Memory Retention Is Not Enough to Master Memory Tasks in Reinforcement Learning

Oleg Shchendrigin, Egor Cherepanov, Alexey K. Kovalev, Aleksandr I. Panov

Keywords: Reinforcement Learning, POMDPs, Memory.

Summary

Existing Reinforcement Learning (RL) benchmarks and memory-augmented agents focus primarily on retention, leaving the equally critical ability of memory rewriting largely unexplored. To address this gap, we introduce a benchmark that explicitly tests continual memory updating under partial observability, i.e. the natural setting where an agent must rely on memory rather than current observations, and use it to compare recurrent, transformer-based, and structured memory architectures. Our experiments reveal that classic recurrent models, despite their simplicity, demonstrate greater flexibility and robustness in memory rewriting tasks than modern structured memories, which succeed only under narrow conditions, and transformer-based agents, which often fail beyond trivial retention cases. These findings expose a fundamental limitation of current approaches and emphasize the necessity of memory mechanisms that balance stable retention with adaptive updating. Our work highlights this overlooked challenge, introduces benchmarks to evaluate it, and offers insights for designing future RL agents with explicit and trainable forgetting mechanisms.

Contribution(s)

1. **Benchmarks for rewriting.** We introduce `Endless T-Maze` and `Color-Cubes`, two families of environments, which together provide four different options to isolate the ability to perform continual, selective memory updates beyond simple cue retention.
   Context: Prior work developed endless memory tasks: Endless Mortar Mayhem, Endless Mystery Path, and Endless Searing Spotlights (Pleines et al., 2025) with the objective of evaluating agents' memory capabilities.
2. **Systematic evaluation.** We compare recurrent, transformer, and structured-memory agents, identifying when rewriting mechanisms succeed or fail and how performance degrades with increasing memory update frequency.
   Context: Structured-memory architectures were compared with recurrent policies (Le et al., 2024) under conditions of memory retention.
3. **Design principles.** We analyze architectural factors linked to rewriting competence, highlighting the effectiveness of *explicit, adaptive forgetting* (e.g., learnable forget gates) over cached-state or rigidly structured memories, and outline practical guidelines for evaluating memory mechanisms in partially observable tasks.
   Context: Prior work has shown that forgetting mechanisms in classical recurrent architectures introduce inductive biases that improve training stability (Morad et al., 2023b).

Memory Retention Is Not Enough to Master Memory Tasks in Reinforcement Learning

Oleg Shchendrigin[1,2], **Egor Cherepanov**[1,3], **Alexey K. Kovalev**[1,3], **Aleksandr I. Panov**[1,3]
o.shchendrigin@innopolis.university, cherepanov@axxx.tech

[1]**MIRAI** [2]**Innopolis University** [3]**AXXX**

Abstract

Effective decision-making in the real world depends on memory that is both stable and adaptive: environments change over time, and agents must retain relevant information over long horizons while also updating or overwriting outdated content when circumstances shift. Existing Reinforcement Learning (RL) benchmarks and memory-augmented agents focus primarily on retention, leaving the equally critical ability of memory rewriting largely unexplored. To address this gap, we introduce a benchmark that explicitly tests continual memory updating under partial observability, i.e. the natural setting where an agent must rely on memory rather than current observations, and use it to compare recurrent, transformer-based, and structured memory architectures. Our experiments reveal that classic recurrent models, despite their simplicity, demonstrate greater flexibility and robustness in memory rewriting tasks than modern structured memories, which succeed only under narrow conditions, and transformer-based agents, which often fail beyond trivial retention cases. These findings expose a fundamental limitation of current approaches and emphasize the necessity of memory mechanisms that balance stable retention with adaptive updating. Our work highlights this overlooked challenge, introduces benchmarks to evaluate it, and offers insights for designing future RL agents with explicit and trainable forgetting mechanisms.

1 Introduction

Many everyday tasks demand memory that can both preserve and revise information (Koslov et al., 2019; Sakon & Kiani, 2022; Bretton et al., 2024). A pedestrian following a sequence of directional signs must replace an earlier instruction (*"turn left at the park"*) when a new sign ahead indicates a detour, while a warehouse robot tracking object locations must update its internal map each time an item is moved. In both cases, previously stored information becomes misleading unless the agent rewrites its memory with current observations (see Figure 1 for a representation).

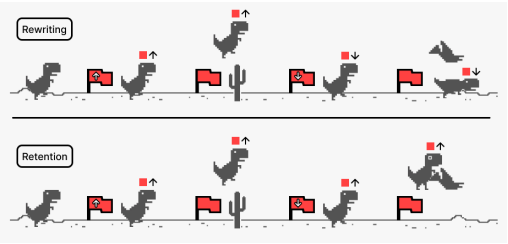

Figure 1: Overview of memory rewriting in partially observable decision-making.

Reinforcement Learning (RL) agents must exhibit the same capacity to revise stored information in order to act reliably in dynamic and partially observable environments. In realistic settings, observations are often incomplete or ambiguous: sensors are noisy, observations are occluded (limited field of view, temporary blockages), and the latent state cannot be inferred from a single frame (Lauri et al., 2022; Meng et al., 2021; Kurniawati, 2022; Zhang et al., 2023). Under partial observability,

effective decision-making requires agents to construct and maintain a memory that summarizes the relevant aspects of past experience (Parisotto et al., 2020; Cherepanov et al., 2026d; Le et al., 2024; Tao et al., 2025; Koepernik et al., 2025; Cherepanov et al., 2026c; Zelezetsky et al., 2025).

Memory in RL is not a single ability but a collection of interdependent processes: what information to encode, how long to retain it, and how to update or discard it (Ye et al., 2020; Mujawar et al., 2021; Aljundi et al., 2018). Most existing work focuses on *retention* – storing a cue and preserving it across time (Esslinger et al., 2022; Ni et al., 2023). However, many decision-making problems depend equally on *rewriting*: selectively discarding outdated content and integrating new evidence while avoiding interference from previously stored but now-irrelevant representations. Agents that only retain information risk acting on obsolete internal states. Despite its centrality in natural cognition, this ability to rewrite memory remains underexplored in artificial agents. Current benchmarks typically reward remembering a single cue until the episode ends, offering little incentive to assess continual, selective updating (Lampinen et al., 2021; Cherepanov et al., 2026a).

In this work, we isolate and systematically investigate *memory rewriting* in RL through four diagnostic tasks that enforce continual memory updates under partial observability. **`Endless T-Maze`** consists of sequential corridors where each new cue invalidates the previous one, requiring the agent to actively overwrite outdated instructions. **`Color-Cubes`** (in `Trivial`, `Medium`, and `Extreme` variants) is a grid-world task where colored cubes stochastically teleport, forcing agents to continually update their internal map and avoid acting on stale information.

We evaluate three representative families of memory-augmented RL agents – recurrent policies, transformer-based architectures, and structured external memories – providing a detailed characterization of their respective strengths, limitations, and generalization patterns. We focus on how each architecture adapts when memory rewriting becomes essential for success, exposing distinct strategies and failure modes across these paradigms.

Our contributions are threefold:

1. **Benchmarks for rewriting.** We introduce `Endless T-Maze` and `Color-Cubes`, two families of environments, which together provide four different options to isolate the ability to perform continual, selective memory updates beyond simple cue retention.
2. **Systematic evaluation.** We compare recurrent, transformer, and structured-memory agents, identifying when rewriting mechanisms succeed or fail and how performance degrades with increasing memory update frequency.
3. **Design principles.** We analyze architectural factors linked to rewriting competence, highlighting the effectiveness of *explicit, adaptive forgetting* (e.g., learnable forget gates) over cached-state or rigidly structured memories, and outline practical guidelines for evaluating memory mechanisms in partially observable tasks.

2 Background

Markov Decision Processes. A fully observable decision-making problem can be formalized as a *Markov Decision Process* (MDP), $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \mu_0, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, $P(\cdot \mid s, a)$ is the transition kernel, $r(s, a)$ the reward, μ_0 the initial-state distribution, and γ the discount factor (Sutton & Barto, 1998). The Markov property implies that the optimal policy depends only on the current state, $\pi^\star(a_t \mid s_t)$, with the objective of maximizing the expected discounted return $\mathbb{E}[\sum_t \gamma^t r(s_t, a_t)]$.

Partially Observable Markov Decision Processes. In many real-world settings, the state x_t is not directly observable. A *Partially Observable MDP* (POMDP) extends the MDP by introducing an observation space $\mathcal{O}$ and observation kernel $O(o \mid s', a)$, defining the tuple $\mathcal{P} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, P, O, r, \mu_0, \gamma)$ (Kaelbling et al., 1998). Here, policies condition on the full interaction history $h_t = (o_0, a_0, \ldots, o_t)$, resulting in $\pi(a_t \mid h_t)$. Because exact Bayesian filtering (Ahmadi et al., 2020) is typically intractable, agents approximate it using a learned memory state $m_t = f_\phi(h_t)$,

which summarizes past information for decision-making via $\pi_\theta(a_t \mid m_t)$. In the fully observable limit, m_t coincides with the true state, recovering the standard MDP.

Dynamic Memory and Rewriting. Under partial observability, agents must not only retain but also *rewrite* memory as task-relevant information changes. Let $m_t \in \mathcal{M}$ denote the memory state and $\eta_t = (a_t, o_{t+1})$ the new input. A general update rule is $m_{t+1} = U_\phi(m_t, \eta_t)$, which can be decomposed as $m_{t+1} = W_\phi\big(F_\phi(m_t),\ E_\phi(\eta_t)\big)$, where F_ϕ selects or forgets parts of the old memory, E_ϕ encodes new input, and W_ϕ integrates both. Rewriting thus represents a selective update process – information is preserved, attenuated, or replaced depending on current relevance – ensuring that the memory state evolves to maintain only the most decision-relevant features of experience.

Memory-intensive environments and dynamic correlations. A partially observable environment can be described by a set of *correlation horizons* $\Xi = \{\xi_n\}$, where each $\xi = t_r - t_e - \Delta t + 1$ measures the temporal gap between an informative event $\alpha_{t_e}^{\Delta t}$, which begins at time t_e and lasts for Δt steps, and the later decision β_{t_r} made at time t_r that depends on it (Cherepanov et al., 2026b). An environment is *memory-intensive* when $\min \Xi > 1$, requiring the agent to recall information across multiple timesteps. In our tasks, the events $\alpha_{t_e}^{\Delta t}$ are *dynamic* and evolve over time ($\alpha_{t_e}^{\Delta t} = \alpha_{t_e}^{\Delta t}(t)$), producing shifting and overlapping correlation horizons. This dynamic structure forces the agent to not only retain long dependencies but also to rewrite outdated information as conditions change – an ability explicitly probed by our `Endless T-Maze` and `Color-Cubes` environments.

3 Related Work

3.1 Memory-augmented RL agents.

Partial observability is a defining property of many real-world decision-making problems, where agents must infer hidden state information from incomplete and noisy observations. To act effectively, such agents require mechanisms that can retain, integrate, and adapt information over time. This need has motivated a broad line of research on *memory-augmented RL* architectures, which differ in how they represent, store, and update temporal dependencies.

Recurrent architectures. A common approach introduces recurrent neural networks into policy and value functions. The **PPO-LSTM** (Schulman et al., 2017) extends Proximal Policy Optimization (PPO, (Schulman et al., 2017)) with Long Short-Term Memory (LSTM, (Hochreiter & Schmidhuber, 1997)) units, allowing agents to maintain internal states that summarize recent experience. The gating mechanisms of the LSTM regulate what information is preserved or forgotten at each step, providing a learnable and adaptive form of memory that serves as a strong baseline for studying sequential decision-making in RL.

Transformer-based architectures. Transformers have recently been adapted to RL to leverage their ability to model long-range dependencies. The **Gated Transformer-XL** (**GTrXL**) (Parisotto et al., 2020) extends the Transformer-XL architecture (TrXL) (Dai et al., 2019) for RL by introducing identity-map reordering and gating mechanisms that improve training stability. It preserves long-range temporal context through cached hidden states, enabling policies to leverage information beyond the immediate observation window. However, transformers lack an explicit mechanism for selective forgetting, which limits their stability in long-horizon or sparse-reward environments.

Structured memory architectures. To achieve more robust and interpretable forms of temporal abstraction, recent work has introduced structured external memory systems. The **Fast and Forgetful Memory** (**FFM**, (Morad et al., 2023b)) models memory as a set of exponentially decaying traces, enabling gradual forgetting of outdated information inspired by computational psychology. The **Stable Hadamard Memory** (**SHM**, (Le et al., 2024)) extends this idea by learning a dynamic calibration matrix that adaptively regulates which components of memory are reinforced or suppressed, enhancing stability during online updates.

Across these architectures – recurrent, transformer-based, and structured – most research has focused on improving memory *retention* and stability. In contrast, the equally important ability to

rewrite memory – selectively discarding obsolete content and incorporating new, task-relevant information – remains largely unexplored. Our work isolates this capability and provides a systematic framework for evaluating how existing memory mechanisms handle continual memory updating in partially observable settings.

3.2 Existing benchmarks for memory testing

A wide range of benchmarks has been developed to evaluate how RL agents store, retain, and use information under partial observability. These environments differ in sensory modality, task abstraction, and the specific aspects of memory they test.

3D embodied environments. First-person view benchmarks such as DeepMind Lab (Beattie et al., 2016) and MiniWorld (Chevalier-Boisvert et al., 2023) test navigation and spatial reasoning under occlusion, though agents often rely on perceptual shortcuts instead of true long-term memory. Memory Maze (Pasukonis et al., 2022) enforces integration of temporally separated cues for localization, while ViZDoom-Two-Colors (Sorokin et al., 2022) presents 3D navigation tasks where agents must remember previously observed color cues to make correct decisions later in the episode.

2D grid-world and diagnostic tasks. Grid- and vector-based environments provide controlled settings for studying temporal dependencies under partial observability. MiniGrid (Chevalier-Boisvert et al., 2023) and Memory Gym (Pleines et al., 2025) test sequence recall and delayed reasoning, while POPGym (Morad et al., 2023a) and POPGym Arcade (Wang et al., 2025b) cover both vector and visual modalities, emphasizing puzzle-like memory and POMDP control tasks. Synthetic POMDPs (Wang et al., 2025a) extend this idea by procedurally generating tasks with tunable observability and horizon length. The T-Maze (Ni et al., 2023) isolates the agent's ability to recall an early cue later, and Endless Memory Gym (Pleines et al., 2025) introduces adaptive, unbounded variants that evaluate continual memory updating over extended horizons.

Cognitive and multi-modal benchmarks. Beyond navigation and sequence recall, several benchmarks target higher-level cognitive abilities. DeepMind's Memory Tasks Suite (Lampinen et al., 2021), Animal-AI (Voudouris et al., 2025), and BabyAI (Chevalier-Boisvert et al., 2018) test reasoning, instruction following, and object permanence. The POBAX benchmark (Tao et al., 2025) evaluates memory in partially observable settings through *memory improvability* – the benefit of adding memory to agents – across diverse tasks that require recalling information over time.

Memory benchmarks in robotics. Memory has also emerged as a key challenge in robotic control, where agents must reason over continuous, partially observable dynamics. Benchmarks such as MIKASA-Robo (Cherepanov et al., 2026a) and MemoryBench (Fang et al., 2025) introduce visuomotor tasks that require recalling and updating latent scene properties across multiple interactions, emphasizing the role of memory in embodied decision-making.

Existing benchmarks mainly test memory *retention* – preserving information over time – but rarely the ability to *rewrite* it when cues become outdated. Our Endless T-Maze and Color-Cubes benchmarks fill this gap by requiring agents to continually discard obsolete cues and integrate new evidence under partial observability. Unlike Endless Memory Gym (Pleines et al., 2025), which evaluates how long agents can preserve growing sequences of information in unbounded tasks, our environments focus on the opposite challenge – when and how agents should overwrite prior beliefs as task conditions change. This shift from testing memory *capacity* to testing memory *adaptivity* provides a complementary perspective on continual learning and exposes limitations in current architectures' ability to update internal representations. Figure 2 contrasts the classic retention-focused T-Maze with our continual rewriting variant.

4 Benchmarking Memory Rewrite

Evaluating memory in RL has traditionally focused on how well agents retain past information. However, many real-world problems demand not just stable retention but also the ability to update

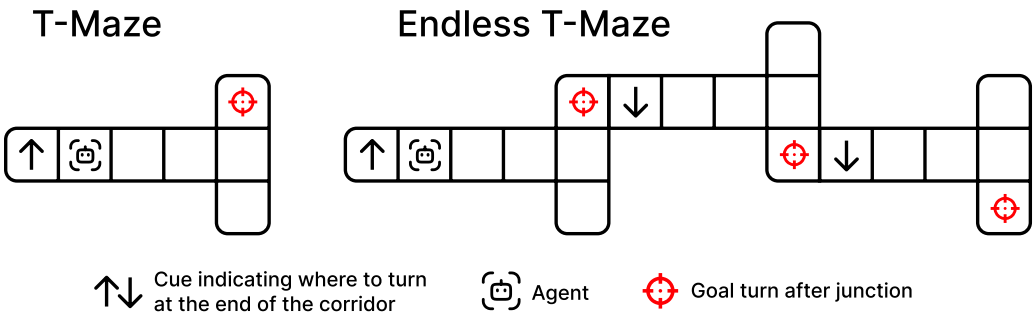

Figure 2: `T-Maze` versus `Endless T-Maze`. In the classic `T-Maze` (left), the agent receives a cue at the start and must remember it until the junction. Our proposed `Endless T-Maze` (right) extends this setup by chaining junctions: at each corridor, a fresh cue overrides all previous ones, and continual rewriting rather than simple retention is necessary for success.

internal representations when previously relevant cues become obsolete. To isolate and measure this capability, we introduce a set of controlled diagnostic environments specifically designed to stress continual memory rewriting under partial observability.

Our framework comprises two families of tasks, **`Endless T-Maze`** and **`Color-Cubes`**, instantiated at three difficulty levels `Trivial`, `Medium`, and `Extreme`. All environments require agents to detect when internal information has become outdated and must be replaced, shifting the evaluation focus from long-term memory retention to adaptive memory manipulation. **`Endless T-Maze`** constitutes a minimal diagnostic that isolates the ability to overwrite old information with new one. At every corridor, the agent receives a fresh left/right cue that completely overrides the previous one, and only the most recent cue is relevant for the subsequent junction. In contrast, **`Color-Cubes`** serves as a full-fledged stress test for selective and context-dependent forgetting. The agent navigates a grid world to collect cubes of a target color, while non-target cubes teleport over time. State updates are revealed only after specific events, and information that was previously irrelevant can later become useful again. As a result, the agent must determine what to discard, what to retain, and what to reinstate, thereby demanding not only memory rewriting but selective maintenance under uncertainty.

We evaluate a diverse range of memory-augmented RL architectures – including recurrent models, transformer-based agents, and structured external memory systems – enabling systematic comparison of the strategies by which different mechanisms support continual memory updating. Taken together, these environments provide a principled platform for investigating when and how agents rewrite memory, a capacity that is essential for adaptive behavior in non-stationary and partially observable domains.

Endless T-Maze. This environment extends the T-Maze (Ni et al., 2023) into an infinite sequence of interconnected corridors, forming a continual version of the cue-based navigation task. At the start of each corridor, the agent receives a binary cue indicating whether to turn left or right at the upcoming junction. Once a turn is made, the cue changes, invalidating all previous information – thereby requiring continual memory overwriting rather than static retention.

The observation space is a compact vector encoding the current cue and the agent's normalized position within the corridor, while the action space includes `MOVE_FORWARD`, `TURN_LEFT`, and `TURN_RIGHT`. Rewards are assigned as $+1$ for a correct turn, -1 for an incorrect turn, and -0.01 if the agent does not move forward to the junction. Although the task is theoretically unbounded, in experiments we fix either the number of corridors or their maximum length to control task difficulty and ensure stable training. We also evaluate two sampling regimes for corridor lengths: (i) **Fixed**, where all corridors share the same length $l_{\max}$, and (ii) **Uniform**, where lengths are sampled from a uniform distribution $l \sim \mathcal{U}[1, l_{\max}]$, introducing stochastic variation in cue timing and memory horizon. Further implementation details are provided in Appendix A.1.

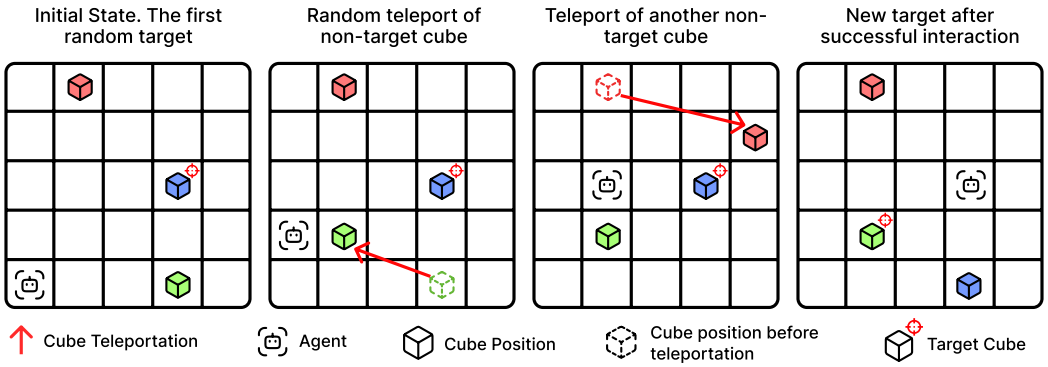

Figure 3: Visualization of key principles of the `Color-Cubes` environment, including initialization, stochastic teleportation, and successful interaction.

Color-Cubes. `Color-Cubes` (Figure 3) is a grid-based environment that evaluates an agent's ability to maintain, detect, and update internal representations under partial observability.

The agent acts on a $G \times G$ grid containing N uniquely colored cubes, each assigned a color identifier $c \in \{0, 1, \ldots, N-1\}$. Each episode consists of K consecutive sub-episodes. At the beginning of each sub-episode, the agent is assigned a new target color c_{target}, indicating which cube it must locate and interact with. During this initialization step, the agent observes the color of every cube and the target color c_{target}, along with its own (x, y) position and the (x, y) coordinates of all cubes. This color information is shown only at the start of the sub-episode or when the cube is accidentally teleported while the agent is moving toward the target, forcing the agent to rely on its memory of the target color and the initial color-position mapping to act correctly. To maintain partial visibility and not give the agent too many clues, there is a teleportation probability parameter p_{teleport}, which should not be set too high.

After the start of each sub-episode, the agent must navigate to the cube with color c_{target} and execute the `INTERACT` action. During the agent's movement, any cube other than the current target cube may perform a stochastic teleportation with a certain probability p_{teleport}, forcing the agent to detect inconsistencies between its memory and the current environment and update its internal map accordingly. Following each successful interaction, the target color changes, and the lifted cube is teleported to a new location. The agent receives a reward of $+1$ for a successful interaction and a small negative reward when moving away from the target cube. An episode terminates after successfully completing a number of sub-episodes or when the time expires.

We define three difficulty levels that progressively increase the demands on memory rewriting:

- **`Trivial`** – Single cube and single target ($N{=}K{=}1$). The agent only needs to memorize the color of one cube and perform one interaction; no rewriting is required.
- **`Medium`** – Multiple cubes with complete state updates (positions and colors). During the agent's movement, several cubes may teleport, and the agent must update its memory to track the new configuration.
- **`Extreme`** – Multiple cubes with *incomplete* teleportation updates (positions only, colors hidden). The agent must infer which cube moved based on prior knowledge, update its internal color–position mapping, and act accordingly, testing both rewriting and inference under uncertainty.

Further details on the environment configuration and dynamics are provided in Appendix A.2.

5 Experiments

We empirically evaluate how different memory architectures cope with continual memory rewriting under partial observability. We organize our investigation around four key research questions (RQs) that collectively probe different aspects of memory rewriting:

- **RQ1 (Retention).** How do different memory mechanisms behave in trivial settings where rewriting is unnecessary (e.g., n=1 `Endless T-Maze` and `Trivial Color-Cubes`)?
- **RQ2 (Rewriting).** Which mechanisms can reliably rewrite (i.e., discard obsolete cues and integrate new ones) when tasks explicitly require continual updates?
- **RQ3 (Generalization).** How well do these mechanisms interpolate and extrapolate across horizons and rewrite frequencies in the `Endless T-Maze` environment (varying corridor lengths l and counts n; fixed vs. uniform regimes)?
- **RQ4 (Prioritization).** When multiple updates accumulate, which mechanisms can re-rank and maintain a consistent, up-to-date memory (e.g., `Color-Cubes Medium`/`Extreme`) rather than merely forgetting?

5.1 Baselines

To identify which architectural principles support effective memory rewriting, we evaluate a broad set of agents covering both established and state-of-the-art (SOTA) memory mechanisms. Our study includes recurrent (**PPO-LSTM and PPO-GRU** (Pleines et al., 2022)), transformer-based (**GTrXL** (Parisotto et al., 2020)), and structured-memory architectures (**FFM** (Morad et al., 2023b), **SHM** (Le et al., 2024)). In addition, we use a memory-free **PPO-MLP** baseline to isolate the effect of explicit memory mechanisms under partial observability. Together, these agents represent the principal design paradigms for sequential decision-making in RL – recurrent dynamics, attention-based context, and structured long-term memory – and enable a systematic analysis of how each handles continual memory updating.

5.2 Training and Evaluation Protocol

We adopt a unified training and evaluation protocol across all experiments to ensure consistent comparability across agents and environments.

Training configurations. For `Endless T-Maze`, agents are trained across multiple combinations of corridor length l, number of corridors n, and corridor-length sampling regime (**Fixed** vs. **Uniform**); all configurations are summarized in Table 1. For `Color-Cubes`, we train agents on the three predefined difficulty modes – `Trivial`, `Medium`, and `Extreme` – with detailed parameters provided in Appendix A.2.

Evaluation setup. For `Endless T-Maze`, each checkpoint is evaluated on the validation configurations listed in Table 1. We include both *matched* settings (same l, n, and sampling regime as training) and *cross-setting* evaluations to probe generalization through *interpolation* (shorter corridors or fewer rewrite operations than in training) and *extrapolation* (longer corridors or more rewrites). For `Color-Cubes`, agents trained on each difficulty mode are evaluated within that mode, allowing us to measure how well memory mechanisms handle increasing demands on continual updating and partial observability.

Metrics and aggregation. Performance is reported as the *success rate*, defined as the fraction of correctly completed trials over 100 independent evaluation episodes per checkpoint. For each training run, we first compute the mean success rate across these 100 episodes. We then report the overall mean and standard error of the mean (mean $\pm$ SEM) across the ten independent runs. This aggregation captures both episode-level variability and the stability of learning across random seeds.

Reproducibility and configuration control. All environment parameters used for training and validation – including l, n, and the sampling regime for `Endless T-Maze`, as well as grid size, number of cubes, and teleportation probability for `Color-Cubes` – are detailed in Appendix A. Each agent was tuned individually for each environment, after which its hyperparameters were kept fixed across all training and validation configurations of that environment to ensure consistency within comparisons. The full list of hyperparameters is provided in Appendix B: SHM in Table 3, FFM in Table 4, MLP in Table 5, GTrXL in Table 6, and PPO-LSTM in Table 7.

Table 1: Performance of all evaluated agents on the `Endless T-Maze`. Each configuration is defined by the corridor length l and the number of corridors n under two regimes: **Fixed** (constant length) and **Uniform** (randomized length). Top-1 and top-2 results are highlighted. Here, n=1 corresponds to the classic T-Maze (pure retention), while larger n values require memory rewriting.

Regime	l	n	PPO-LSTM	GTrXL	SHM	FFM	PPO-MLP
Fixed	5	1	1.00±0.00	0.50±0.19	1.00±0.00	1.00±0.00	0.50±0.00
	5	3	1.00±0.00	0.17±0.06	0.67±0.24	1.00±0.00	0.24±0.05
	5	5	1.00±0.00	0.04±0.02	0.84±0.13	1.00±0.00	0.05±0.02
	5	10	1.00±0.00	0.61±0.17	0.23±0.22	1.00±0.00	0.00±0.00
	10	1	1.00±0.00	0.43±0.01	0.84±0.16	1.00±0.00	0.50±0.00
	10	3	1.00±0.00	0.17±0.01	0.93±0.07	1.00±0.00	0.16±0.02
	10	5	0.69±0.31	0.19±0.12	0.52±0.24	1.00±0.00	0.04±0.00
	10	10	0.83±0.17	0.02±0.02	0.02±0.01	0.73±0.27	0.00±0.00
Uniform	5	1	1.00±0.00	0.47±0.08	0.55±0.23	0.92±0.08	0.26±0.02
	5	3	1.00±0.00	0.15±0.09	0.04±0.01	0.43±0.23	0.01±0.01
	5	5	1.00±0.00	0.04±0.01	0.04±0.02	0.03±0.03	0.00±0.00
	5	10	1.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00
	10	1	1.00±0.00	0.57±0.02	0.91±0.09	1.00±0.00	0.24±0.12
	10	3	0.98±0.02	0.16±0.03	0.06±0.04	0.00±0.00	0.00±0.00
	10	5	1.00±0.00	0.06±0.01	0.04±0.01	0.00±0.00	0.00±0.00
	10	10	1.00±0.00	0.01±0.01	0.00±0.00	0.00±0.00	0.00±0.00

6 Results

In order to group meaningful results across different environments and parameters, we propose to consider four research questions.

RQ1: How different memory mechanisms deal with the trivial cases, where rewriting is not necessary? In the simplest `Endless T-Maze` task, where n=1 and the environment becomes the classic T-Maze, the agent only needs to write the cue once and retain it through the episode; we use this setting to evaluate the baseline performance of the considered models. The results are presented in Table 1 (rows where n=1).

The PPO-LSTM, SHM, and FFM agents demonstrated best performance, achieving a perfect success rate (1.00 ± 0.00) in all scenarios. In contrast, the GTrXL agent's success rates hovering around 50% like MLP's, indicating it could not reliably solve the task. In the `Trivial Color-Cubes` case represented in the first row in Table 2, the situation is slightly different. PPO-LSTM showed a result of 0.52 ± 0.10, while FFM, GTrXL, and SHM showed perfect success.

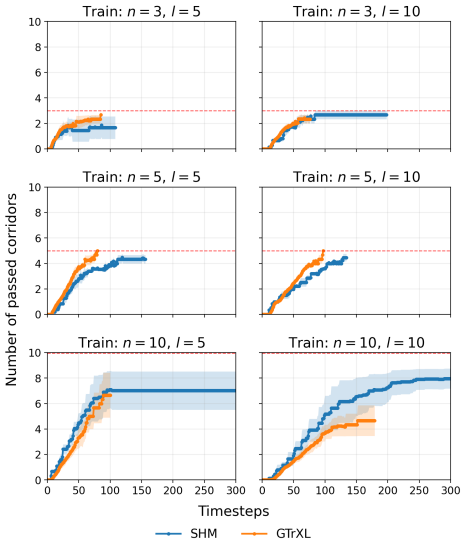

Figure 4: Intermediate progress of SHM and GTrXL in `Endless T-Maze`. Dashed red lines mark the target number of corridors.

Based on the results, we can assume that the memory mechanisms of the PPO-LSTM, SHM, and FFM architectures are capable of retaining the necessary information within the scope of our experimental protocols. A possible reason for the low success rate of GTrXL in conditions where the context length exceeds the episode length on `Endless T-Maze` is its instability in sparse reward

Table 2: Results on all `Color-Cubes` tasks. Medium and Extreme mode core parameters are: cubes = 3, sub-episodes = 3, grid size = 5, teleport chance = 0.3.

	PPO-LSTM	GTrXL	SHM	FFM	MLP
Trivial	0.52±0.10	1.00±0.00	1.00±0.00	1.00±0.00	0.00±0.00
Medium	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00
Extreme	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00

conditions, which is confirmed by its high success rate on `Color-Cubes`, where the agent regularly receives penalties or rewards. The stability challenges of Transformer-based RL agents under sparse rewards have also been noted in prior work (Chen et al., 2024).

RQ2: What memory mechanisms cope with rewriting? To try to identify key features of memory mechanisms that aid in rewriting we consider `Endless T-Maze` validation tasks in which the length of the corridor, the number of corridors, and the sampling were identical to the training configuration, and the length of the corridor was greater than 1 for `Endless T-Maze` in Table 1.

The PPO-LSTM agent demonstrated complete success in most `Endless T-Maze` tasks considered. Meanwhile, FFM was only able to successfully cope with tasks in fixed sampling mode with 1.00 ± 0.00 result. SHM also demonstrated success in fixed tasks, but within 5 corridors (n=5) and less than FFM in configurations with n=5 (0.84 ± 0.13 in l=5 and 0.52 ± 0.24 in l=10).

GTrXL and MLP were unable to produce meaningful results in tasks that required memory rewriting.

To better understand how GTrXL and SHM behave in the `Endless T-Maze`, we examine how many corridors each agent can successfully traverse within an episode despite their overall low success rates (Figure 4). This analysis reveals how long these architectures can maintain accurate cue following before their memory representations deteriorate. In tasks with n=3 and n=5, both agents typically manage to navigate one corridor fewer than required, as indicated by the interrupted curves showing partial completion of the maze.

These results reveal that the baselines' ability to handle memory rewriting is highly dependent on environmental predictability. The mechanism in PPO-LSTM is robust enough to succeed in both predictable (**Fixed**) and stochastic (**Uniform**) settings. Conversely, the mechanisms in FFM and SHM, while effective for static patterns, are not sufficiently flexible to adapt to the variability of the uniform mode. This suggests their rewriting capability is effective but limited to predictable scenarios. Also, SHM results regarding the number of steps required to complete episodes may be indicative of brute force, but stable successful execution of $n-1$ corridors in some tasks may refute this.

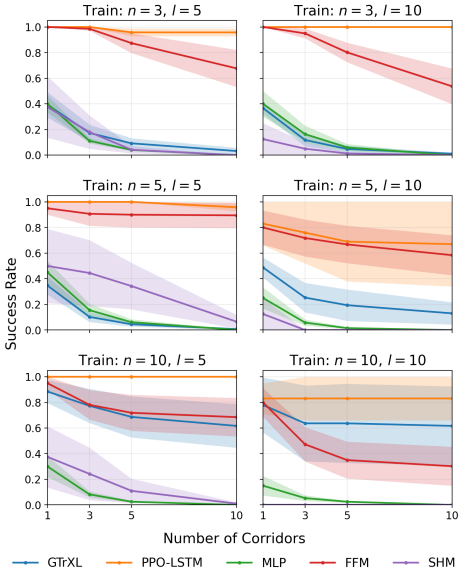

Figure 5: Baseline comparison under interpolation and extrapolation in `Endless T-Maze`. Results report success rate as mean±SEM.

RQ3: What memory mechanisms can demonstrate generalization in memory rewriting? When considering this research question, we consider the terms "interpolation" and "extrapolation" in context of `Endless T-Maze` results. Interpolation means that the agent was able to cope with the validation task where the number of corridors or their length was less than the training configuration, while Extrapolation means the opposite result – the number of corridors or their length in the validation task was greater.

Figure 5 shows that PPO-LSTM demonstrated interpolation for all lengths in non-trivial training configurations with uniform mode, achieving complete success regardless of the number of corridors trained and validated. In addition, PPO-LSTM was able to demonstrate incomplete but successful extrapolation to a corridor length of 10, again regardless of the number of corridors with a non-trivial uniform 5 training task.

FFM, SHM, and GTrXL performed successfully in non-trivial fixed training configurations, the models show generalization to other numbers of corridors. The FFM agent demonstrated both extrapolation and interpolation across all such configurations, while SHM and GTrXL demonstrated only interpolation, and the transformer architecture performs better.

It is worth considering how the attention mechanism performed under uniform conditions using the example of the GTrXL baseline results. Agent managed to interpolate one uniform training task to fixed configuration task by the number of corridors. In particular, the training configuration n=3, l=5, and regime=Uniform showed a slight extrapolation to the task with 5 corridors with success 0.38 ± 0.21. The generalization results allow us to establish a clear hierarchy of flexibility among the memory mechanisms.

At the top is PPO-LSTM, which learns an abstract and adaptable strategy, enabling it to generalize successfully even in uniform environments. Below it lies FFM, which demonstrates a more limited and conditional form of flexibility. Its ability to both interpolate and extrapolate, but exclusively within the predictable confines of the Fixed mode, suggests it can generalize a learned pattern, but cannot adapt the pattern itself. At the bottom of this hierarchy is SHM, which proves to be the most rigid. Its ability is restricted to interpolation only, indicating that its learned strategy is tightly coupled with the specific parameters of the training environment and cannot be scaled even in predictable settings.

This distinct ranking of adaptability provides the piece of evidence, suggesting that the key to success lies in a fundamental property of the memory mechanism that dictates how it acquires and applies knowledge in new situations.

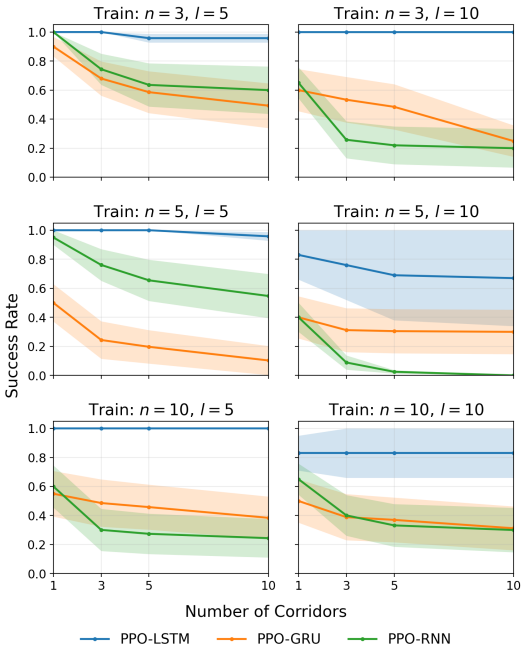

Figure 6: Recurrent-model comparison on `Endless T-Maze` under interpolation and extrapolation conditions. Corridor lengths and sampling regimes are the same during training and validation, while the number of corridors varies according to the experiment protocol. Results report success rate as mean±SEM.

RQ4: Which memory mechanisms are capable of efficiently re-ranking rewritten information? After the first sub-episode, a new random target is chosen in `Medium` and `Extreme` modes. To successfully complete the second and subsequent sub-episodes, the agent must remember and record all teleportations that have occurred, and in `Extreme` mode, it must also make logical conclusions in order to correctly recognize which colors correspond to which coordinates. In other words, we also test the ability not only to adaptively overwrite and forget old information, but also dynamically update the storage, reevaluating the relevance of all stored data in order to be able to focus on the current task.

Under these conditions, all baselines showed zero success (see Table 2), which may indicate that their memory mechanisms are not sensitive to tasks where simply forgetting does not help.

7 Ablation study

Since the PPO-LSTM model outperformed other baselines under considered scenarios, a hypothesis was put forward that its effectiveness may be due to the specifics of its internal architecture, namely the presence of gates that separate and control different aspects of calculations.

To test this hypothesis, a series of comparative experiments was conducted, including modifications of PPO with alternative recurrent architectures – PPO-RNN and PPO-GRU. The PPO-RNN model, based on a classic recurrent layer without gating mechanisms, will help in analyzing the general role of gates in memory rewriting tasks. Also PPO-GRU is an intermediate option in which the architecture is simplified compared to LSTM: the cell and hidden states are combined, and the number of gates is reduced from three (forget, input, output) to two (reset and update). This comparison allows us to evaluate not only the usefulness of gating as such, but also the impact of specific memory and information flow control mechanisms on overall performance.

Figure 6 summarizes the recurrent-model ablation comparison.

For performance comparison, we used scenarios from experiments in the Endless T-Maze environment in fixed mode, with corridor lengths varying between 5 and 10, and the number of corridors varying between 3, 5, and 10. These scenarios were chosen because their extrapolation and interpolation conditions enabled a comparative analysis of the previously selected baselines' performance.

Figure 6 shows that RNN succeeds only in a limited subset of interpolation and extrapolation settings precisely those where SHM and FFM (also without gates) previously achieved partial success. At the same time, PPO-GRU demonstrated the ability to interpolate and extrapolate not only when changing the number of corridors in scenarios where SHM and FFM showed results, but also demonstrated a limited ability to generalize when changing the corridor length. However, PPO-GRU was inferior to PPO-LSTM in terms of generalization success over lengths.

A comparison of PPO-RNN as a representative of recurrent models without gating with PPO-GRU and PPO-LSTM, which showed significantly higher results, indicates that gating mechanisms have a key influence on the success of rewriting tasks. The higher performance of PPO-GRU over SHM, FFM, and PPO-RNN in its ability to generalize the solution to length variation further confirms the usefulness of gates in such tasks. At the same time, the advantage of PPO-LSTM over PPO-GRU in length generalization scenarios demonstrates the importance of an adaptive forgetting mechanism.

8 Discussion

The results from the preceding research questions establish a consistent performance hierarchy among the tested architectures. Across tasks involving information retention (RQ1), rewriting (RQ2), and generalization (RQ3), the observed ranking was: PPO-LSTM, FFM, GTrXL, SHM and MLP.

This ranking correlates with the architectural approach each model takes to managing outdated information – specifically, with its mechanism for forgetting. An analysis of the architectures reveals a trend:

- **PPO-LSTM**, the top-performing model, utilizes a dedicated forget gate with learnable parameters. This allows for an adaptive, context-dependent approach to information retention and removal.
- **FFM** architecture implements a more defined, rule-based forgetting process where older information is systematically replaced. This approach was effective in predictable fixed environments but was associated with a significant performance degradation in uniform settings.
- **GTrXL** as mentioned earlier, the model is vulnerable to sparse reward environments, but at the same time it was able to show generalization in some scenarios. Nevertheless, the stabilization of architecture learning and information writing to the cache occurs with the help of gating, which once again emphasizes their necessity.

- **SHM** has a similar concept of matrix memory structuring to FFM, but lacks an explicit forgetting mechanism. SHM has a dynamic update of information significance. This design correlated with lower performance compared to FFM, particularly in generalization tasks.

Based on this observed correlation, we hypothesize that the nature and adaptability of the forgetting mechanism may be key factors determining an agent's effectiveness in performing tasks that require memory rewriting in dynamic environments.

A comparison in an ablation study with PPO-RNN confirmed that PPO-LSTM gates contributed to its success, while a comparison of its gates with PPO-GRU gates highlighted the need for an adaptive forgetting mechanism.

However, PPO-LSTM performed worse than other memory-based baselines in Trivial Color-Cubes. Since this case study does not involve rewriting, it cannot be said that SHM, FFM, and GTrXL are superior, but PPO-LSTM may not be robust to grid environments.

9 Limitations & Future Work

One of the key limitations of this study is the small number of scientific papers devoted directly to the mechanisms of memory rewriting in RL. This complicates the comparison of the results obtained. In addition, a broader range of benchmarks and a larger number of diverse agents are needed to more accurately assess and isolate individual memory abilities and mechanisms. Finally, based on the analysis of the results obtained, a promising direction for further work is the development of a proprietary mechanism that specifically addresses the task of memory rewriting.

10 Conclusion

Our study reveals that memory retention alone is insufficient for solving RL tasks that require continual adaptation under partial observability. Through the proposed `Endless T-Maze` and `Color-Cubes` benchmarks, we isolate and evaluate the ability of RL agents to rewrite memory – to selectively discard obsolete information and integrate new evidence as environments evolve. Across recurrent, transformer-based, and structured-memory architectures, we observe a consistent hierarchy of adaptive competence: agents with explicit, learnable forgetting mechanisms (such as LSTM-based policies) exhibit strong performance and generalization, whereas attention- and cache-based models struggle when rewriting is essential rather than optional. These findings suggest that the core challenge in memory-intensive decision-making is not the preservation of information but its controlled transformation over time. Future memory systems should therefore move beyond static representations of history toward architectures that model memory as an active, continuously updated belief state – capable of forgetting, rewriting, and reallocating representational capacity as conditions change. We view this as a step toward closing the gap between artificial and biological memory, where adaptation and forgetting are not failure modes but fundamental features of intelligent behavior.

References

Mohamadreza Ahmadi, Nils Jansen, Bo Wu, and Ufuk Topcu. Control theory meets pomdps: A hybrid systems approach. *IEEE Transactions on Automatic Control*, 66(11):5191–5204, 2020.

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 139–154, 2018.

Charles Beattie, Joel Z. Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, Julian Schrittwieser, Keith Anderson, Sarah York, Max Cant, Adam Cain, Adrian Bolton, Stephen Gaffney, Helen King, Demis Hassabis, Shane Legg, and Stig Petersen. Deepmind lab, 2016. URL https://arxiv.org/abs/1612.03801.

Zachary H Bretton, Hyojeong Kim, Marie T Banich, and Jarrod A Lewis-Peacock. Suppressing the maintenance of information in working memory alters long-term memory traces. *Journal of Cognitive Neuroscience*, 36(10):2117–2136, 2024.

Chang Chen, Yi-Fu Wu, Jaesik Yoon, and Sungjin Ahn. Transdreamer: Reinforcement learning with transformer world models, 2024. URL https://arxiv.org/abs/2202.09481.

Egor Cherepanov, Nikita Kachaev, Alexey Kovalev, and Aleksandr Panov. Memory, benchmark & robots: A benchmark for solving complex tasks with reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=9cLPurIZMj.

Egor Cherepanov, Nikita Kachaev, Artem Zholus, Alexey Kovalev, and Aleksandr Panov. Unraveling the complexity of memory in RL agents: an approach for classification and evaluation. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=lJKdOYFF5W.

Egor Cherepanov, Alexey Kovalev, and Aleksandr Panov. ELMUR: External layer memory with update/rewrite for long-horizon RL. In *The Fourteenth International Conference on Learning Representations*, 2026c. URL https://openreview.net/forum?id=bm3rbtEMFj.

Egor Cherepanov, Aleksei Staroverov, Alexey Kovalev, and Aleksandr Panov. Recurrent action transformer with memory. In *The Fourteenth International Conference on Learning Representations*, 2026d. URL https://openreview.net/forum?id=kByN4v0M3e.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. *arXiv preprint arXiv:1810.08272*, 2018.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo de Lazcano, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks, 2023. URL https://arxiv.org/abs/2306.13831.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context, 2019. URL https://arxiv.org/abs/1901.02860.

Kevin Esslinger, Robert Platt, and Christopher Amato. Deep transformer q-networks for partially observable reinforcement learning. *arXiv preprint arXiv:2206.01078*, 2022.

Haoquan Fang, Markus Grotz, Wilbert Pumacay, Yi Ru Wang, Dieter Fox, Ranjay Krishna, and Jiafei Duan. Sam2act: Integrating visual foundation model with a memory architecture for robotic manipulation. *arXiv preprint arXiv:2501.18564*, 2025.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997. URL https://api.semanticscholar.org/CorpusID:1915014.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

Peter Koepernik, Ruo Yu Tao, Ronald Parr, George Konidaris, and Cameron Allen. General value discrepancies mitigate partial observability in reinforcement learning. In *Finding the Frame Workshop at RLC 2025*, 2025. URL https://openreview.net/forum?id=IrpcqYhREq.

Seth R Koslov, Arjun Mukerji, Katlyn R Hedgpeth, and Jarrod A Lewis-Peacock. Cognitive flexibility improves memory for delayed intentions. *ENeuro*, 6(6), 2019.

Hanna Kurniawati. Partially observable markov decision processes and robotics. *Annu. Rev. Control. Robotics Auton. Syst.*, 5:253–277, 2022. URL https://api.semanticscholar.org/CorpusID:245789500.

Andrew Lampinen, Stephanie Chan, Andrea Banino, and Felix Hill. Towards mental time travel: a hierarchical memory for reinforcement learning agents. *Advances in Neural Information Processing Systems*, 34:28182–28195, 2021.

Mikko Lauri, David Hsu, and Joni Pajarinen. Partially observable markov decision processes in robotics: A survey. *IEEE Transactions on Robotics*, 39(1):21–40, 2022.

Hung Le, Kien Do, Dung Nguyen, Sunil Gupta, and Svetha Venkatesh. Stable hadamard memory: Revitalizing memory-augmented agents for reinforcement learning, 2024. URL https://arxiv.org/abs/2410.10132.

Lingheng Meng, Rob Gorbet, and Dana Kulić. Memory-based deep reinforcement learning for pomdps. In *2021 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 5619–5626. IEEE, 2021.

Steven Morad, Ryan Kortvelesy, Matteo Bettini, Stephan Liwicki, and Amanda Prorok. Popgym: Benchmarking partially observable reinforcement learning, 2023a. URL https://arxiv.org/abs/2303.01859.

Steven Morad, Ryan Kortvelesy, Stephan Liwicki, and Amanda Prorok. Reinforcement learning with fast and forgetful memory, 2023b. URL https://arxiv.org/abs/2310.04128.

Swaleha Mujawar, Jaideep Patil, Bhushan Chaudhari, and Daniel Saldanha. Memory: Neurobiological mechanisms and assessment. *Industrial psychiatry journal*, 30(Suppl 1):S311–S314, 2021.

Tianwei Ni, Michel Ma, Benjamin Eysenbach, and Pierre-Luc Bacon. When do transformers shine in rl? decoupling memory from credit assignment. *Advances in Neural Information Processing Systems*, 36:50429–50452, 2023.

Emilio Parisotto, Francis Song, Jack Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, et al. Stabilizing transformers for reinforcement learning. In *International conference on machine learning*, pp. 7487–7498. PMLR, 2020.

Jurgis Pasukonis, Timothy Lillicrap, and Danijar Hafner. Evaluating long-term memory in 3d mazes, 2022. URL https://arxiv.org/abs/2210.13383.

Marco Pleines, Matthias Pallasch, Frank Zimmer, and Mike Preuss. Generalization, mayhems and limits in recurrent proximal policy optimization, 2022. URL https://arxiv.org/abs/2205.11104.

Marco Pleines, Matthias Pallasch, Frank Zimmer, and Mike Preuss. Memory gym: Towards endless tasks to benchmark memory capabilities of agents. *Journal of Machine Learning Research*, 26 (6):1–40, 2025.

John J Sakon and Roozbeh Kiani. Differences in memory for what, where, and when components of recently formed episodes. *Journal of neurophysiology*, 128(2):310–325, 2022.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Artyom Sorokin, Nazar Buzun, Leonid Pugachev, and Mikhail Burtsev. Explain my surprise: Learning efficient long-term memory by predicting uncertain outcomes. 07 2022. DOI: 10.48550/arXiv.2207.13649.

Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Ruo Yu Tao, Kaicheng Guo, Cameron Allen, and George Konidaris. Benchmarking partial observability in reinforcement learning with a suite of memory-improvable domains. *arXiv preprint arXiv:2508.00046*, 2025.

Konstantinos Voudouris, Ibrahim Alhas, Wout Schellaert, Matteo G. Mecattaf, Ben Slater, Matthew Crosby, Joel Holmes, John Burden, Niharika Chaubey, Niall Donnelly, Matishalin Patel, Marta Halina, José Hernández-Orallo, and Lucy G. Cheke. The animal-ai environment: A virtual laboratory for comparative cognition and artificial intelligence research, 2025. URL https://arxiv.org/abs/2312.11414.

Yongyi Wang, Lingfeng Li, Bozhou Chen, Ang Li, Hanyu Liu, Qirui Zheng, Xionghui Yang, and Wenxin Li. Synthetic pomdps to challenge memory-augmented rl: Memory demand structure modeling. *arXiv preprint arXiv:2508.04282*, 2025a.

Zekang Wang, Zhe He, Borong Zhang, Edan Toledo, and Steven Morad. Popgym arcade: Parallel pixelated pomdps, 2025b. URL https://arxiv.org/abs/2503.01450.

Zhifang Ye, Liang Shi, Anqi Li, Chuansheng Chen, and Gui Xue. Retrieval practice facilitates memory updating by enhancing and differentiating medial prefrontal cortex representations. *Elife*, 9:e57023, 2020.

Daniil Zelezetsky, Egor Cherepanov, Alexey K Kovalev, and Aleksandr I Panov. Re: Frame–retrieving experience from associative memory. *arXiv preprint arXiv:2508.19344*, 2025.

Haoxu Zhang, Parham M Kebria, Shady Mohamed, Samson Yu, and Saeid Nahavandi. A review on robot manipulation methods in human-robot interactions. *arXiv preprint arXiv:2309.04687*, 2023.

Supplementary Materials

The following content was not necessarily subject to peer review.

A Detailed Environments Descriptions

A.1 Detailed Description: Endless T-Maze.

The `Endless T-Maze` environment extends the classical T-Maze (Ni et al., 2023) into a continual, multi-stage task that explicitly tests an agent's ability to rewrite memory as new cues arrive. The environment consists of a sequential chain of N T-shaped corridors, each ending in a left–right junction. At the beginning of every corridor, the agent receives a binary cue represented as a two-dimensional vector – $(1, 0)$ for "turn left" or $(0, 1)$ for "turn right". This cue is displayed only at the corridor's start and becomes invalid once the next corridor begins, forcing the agent to overwrite previously stored information rather than accumulate it.

During traversal, the agent moves forward along the corridor, observing its normalized position (a scalar between 0 and 1) and the cue vector, which becomes $(0, 0)$ after the first step. The observation space therefore consists of three values: $(\text{cue}_1, \text{cue}_2, \text{position})$. The discrete action space includes `MOVE_FORWARD`, `TURN_LEFT`, and `TURN_RIGHT`. A correct turn at a junction yields a reward of $+1$, an incorrect turn results in a penalty of -1, and each idle or forward step incurs a small penalty of -0.01 to encourage efficient movement.

Two corridor-length regimes are supported:

- **Fixed:** all corridors have the same predefined length $l_{\max}$.
- **Uniform:** corridor lengths are sampled from a uniform distribution $l \sim \mathcal{U}[1, l_{\max}]$, introducing variability in memory duration between cue presentation and decision.

Although the environment is conceptually infinite, in practice episodes are truncated after a fixed number of corridors or a global step limit to maintain stable training and evaluation. This setup directly measures an agent's capacity to discard obsolete cues and replace them with new, task-relevant information in a continually evolving sequence of decisions.

A.2 Detailed Description: Color-Cubes.

The environment consists of a two-dimensional grid of size $G \times G$ with N cubes placed at random, unique positions. Each cube is assigned a unique color $c \in \{0, 1, \ldots, N-1\}$, and the agent's goal is to sequentially locate and interact with cubes of a specified target color.

An episode is divided into K sub-episodes. At the start of each sub-episode, the agent receives the target color c_{target}. The agent must navigate to the corresponding cube and execute the `INTERACT` action. A successful interaction concludes the sub-episode, grants a reward of $+1.0$, teleports the collected cube to a new random unoccupied cell, and assigns a new target color from the remaining set of cubes. The agent also receives a new `full_state_update`. At every timestep where no successful interaction occurs, a random non-target cube teleports with probability p_{teleport}, triggering another `full_state_update`. If neither event happens, the update is withheld, forcing the agent to act using only its internal memory. Because the world state may change without notice, outdated memory can lead the agent to an empty cell where a cube previously stood.

Difficulty variants.

- **`Trivial:`** One cube and one target ($N{=}K{=}1$). The agent only needs to remember and interact with a single target cube.
- **`Medium:`** The standard setup with multiple cubes and complete state updates (positions and colors).
- **`Extreme:`** Teleportation updates omit color identifiers. The agent must recall the initial color–position mapping, identify which cube moved, and update its internal representation accordingly.

An episode terminates after K successful interactions or upon reaching the time limit. This setup explicitly tests whether agents can maintain, detect, and rewrite memory representations when the environment changes unpredictably.

B Training Configuration

Table 3: Hyperparameters for SHM.

Hyperparameter	**Value**
Hidden size (h)	512
Memory size (m)	128
Post-processing size	1024
Learning rate	5×10^{-5}
Discount (γ)	0.99
GAE lambda (λ)	1.0
Gradient clipping	0.5
Entropy coefficient	0.001
Value loss coefficient	0.5
BPTT length	1024
Train batch size	65536
Minibatch size	8192
PPO epochs	6
Number of workers	12
Total timesteps	2M
Framework	Ray RLlib

Table 4: Hyperparameters for FFM.

Hyperparameter	**Value**
Hidden size	128
Memory size (m)	128
Post-processing size	1024
Learning rate	5×10^{-5}
Discount (γ)	0.99
GAE lambda (λ)	1.0
Gradient clipping	0.5
Entropy coefficient	0.001
Value loss coefficient	0.5
BPTT length	1024
Train batch size	65536
Minibatch size	8192
PPO epochs	6
Number of workers	8
Total timesteps	2M
Framework	Ray RLlib

Table 5: Hyperparameters for MLP.

Hyperparameter	**Value**
Hidden size	256
Memory	None
Learning rate	5×10^{-5}
Discount (γ)	0.99
GAE lambda (λ)	1.0
Gradient clipping	0.5
Entropy coefficient	0.001
Value loss coefficient	0.5
BPTT length	1024
Train batch size	65536
Minibatch size	8192
PPO epochs	6
Number of workers	2
Total timesteps	2M
Framework	Ray RLlib

Table 6: Hyperparameters for GTrXL.

Hyperparameter	**Value**
Hidden size	512
Memory length	100
Transformer blocks	4
Attention heads	8
Embedding dimension	512
Positional encoding	Relative
GTrXL gating	Enabled
Learning rate (initial)	2.75×10^{-4}
Learning rate (final)	1×10^{-5}
Discount (γ)	0.995
GAE lambda (λ)	0.95
Gradient clipping	0.25
Entropy coeff (initial)	0.001
Entropy coeff (final)	1×10^{-6}
Value loss coefficient	0.5
PPO epochs	3
Number of workers	32
Steps per worker	512
Minibatches per epoch	8
Total timesteps	2M
Framework	Custom PPO

Table 7: Hyperparameters for PPO-LSTM.

Hyperparameter	**Value**
LSTM hidden size	128
Actor network	[128, 128]
Critic network	[128, 128]
Learning rate (initial)	3×10^{-4}
Learning rate (final)	1×10^{-5}
LR schedule	Linear
Discount (γ)	0.99
GAE lambda (λ)	0.98
Gradient clipping	0.5
Entropy coefficient	0.1
Value loss coefficient	0.5
Sequence length	128
Minibatch size	512
PPO epochs	10
Number of environments	16
Steps per environment	128
Total timesteps	2M
Framework	Stable-Baselines3