAL DRIFT									
	WebShop (Semantic Shift)			**FrozenLake** (topology shift)			**MountainCar** (dynamics shift)		
Method	**Source**	**→ Drift I**	**→ Drift II**	**Source**	**→ Drift I**	**→ Drift II**	**Source**	**→ Drift I**	**→ Drift II**
No Memory (Plain)	30	0	0	0	0	0	50	60	30
Vanilla	80	0	0	80	0	0	95	95	95
+GLOVE	100	60 (↑60)	95 (↑95)	80	75 (↑75)	50 (↑50)	85	95	95
MemoryBank	95	30	20	75	0	5	75	90	0
+GLOVE	55	75 (↑45)	25 (↑5)	35	70 (↑70)	75 (↑70)	60	100 (↑10)	0
Voyager	75	0	0	10	0	0	95	90	95
+GLOVE	85	75 (↑75)	90 (↑90)	65	0	75 (↑75)	95	85 (↓5)	90 (↓5)
Generative Agent	100	0	0	75	0	0	100	100	70
+GLOVE	90	60 (↑60)	90 (↑90)	80	0	70 (↑70)	80	95 (↓5)	90 (↑20)

Table 10: **Qwen3-30B: GLOVE's robustness to environmental drift (Explicit).** Success rate in percent. The shaded rows show the performance of GLOVE-augmented agents. We annotate the performance gap relative to the baseline (e.g., (↑56.3) indicates improvement).

VERIFIER-VISIBLE STRUCTURAL DRIFT									
	WebShop (Semantic Shift)			**FrozenLake** (topology shift)			**MountainCar** (dynamics shift)		
Method	**Source**	**→ Drift I**	**→ Drift II**	**Source**	**→ Drift I**	**→ Drift II**	**Source**	**→ Drift I**	**→ Drift II**
No Memory (Plain)	0	30	0	100	0	0	100	100	100
Vanilla	65	0	0	100	80	0	100	100	100
+GLOVE	85	95 (↑95)	85 (↑85)	100	80	75 (↑75)	100	100	100
MemoryBank	95	35	30	100	80	55	100	100	100
+GLOVE	95	95 (↑60)	80 (↑50)	100	80	75 (↑20)	100	100	100
Voyager	90	0	0	100	80	0	100	100	100
+GLOVE	85	95 (↑95)	90 (↑90)	100	80	75 (↑75)	100	100	100
Generative Agent	75	0	0	100	80	0	100	100	100
+GLOVE	95	95 (↑95)	90 (↑90)	100	80	75 (↑75)	100	100	100

Table 11: **Qwen2.5-7B: GLOVE's robustness to environmental drift (Explicit).** Success rate in percent. The shaded rows show the performance of GLOVE-augmented agents. We annotate the performance gap relative to the baseline (e.g., (↑56.3) indicates improvement).

VERIFIER-VISIBLE STRUCTURAL DRIFT									
	WebShop (Semantic Shift)			**FrozenLake** (topology shift)			**MountainCar** (dynamics shift)		
Method	**Source**	**→ Drift I**	**→ Drift II**	**Source**	**→ Drift I**	**→ Drift II**	**Source**	**→ Drift I**	**→ Drift II**
No Memory (Plain)	70	5	0	100	0	0	0	0	0
Vanilla	90	0	0	100	0	0	0	0	0
+GLOVE	95	75 (↑75)	75 (↑75)	100	80 (↑80)	75 (↑75)	0	0	0
MemoryBank	100	30	15	100	25	0	0	0	0
+GLOVE	100	15 (↓15)	50 (↑35)	100	80 (↑55)	75 (↑75)	0	0	0
Voyager	100	0	0	100	0	0	0	0	0
+GLOVE	100	85 (↑85)	85 (↑85)	100	75 (↑75)	50 (↑50)	0	0	0
Generative Agent	100	0	0	100	0	0	0	0	0
+GLOVE	100	80 (↑80)	75 (↑75)	100	80 (↑80)	75 (↑75)	0	0	0

D.2.1 Direct Adaptive Baselines and Statistical Reliability

Adaptive baseline protocols. We evaluate six direct online update rules on Qwen3-30B FrozenLake explicit drift. LATESTOVERWRITE retains only the latest observed outcome for each (s, a) pair; RECENTK-K3 retains the three most recent transitions; SLIDINGWINDOW-W5 estimates the empirical outcome distribution from the five most recent transitions; RECENCYWEIGHTED-H5 exponentially weights the pair-local history with an observation half-life of five; PAGEHINKLEYRESET applies a pair-specific Page–Hinkley change detector and resets the corresponding local history after a detected outcome change; and PERIODICREFRESH-S30 clears the memory table every 30 natural environment steps. All methods use the same deterministic maps, prompt, action parser, model settings, 20-step episode cap, and 20 Source episodes.

In the natural protocol, adaptive baselines update only from their own trajectories, whereas GLOVE uses its standard verification procedure. In the fixed-budget protocol, every method receives exactly $B = 310$ post-drift environment interactions; GLOVE uses 280 task interactions and 30 verification interactions, while each baseline uses all 310 interactions for task execution. If the budget expires during an episode, its interactions and memory updates consume budget, but the unfinished episode is not counted as completed.

Paired multi-run analysis. To assess run-to-run variability, we additionally evaluate Vanilla, MemoryBank, and their GLOVE-augmented counterparts on the same Qwen3-30B FrozenLake explicit-drift setting over 10 complete paired runs, with 20 evaluation episodes per phase. For each run and backend, the baseline and its GLOVE variant are initialized from byte-equivalent source-memory snapshots and evaluated on identical task instances. Because memory is updated sequentially within each run, we use paired run-level differences as the inferential unit. We report pooled success/failure counts, run-level mean and sample standard deviation, 95% t-intervals for paired percentage-point differences, Bonferroni-adjusted exact sign tests over four planned comparisons, and Cohen's h for binary outcomes.

Table 12: **Paired multi-run statistical analysis on Qwen3-30B FrozenLake explicit drift.** Each condition contains 10 complete runs with 20 episodes per phase. "S/F" denotes pooled successes/failures. Mean $\pm$ SD is computed over run-level success rates. Confidence intervals are 95% t-intervals over paired percentage-point differences. p-values are Bonferroni-adjusted exact sign tests over four planned comparisons; h denotes Cohen's h.

Backend	Phase	Baseline S/F	Baseline Mean $\pm$ SD	+GLOVE S/F	+GLOVE Mean $\pm$ SD	Gap [95% CI]	Adj. p / h
Vanilla	Drift I	160/40	$80.0 \pm 0.0\%$	160/40	$80.0 \pm 0.0\%$	0.0 pp [0.0, 0.0]	1.000 / 0.000
Vanilla	Drift II	0/200	$0.0 \pm 0.0\%$	150/50	$75.0 \pm 0.0\%$	+75.0 pp [75.0, 75.0]	0.0078 / 2.094
MemoryBank	Drift I	131/69	$65.5 \pm 30.8\%$	160/40	$80.0 \pm 0.0\%$	+14.5 pp [−7.5, 36.5]	1.000 / 0.328
MemoryBank	Drift II	37/163	$18.5 \pm 7.5\%$	150/50	$75.0 \pm 0.0\%$	+56.5 pp [51.2, 61.8]	0.0078 / 1.205

Source success is 200/200 for every condition and is omitted from Table 12. Both Drift-II gains are positive in all 10 paired runs and remain significant after Bonferroni correction. The MemoryBank Drift-I comparison contains two positive differences and eight ties and is inconclusive. Zero observed standard deviation means that the phase-level success counts are identical across these runs, rather than that provider-side stochasticity is absent. For settings without repeated paired runs, we report descriptive results without making statistical-significance claims.

D.3 Experiment Results: Implicit Environmental Drifts

Table 13: **GPT-4o: GLOVE's robustness to environmental drift (Implicit).** Score obtained. The shaded rows show the performance of GLOVE-augmented agents. We annotate the performance gap relative to the baseline (e.g., (↑56.3) indicates improvement).

VERIFIER-HIDDEN DRIFT				
	WebShop (semantic shift)		**FrozenLake** (reward reversal)	
Method	**Source**	→ **Hidden drift**	**Source**	→ **Hidden drift**
No Memory (Plain)	82.5	87.5	0	0
Vanilla	100	75	47.5	50
+GLOVE	97.5	93.8 (↑18.8)	35	97.5 (↑47.5)
MemoryBank	100	81.2	62.5	57.5
+GLOVE	98.8	98.8 (↑17.5)	67.5	97.5 (↑40)
Voyager	98.8	75	67.5	50
+GLOVE	98.8	98.8 (↑23.8)	40	97.5 (↑47.5)
Generative Agent	98.8	75	62.5	50
+GLOVE	98.8	75	62.5	92.5 (↑42.5)

Table 14: **Grok-3: GLOVE's robustness to environmental drift (Implicit).** Score obtained. The shaded rows show the performance of GLOVE-augmented agents. We annotate the performance gap relative to the baseline (e.g., (↑56.3) indicates improvement).

VERIFIER-HIDDEN DRIFT				
	WebShop (semantic shift)		**FrozenLake** (reward reversal)	
Method	**Source**	→ **Hidden drift**	**Source**	→ **Hidden drift**
No Memory (Plain)	76.2	96.2	0	0
Vanilla	96.2	75	30	90
+GLOVE	97.5	98.8 (↑23.8)	30	100 (↑10)
MemoryBank	97.5	81.2	15	82.5
+GLOVE	88.8	98.8 (↑17.5)	20	97.5 (↑15)
Voyager	98.8	75	62.5	50
+GLOVE	100	98.8 (↑23.8)	27.5	100 (↑50)
Generative Agent	92.5	82.5	35	50
+GLOVE	93.8	97.5 (↑15)	57.5	97.5 (↑47.5)

Table 15: **DeepSeek-V3.2: GLOVE's robustness to environmental drift (Implicit).** Score obtained. The shaded rows show the performance of GLOVE-augmented agents. We annotate the performance gap relative to the baseline (e.g., (↑56.3) indicates improvement).

	VERIFIER-HIDDEN DRIFT			
	WebShop (semantic shift)		**FrozenLake** (reward reversal)	
Method	**Source**	**→ Hidden drift**	**Source**	**→ Hidden drift**
No Memory (Plain)	80	95	0	0
Vanilla	93.8	75	52.5	50
+GLOVE	86.2	97.5 (↑22.5)	50	92.5 (↑42.5)
MemoryBank	100	82.5	10	82.5
+GLOVE	100	97.5 (↑15)	10	95 (↑12.5)
Voyager	90	75	22.5	82.5
+GLOVE	93.8	97.5 (↑22.5)	30	95 (↑12.5)
Generative Agent	97.5	75	22.5	87.5
+GLOVE	97.5	97.5 (↑22.5)	20	100 (↑12.5)

Table 16: **Llama3.3-70B: GLOVE's robustness to environmental drift (Implicit).** Score obtained. The shaded rows show the performance of GLOVE-augmented agents. We annotate the performance gap relative to the baseline (e.g., (↑56.3) indicates improvement).

	VERIFIER-HIDDEN DRIFT			
	WebShop (semantic shift)		**FrozenLake** (reward reversal)	
Method	**Source**	**→ Hidden drift**	**Source**	**→ Hidden drift**
No Memory (Plain)	91.2	80	0	0
Vanilla	86.2	75	42.5	95
+GLOVE	100	75	42.5	100 (↑5)
MemoryBank	98.8	81.2	35	90
+GLOVE	100	88.8 (↑7.5)	35	90
Voyager	96.2	75	40	100
+GLOVE	100	95 (↑20)	40	100
Generative Agent	100	75	40	100
+GLOVE	100	75	42.5	100

Table 17: **Llama3.1-8B: GLOVE's robustness to environmental drift (Implicit).** Score obtained. The shaded rows show the performance of GLOVE-augmented agents. We annotate the performance gap relative to the baseline (e.g., (↑56.3) indicates improvement).

	VERIFIER-HIDDEN DRIFT			
	WebShop (semantic shift)		**FrozenLake** (reward reversal)	
Method	**Source**	→ **Hidden drift**	**Source**	→ **Hidden drift**
No Memory (Plain)	47.5	60	0	0
Vanilla	30	36.2	10	50
+GLOVE	42.5	76.2 (↑40)	12.5	95 (↑45)
MemoryBank	26.2	27.5	7.5	65
+GLOVE	50	67.5 (↑40)	5	65
Voyager	32.5	26.2	20	35
+GLOVE	15	42.5 (↑16.2)	15	100 (↑65)
Generative Agent	48.8	55	15	60
+GLOVE	62.5	60 (↑5)	17.5	90 (↑30)

Table 18: **Qwen3-30B: GLOVE's robustness to environmental drift (Implicit).** Score obtained. The shaded rows show the performance of GLOVE-augmented agents. We annotate the performance gap relative to the baseline (e.g., (↑56.3) indicates improvement).

	VERIFIER-HIDDEN DRIFT			
	WebShop (semantic shift)		**FrozenLake** (reward reversal)	
Method	**Source**	→ **Hidden drift**	**Source**	→ **Hidden drift**
No Memory (Plain)	75	100	0	0
Vanilla	75	100	90	50
+GLOVE	75	100	85	50
MemoryBank	75	100	90	37.5
+GLOVE	75	100	85	40 (↑2.5)
Voyager	75	100	85	50
+GLOVE	75	100	85	50
Generative Agent	75	100	90	50
+GLOVE	75	100	85	50

Table 19: **Qwen2.5-7B: GLOVE's robustness to environmental drift (Implicit).** Score obtained. The shaded rows show the performance of GLOVE-augmented agents. We annotate the performance gap relative to the baseline (e.g., (↑56.3) indicates improvement).

	VERIFIER-HIDDEN DRIFT			
	WebShop (semantic shift)		**FrozenLake** (reward reversal)	
Method	**Source**	→ **Hidden drift**	**Source**	→ **Hidden drift**
No Memory (Plain)	0	15	0	0
Vanilla	30	75	42.5	100
+GLOVE	47.5	75	42.5	100
MemoryBank	17.5	53.8	35	95
+GLOVE	45	75 (↑21.3)	37.5	100 (↑5)
Voyager	50	75	42.5	95
+GLOVE	47.5	75	45	100 (↑5)
Generative Agent	45	75	42.5	100
+GLOVE	47.5	75	42.5	100

D.3.1 Full-Trajectory Context Ablation

We further test whether the failure of static memory under explicit drift is mainly caused by the narrowness of local state-matched retrieval. We consider two broader-context variants in WebShop. First, FULLTRAJ-STATIC provides a complete successful source-environment trajectory at every decision step during drifted evaluation. Second, we include a trajectory-aware Voyager-style memory that uses both trajectory summaries and full-trajectory context, representing a richer episodic/procedural memory backend.

Table 20: **Full-trajectory context ablation on WebShop explicit drift.** Success rate in percent over 20 episodes. FULLTRAJ-STATIC provides a complete successful source-environment trajectory at each decision step. The Voyager-style variant uses both a full trajectory and a trajectory summary, representing a broader trajectory-aware memory backend.

Method	Source	→ Drift I	→ Drift II
Static Memory (StateMatched)	85	0	0
Static Memory (FULLTRAJ-STATIC)	95	0	0
StateMatched + GLOVE (Vanilla)	85	85	95
Trajectory Memory + GLOVE (Voyager)	80	90	95

Table 20 shows that broader trajectory context alone does not remove stale-memory failure. FULLTRAJ-STATIC improves source-environment success from 85% to 95%, but still collapses to 0% after both drifts. By contrast, once GLOVE is added, both backends remain robust after drift: state-matched Vanilla + GLOVE reaches 85% and 95%, while Voyager + GLOVE reaches 90% and 95% on Drift I and Drift II. These results suggest that broader episodic context can improve performance in the stationary source environment, but it does not by itself enable the agent to discount stale source-environment experience after drift. The main benefit under drift comes from explicit memory verification and realignment.

D.4 Positioning with Respect to Adaptive Model-Based RL

GLOVE share similarities with adaptive model-based RL as both use fresh interaction to correct beliefs about environment behavior. This appendix is included to clarify the boundary of this connection, rather than to treat RL algorithms as the primary baselines. Classical adaptive model-based RL typically learns or updates a reward model and uses it for reward-conditioned planning or policy improvement. GLOVE instead operates as a deployment-time memory-validation layer for memory-augmented LLM agents: it asks whether a retrieved memory entry, which may be a textual observation, a procedural trace, a trajectory summary, or a semantic association, should still be trusted before it is reused as prompt context.

This distinction is important in text-rich agent tasks. In WebShop, for example, the effective decision context includes natural-language instructions, noisy HTML, URLs, product attributes, and page affordances. A drift may be semantic, such as a changed grounding of a user preference, even when the surface interface remains similar. A tabular or low-dimensional model-based RL baseline would first require a stable Markov abstraction and a reward model over this textual state/action space. Learned latent world models are possible in principle, but then the comparison would conflate memory validation with representation learning, semantic grounding, and policy optimization. GLOVE addresses a narrower deployment-time problem in which, given the memory abstraction already consumed by the LLM agent, the system locally falsifies stale memory and realigns it with current observations or feedback.

Where adaptive model-based RL is naturally applicable, we include a small FrozenLake positioning check with Dyna-Q and Dyna-Q+ [17, 18]. As shown in Table 21, these baselines adapt well under the smallest map drift, but degrade substantially under the larger structured drift. In the same large-drift setting, GLOVE preserves its pre-drift success rate. This comparison should be interpreted as a scope check rather than a fully symmetric RL benchmark that GLOVE is not a replacement for model-based RL, and adaptive RL is not the main target class of this paper. The intended takeaway is that stale decision memory is a distinct failure mode in LLM agents, especially when memory is

textual or procedural, and that local memory verification can be complementary to policy learning or planning methods.



(a) Small-drift setting: source map. (b) Small-drift setting: drifted map. (c) Large-drift setting: source map. (d) Large-drift setting: drifted map.

Figure 12: FrozenLake maps used for the experiments in Table 21.

Table 21: Positioning check against classical adaptive model-based RL in FrozenLake.

Setting	Method	Success before ↑	Success after ↑
Small drift	Dyna-Q	0.74 ± 0.15	0.90 ± 0.01
Small drift	Dyna-Q+	0.69 ± 0.17	0.80 ± 0.11
Large drift	Dyna-Q	0.20 ± 0.24	0.27 ± 0.26
Large drift	Dyna-Q+	0.52 ± 0.22	0.21 ± 0.22
Large drift	GLOVE	0.80	0.80

D.5 Cost Analysis

Active verification introduces additional environment interactions, so its benefit should be evaluated together with its cost. We analyze this cost from three perspectives: when verification is triggered, how the probing budget affects robustness, and how matching granularity changes the number of verification steps.

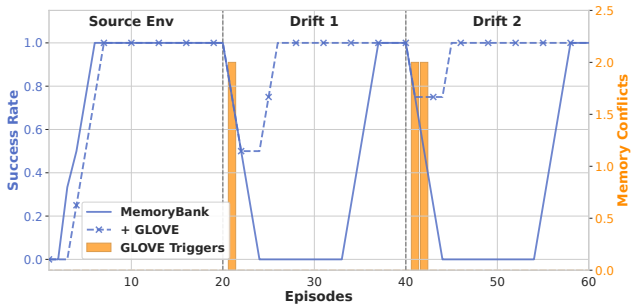


Figure 13: Verification dynamics under explicit drift in WebShop. GLOVE triggers verification sparsely, with conflict spikes concentrated around drift events, enabling rapid recovery without continuous probing.

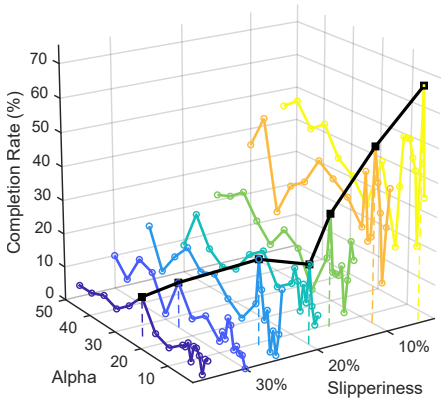
As shown in Fig. 13, GLOVE does not increase interaction uniformly throughout deployment. Instead, verification is triggered sparsely and concentrates around drift events, where memory conflicts spike. This allows GLOVE to recover quickly after environmental change while maintaining low overhead during stable periods.

Table 22 provides a cost-normalized view of GLOVE's verification overhead. Total environment interactions include both normal task-execution steps and verification steps, while active-probe interactions count only the additional steps used for direct GLOVE probing. In the source environment, GLOVE adds almost no overhead, increasing total interactions from 116 to 118 with zero

Table 22: Cost-normalized comparison on WebShop explicit drift. Total interactions include both task-execution and verification steps. Active-probe interactions count only direct GLOVE verification probes.

Method	Phase	Total env. interactions	Active-probe interactions
MemoryBank	Source	116	0
MemoryBank	Drift I	128	0
MemoryBank	Drift II	136	0
MemoryBank + GLOVE	Source	118	0
MemoryBank + GLOVE	Drift I	254	86
MemoryBank + GLOVE	Drift II	419	151

active probes. After drift, the additional cost is concentrated around realignment: total interactions increase from 128/136 to 254/419 in Drift I/II, with 86/151 active-probe interactions. This indicates that GLOVE's benefit does not come from uniformly using more interactions, but from selectively spending verification budget when memory-environment conflicts arise.



Slipperiness	Best α
35%	25
30%	25
25%	9
20%	4
15%	9
10%	6
5%	2

Figure 14: Effect of probing budget α in FrozenLake. Larger probing budgets improve robustness under higher stochasticity, while smaller budgets are sufficient in more stable environments.

Fig. 14 shows that the probing budget α acts as a cost–reliability control parameter. Larger α improves robustness under more stochastic dynamics by providing more reliable transition estimates, while smaller budgets are sufficient in more stable settings. Thus, α can be adjusted according to the expected noise level of the environment.

Table 23: WebShop trigger-sensitivity study. η controls how much attribute information is retained in the confirmation-state representation. Moderate matching granularity saturates performance while keeping verification cost low.

η	Avg. Reward	Avg. Steps / Episode	Avg. Verify Env Steps / Episode
0.0	0.408	10.93	38.83
0.5	0.683	8.85	1.48
1.0	0.683	8.67	1.40

Finally, Table 23 shows that verification cost also depends on the granularity of counterpart matching. Overly coarse matching ($\eta = 0.0$) triggers excessive verification, leading to 38.83 verification steps per episode. In contrast, moderate and strict matching ($\eta = 0.5$ and $\eta = 1.0$) attain the same average reward while keeping verification overhead comparably low, at 1.48 and 1.40 verification steps per episode respectively. Overall, these results show that GLOVE's overhead is selective and controllable rather than continuously incurred.

D.6 JetRover Real-World Robotic Result

Figure 3 reports the real-robot evaluation on JetRover. The robot searches for the `blue` block in a cardboard maze. After source-memory formation, we induce semantic drift by swapping the `blue` and `green` blocks. Static Memory keeps visiting the stale target location after the swap, while GLOVE detects the semantic mismatch, updates memory, and recovers correct search behavior under physical nuisance variation.

D.7 Result Analysis

While GLOVE yields substantial improvements in most settings, it does not uniformly dominate across all configurations. In particular, in **FrozenLake Drift I** (e.g., Table 6), GLOVE occasionally underperforms the baseline. The reason is that the induced drift in this setting is relatively mild, such that high-capacity models can already maintain near-optimal performance without explicit verification. In these cases, additional probing may introduce unnecessary perturbations, whereas under more severe drift (e.g., **Drift II**), GLOVE consistently outperforms the baseline by actively correcting stale transitions.

Another source of performance degradation is the use of small backbone models. As shown in Tables 9 and 11, models such as **Llama3.1-8B** and **Qwen2.5-7B** exhibit limited ability to interpret and exploit the verified transition summaries produced by GLOVE. This suggests that effective utilization of corrected memory requires a certain level of instruction-following and reasoning capacity. When this condition is violated, verification may fail to translate into improved action selection.

These observations highlight that GLOVE is most beneficial when environmental drift is substantial and when the backbone model can reliably incorporate corrective feedback. They also indicate a cost-benefit trade-off. In low-drift regimes, the marginal gains of verification may not justify additional interaction. Nevertheless, across the full benchmark tests, GLOVE remains a positive augmentation, particularly in regimes where stale memory would otherwise lead to catastrophic failure.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction reflect the paper's contributions and scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations in Appendix A.1.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Assumptions and proof are provided in the paper.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Source code and information needed to reproduce the main results are provided.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We release the GLOVE implementation, the four memory backends (vanilla, MemoryBank, Voyager, and Generative Agents), and the augmented evaluation environments (the modified WebShop purchase flow, the FrozenLake map variants, and the MountainCar dynamics variants), together with installation and run instructions, at https://github.com/NICE-HKU/GLOVE. These reproduce the main memory-realignment experiments in Tables 1 and 2. The rule-based adaptive memory baselines and the cost and paired-run analyses added during revision are not released as separate scripts, as they are short variants of the released memory-update interface that are fully specified in Appendix D.2.1 and Appendix D.5. The real-robot results require the physical JetRover platform and cannot be reproduced in simulation.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: We cover related details in the Appendix D.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We report a 10-run paired analysis (run-level mean $\pm$ sample SD, 95% CI, Bonferroni-adjusted exact sign tests, and Cohen's h) for the Qwen3-30B FrozenLake explicit-drift setting in Appendix D.2.1. Settings without repeated paired runs are reported descriptively, without significance claims.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All experiments are inference-only: no model is trained or fine-tuned, and no local GPU is used. Every LLM backbone is queried through the provider's cloud inference API; the memory backends and environments run on a single CPU machine with negligible storage, and API inference dominates wall-clock time and cost. The JetRover experiment additionally uses the robot's onboard compute; no unreported runs required further compute.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research respects the NeurIPS Code of Ethics in every aspect.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the broader impacts of this work in Appendix A.2.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our works provide a plug-and-play memory-augment module that does not have related risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Their contribution has already been properly acknowledged and credited.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not introduce new assets other than source code of implementation.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not include crowdsourcing and research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: N/A

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: LLMs are used as one of the main components of the proposed LLM memory system module.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.