Unraveling the Complexity of Memory in RL Agents: an Approach for Classification and Evaluation

Anonymous Author(s)
Affiliation
Address
email

Abstract: The incorporation of memory into agents is essential for numerous tasks within the domain of Reinforcement Learning (RL). In particular, memory is paramount for tasks that require the use of past information, adaptation to novel environments, and improved sample efficiency. However, the term "memory" encompasses a wide range of concepts, which, coupled with the lack of a unified methodology for validating an agent's memory, leads to erroneous judgments about agents' memory capabilities and prevents objective comparison with other memory-enhanced agents. This paper aims to streamline the concept of memory in RL by providing practical precise definitions of agent memory types, such as long-term vs. short-term memory and declarative vs. procedural memory, inspired by cognitive science. Using these definitions, we categorize different classes of agent memory, propose a robust experimental methodology for evaluating the memory capabilities of RL agents, and standardize evaluations. Furthermore, we empirically demonstrate the importance of adhering to the proposed methodology when evaluating different types of agent memory by conducting experiments with different RL agents and what its violation leads to.

Keywords: Memory, POMDP, RL

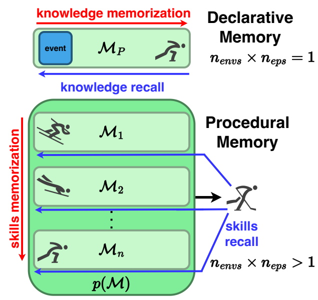
1 Introduction

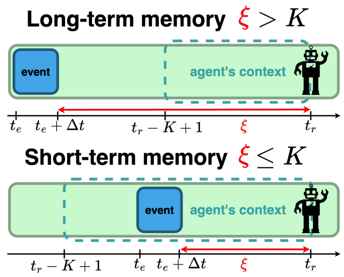
Reinforcement Learning (RL) addresses problems under the Markov Decision Process (MDP) framework, but partial observability remains challenging, requiring agents to process their interaction history [1, 2, 3]. In complex environments with noisy observations and long episodes, storing and retrieving key information is essential [4, 5]. Yet, the notion of *memory* in RL lacks a unified definition. Some works view it as handling dependencies within a fixed context [1, 6], others as leveraging out-of-context information [7], or as adapting to new environments in Meta-RL [8].

Without clear definitions and standardized evaluation, claims about agent memory are often vague or misleading. Memory is frequently attributed to recurrence or attention, but without isolating memory effects, such assumptions can be incorrect. Agents may appear to possess long-term memory due to task shortcuts, conflating mechanisms and obscuring limitations. This hampers fair comparison and progress toward genuinely memory-capable agents.

In this work, we clarify memory in RL by linking it directly to agent mechanisms. We formalize key memory types—short vs. long-term and declarative vs. procedural—and propose evaluation in memory-intensive tasks. Our classification, grounded in temporal dependencies and information type, enables fairer comparisons, diagnosis of architectural limits, and principled improvements. Importantly, our goal is not to replicate human memory, but to adapt established neuroscience concepts already used informally in RL [9, 10, 6].

Submitted to the 9th Conference on Robot Learning (CoRL 2025). Do not distribute.





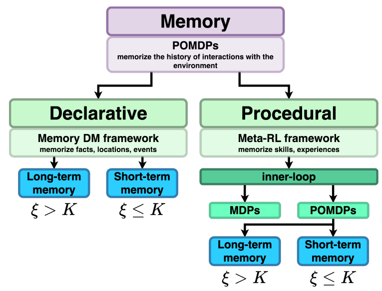


Figure 1: Illustration of declarative and procedural memory. Red arrows represent memorization steps, blue arrows indicate the recall of task-relevant information.

Figure 2: STM vs. LTM. t_e - event start, t_r - recall time; K – context length, ξ — correlation horizon. If the event lies beyond K, LTM is needed; if within, STM is enough.

Figure 3: Classification of memory types of RL agents. While the Memory DM framework contrasts with Meta-RL, its formalism can also describe inner-loop tasks when they are POMDPs.

In summary, our contributions are:

1. Formal definitions of memory types in RL—*short vs. long-term* and *declarative vs. procedural*—grounded in neuroscience (Section 4).
2. A task-level decoupling of *Memory Decision-Making* and *Meta-RL*, clarifying distinct roles of memory (Section 4).
3. A principled methodology to evaluate STM and LTM in Memory DM tasks, with criteria for memory boundaries (Section 4.2).
4. Evidence that neglecting this methodology leads to misleading claims about memory, underscoring the need for proper evaluation (Section 5).

2 Background

2.1 Memory of Humans and Agents

RL studies often reference memory types from cognitive science—long-term [11, 6], working [12], associative [13], episodic [14]—but typically reduce them to coarse temporal scales (short vs. long-term). Such simplifications ignore the relative nature of memory and hinder evaluation. We instead formalize agent memory types and propose a principled evaluation framework.

2.1.1 Memory in Cognitive Science

Human behavior relies on memory to acquire, retain, and reuse knowledge [15, 16]. Neuroscience distinguishes memory by temporal scale and content. At a high level, ***"memory is the ability to retain information and recall it later"***. This aligns with RL usage, and we adopt it to define agent memory types. Neuroscience separates *short-term* memory (seconds) from *long-term* memory (lifetime) [17], and *declarative* (explicit) from *procedural* (implicit) memory [18]. Declarative memory involves consciously recalled facts and events, while procedural memory covers unconscious skills. While established in biology, RL requires precise, testable counterparts. We adapt these categories into a formal framework for agents.

2.1.2 Memory in RL

Definitions of memory in RL vary widely. In Partially Observable Markov Decision Processes (POMDPs), agents must retain information for future use, involving two temporal dependencies: 1) within a bounded window (e.g., transformer context [1, 6, 19]); 2) beyond the current context, requiring persistent recall [7, 20]. As in Section 2.1.1, short- vs. long-term memory describe temporal scopes of declarative memory. Meta-RL instead reflects procedural memory, reusing skills across tasks [8]. Yet many works conflate these, testing "long-term memory" only in Meta-RL [21], without isolating decision-making from past observations. We address this by formalizing RL memory types via task structure and temporal dependencies. Our focus is on **declarative memory**, which governs decisions from past observations, in both short- and long-term forms.

2.1.3 Memory and Credit Assignment

Agent memory studies often separate *memory* from *credit assignment* [22, 6, 23]. Ni et al. [6] define memory as recalling past events and credit assignment as identifying when reward-relevant actions occurred. While distinct, both capture temporal dependencies. We therefore treat them jointly, adopting the general definition from Section 2.1.1, which unifies their shared temporal nature.

3 Related Works

Research on memory-enhanced RL has produced many architectures [24, 11, 9] and benchmarks [25, 26, 23, 27], yet "memory" remains inconsistently defined and misaligned with what experiments test.

Some define memory as retaining recent observations within an episode—via recurrence [2], transformer context [1, 28], or external stores [11, 29]. Others extend it to long-range dependencies through compression [30], key–value updates [31, 32], or spatial maps [33]. A separate view considers cross-episode transfer in Meta-RL [21, 34]. This diversity—from within-episode recall to multi-task adaptation—highlights the lack of a unified notion. Our work addresses this gap with a taxonomy grounded in temporal dependencies and task structure.

Ni et al. [6] separate memory (recalling past events) from credit assignment (linking rewards to actions). Kang et al. [10] emphasize reconstructive memory [35] as reflective interaction. These perspectives stress the need for precise definitions. We formalize memory types and propose an evaluation framework. Concurrently, Yue et al. [19] introduced memory dependency pairs (p, q) for imitation learning; while useful, they lack a theoretical base of RL memory and a broader taxonomy.

4 Memory Decision Making

POMDP tasks involving memory fall into two categories: *Meta-RL*, focused on skill transfer across tasks, and *Memory DM*, where agents recall past information for future decisions. This distinction matters: Meta-RL relies on procedural memory for rapid adaptation, while Memory DM uses declarative memory to guide decisions within a single environment. Yet many works reduce memory to temporal range, ignoring the behavioral roles that distinguish these types. To formalize Memory DM tasks, we first define the agent's context length:

Definition 4.1. ***Agent context length*** *($K \in \mathbb{N}$) – is the maximum number of previous steps (triplets of (o, a, r)) that the agent can process at time t.*

For example, an MLP-based agent processes one step at a time ($K = 1$), while a transformer-based agent can process a sequence of up to $K = K_{attn}$ triplets, where K_{attn} is determined by attention. Looking ahead, RNNs also have a $K = 1$, but using hidden states allows longer dependencies to be handled. Using the introduced Definition 4.1 for agent context length, we can introduce a formal definition for the Memory DM framework we focus on in this paper:

Definition 4.2. ***Memory Decision-Making (Memory DM)*** *– is a class of POMDPs in which the agents decision-making process at time t is based on the history $h_{0:t-1} = \{(o_i, a_i, r_i)\}_{i=0}^{t-1}$ if $t > 0$ otherwise $h = \varnothing$. The objective is to determine an optimal policy $\pi^*(a_t \mid o_t, h_{0:t-1})$ that maps the current observation o_t and history $h_{0:t-1}$ of length t to an action a_t, maximizing the expected cumulative reward within a single POMDP environment $\mathcal{M}_P$: $J^\pi = \mathbb{E}_\pi \left[\sum\limits_{t=0}^{T-1} \gamma^t r_t \right]$, where T – episode duration, $\gamma \in [0, 1]$ – discount factor.*

In the Memory DM framework (Definition 4.2), memory refers to the agent's ability to recall information from the past within a single environment and episode. In contrast, in the Meta-RL framework (Definition 4.3), memory involves recalling information about the agent's behavior from other environments or previous episodes:

Definition 4.3. ***Meta-RL*** *– is a class of POMDPs where the agent learns to learn from its past experiences across multiple tasks and memorize the common patterns and structures to facilitate efficient adaptation to new tasks. Let $\mathcal{D} = \{\tau_j^{\mathcal{M}_i}\}_{j=0}^{H-1}$ is all of the data of H episodes of length*

T collected in the MDP $\mathcal{M}_i \sim p(\mathcal{M})$. A Meta-RL algorithm is a function f_θ that maps the data $\mathcal{D}$ to a policy π_ϕ, where $\phi = f_\theta(\mathcal{D})$. The objective to determine an optimal f_θ: $J^\theta = \mathbb{E}_{\mathcal{M}_i \sim p(\mathcal{M})}\left[\mathbb{E}_{\mathcal{D}}\left[\sum_{\tau \in \mathcal{D}_{I:H}} G_i(\tau) \middle| f_\theta, \mathcal{M}_i\right]\right]$, where $G_i(\tau)$ – discounted return in the MDP $\mathcal{M}_i$, I – index of the first episode during the trial in which return counts towards the objective [36].

To operationalize the distinction between memory types in RL, we translate the neuroscience concepts of declarative and procedural memory (Section 2.1.1) into measurable task-level criteria:

Definition 4.4 (**Declarative and Procedural memory in RL**). *Let n_{envs} be the number of training environments and n_{eps} the number of episodes per environment. Then,*

1. ***Declarative Memory** – a type of agent memory when an agent transfers its knowledge within a single environment and across a single episode within that environment:*

$$\textit{Declarative Memory} \iff n_{envs} \times n_{eps} = 1. \tag{1}$$

2. ***Procedural Memory** – a type of agent memory when an agent transfers its skills across multiple environments or multiple episodes within a single environment:*

$$\textit{Procedural Memory} \iff n_{envs} \times n_{eps} > 1. \tag{2}$$

In this formulation, *knowledge* refers to observable, environment-specific information – such as object locations or facts – used within a single episode. *Skills*, in contrast, are policies reused across tasks or trials. Accordingly, Memory DM primarily evaluates declarative memory, while Meta-RL settings test procedural memory (Section 3).

Having distinguished declarative and procedural memory, we now examine the temporal structure of memory in the Memory DM framework, focusing on its division into short-term and long-term forms.

Definition 4.5 (**Memory DM types of memory**). *Let K be the agent context length, $\alpha_{t_e}^{\Delta t} = \{o_i, a_i, r_i\}_{i=t_e}^{t_e+\Delta t}$ – an event of duration Δt that begins at $t = t_e$ and ends at $t = t_e + \Delta t$, and $\beta_{t_r}(\alpha_{t_e}^{\Delta t}) = a_t \mid (o_t, \alpha_{t_e}^{\Delta t})$ – a decision-making point (recall) at time $t = t_r$ based on the current observation o_t and information about the event $\alpha_{t_e}^{\Delta t}$. Let also $\xi = t_r - t_e - \Delta t + 1$ be the **correlation horizon**, i.e. the minimal time delay between the event $\alpha_{t_e}^{\Delta t}$ that supports the decision-making and the moment of recall of this event β_{t_r}. Then,*

1. ***Short-term memory (STM)** - an agent's ability to use information about local correlations from the past within the context of length K at decision time:*

$$\beta_{t_r}(\alpha_{t_e}^{\Delta t}) = a_t \mid (o_t, \alpha_{t_e}^{\Delta t})\ \forall\ \xi = t_r - t_e - \Delta t + 1 \leq K.$$

2. ***Long-term memory (LTM)** - an agent ability to utilize information about global correlations from the past outside of the agent context of length K, during decision-making:*

$$\beta_{t_r}(\alpha_{t_e}^{\Delta t}) = a_t \mid (o_t, \alpha_{t_e}^{\Delta t})\ \forall\ \xi = t_r - t_e - \Delta t + 1 > K.$$

An illustration for the definitions of classifying Memory DM tasks into LTM and STM from Definition 4.5 is shown in 2.

The two definitions of declarative memory encompass all work related to Memory DM tasks, where decisions are based on past information. Meta-RL consists of an inner-loop, where the agent interacts with the environment $\mathcal{M} \sim p(\mathcal{M})$, and an outer-loop for transferring knowledge between tasks. Typically, $\mathcal{M}$ is an MDP that doesn't require memory, serving only the outer-loop, which is what "memory" refers to in Meta-RL studies.

The tasks in which the agent makes decisions based on interaction histories in the inner-loop are not named separately, since the classification of Meta-RL task types (multi-task, multi-task 0-shot, and single-task) is based solely on outer-loop parameters (n_{envs} and n_{eps}) and does not consider inner-loop task types. However, we can classify the agent's memory for these tasks as declarative short-term or long-term memory (Section 3).

We introduce an additional decoupling of Meta-RL task types into green (with POMDP inner-loop tasks) and blue (with MDP inner-loop tasks). In the green case, the agent's memory is required for both skill transfer in the outer-loop and decision-making based on interaction histories in the inner-loop, and therefore within the inner-loop can be considered as a Memory DM. In the blue case, memory is needed only for skill transfer. While this paper focuses on Memory DM tasks, the terminology allows for further classification of various Meta-RL tasks, with POMDP sub-classes highlighted in green. The proposed classification of tasks requiring agent memory is presented in Section 1.

Table 1: Classification of tasks requiring agent memory. Green marks tasks covered by our LTM/STM definitions, blue marks those outside. Meta-RL tasks with a POMDP inner-loop are marked green as Memory DM tasks. POMDP† denotes a Memory DM task treated as an inner-loop task without an outer-loop.

n_{envs}	n_{eps}	**POMDP**	**Inner-loop task**	**Memory**	**Tasks that require agent memory**	
					Memory DM	
					LTM $\xi > K$	STM $\xi \leq K$
1	1	Memory DM	POMDP†	Dec.	Long-term memory task	Short-term memory task
					Meta-RL: Outer-loop and inner-loop memory	
					LTM $\xi > K$	STM $\xi \leq K$
1	¿1	Meta-RL	POMDP	Proc.	Single-task	Single-task
¿1	1	Meta-RL	POMDP	Proc.	Multi-task 0-shot	Multi-task 0-shot
¿1	¿1	Meta-RL	POMDP	Proc.	Multi-task	Multi-task
					Meta-RL: Outer-loop memory only	
					No memory $\xi = 1$	No memory $\xi = 1$
1	¿1	Meta-RL	MDP	Proc.	Single-task	Single-task
¿1	1	Meta-RL	MDP	Proc.	Multi-task 0-shot	Multi-task 0-shot
¿1	¿1	Meta-RL	MDP	Proc.	Multi-task	Multi-task

4.1 Memory-intensive environments

To effectively test a Memory DM agent's use of short-term and long-term memory, it is crucial to design appropriate experiments. Not all environments are suitable for assessing agent memory; for example, omnipresent Atari games [37] with frame stacking or MuJoCo control tasks [38] may yield unrepresentative results. To facilitate the evaluation of agent memory capabilities, we formalize the definition of memory-intensive environments:

Definition 4.6 (**Memory-Intensive Environments**). *Let $\mathcal{M}_P$ be a POMDP, and let $\Xi = \{\xi_n\}_n = \{(t_r - t_e - \Delta t + 1)_n\}_n$ denote the set of correlation horizons for all event-recall pairs $(\alpha_{t_e}^{\Delta t}, \beta_{t_r})$. Then $\mathcal{M}_P$ is a memory-intensive environment, denoted $\tilde{\mathcal{M}}_P$, if and only if:* $\min_n \xi_n > 1$.

Corollary 1. *A task corresponds to an MDP (i.e., is Markovian) if and only if all correlation horizons are trivial:* $\max\limits_n \Xi = 1$.

Proof. In an MDP, the optimal action depends only on the current state (or observation), i.e., no past information is needed. This implies $\xi_n = 1$ for all event-recall pairs, hence $\max_n \xi_n = 1$. Conversely, if $\max_n \xi_n = 1$, then no decision depends on events beyond the current step, satisfying the Markov property. ∎

Using the definitions of memory-intensive environments (Definition 4.6) and agent memory types (Definition 4.5), we can configure experiments to test short-term and long-term memory in the Memory DM framework. Notably, the same memory-intensive environment can validate both types of memory, as outlined in 2:

Theorem 2 (**On the context memory border**). *Let $\tilde{\mathcal{M}}_P$ be a memory-intensive environment and K be an agents context length. Then there exists context memory border $\overline{K} \geq 1$ such that if $K \leq \overline{K}$ then the environment $\tilde{\mathcal{M}}_P$ is used to validate exclusively long-term memory in Memory DM framework:*

$$\exists\, \overline{K} \geq 1 : \forall\, K \in [1, \overline{K}] : K < \min_n \Xi. \tag{3}$$

Proof. Let $\overline{K} = \min \Xi - 1$. Then $\forall\, K \leq \overline{K}$ is guaranteed that no correlation horizon ξ is in the agent history $h_{t-K+1:t}$, hence the context length $K \leq \min \Xi - 1$ generates the LTM problem exclusively. Since context length cannot be negative or zero, it turns out that $1 \leq K \leq \overline{K} = \min \Xi - 1$, which was required to prove. ∎

The following result, though intuitive, formalizes a practical criterion for isolating long-term memory evaluation by constraining the agent's context window. It serves as the foundation for configuring experiments in the Memory DM framework. According to Theorem 2, in a memory-intensive environment $\tilde{\mathcal{M}}_P$, the value of the context memory border $\overline{K}$ can be found as

$$\overline{K} = \min \Xi - 1 = \min_n \left\{ (t_r - t_e - \Delta t + 1)_n \right\}_n - 1. \quad (4)$$

Using Theorem 2, we can establish the necessary conditions for validating short-term memory: **1) Weak condition to validate short-term memory**: if $\overline{K} < K < \max \Xi$, then the memory-intensive environment $\tilde{M}_P$ is used to validate both short-term and long-term memory. **2) Strong condition to validate short-term memory**: if $\max \Xi < K$, then the memory-intensive environment $\tilde{M}_P$ is used to validate exclusively short-term memory.

According to Theorem 2, if $K \in [1, \overline{K}]$, none of the correlation horizons ξ will be in the agent's context, validating only long-term memory. When $\overline{K} < K < \max \Xi \leq T - 1$, long-term memory can still be tested, but some correlation horizons ξ will fall within the agent's context and won't be used for long-term memory validation. In such a case it is not possible to estimate long-term memory explicitly. When $K \geq \max \Xi$, all correlation horizons ξ are within the agent's context, validating only short-term memory. Summarizing the obtained results, the final division of the required agent context lengths for short-term memory and long-term memory validation is as follows: **(i)** $K \in [1, \overline{K}] \Rightarrow$ validating LTM only; **(ii)** $K \in (\overline{K}, \max \Xi) \Rightarrow$ validating both STM and LTM; **(iii)** $K \in [\max \Xi, \infty) \Rightarrow$ validating STM only.

Algorithm 1 Setup for testing long- and short-term memory in the Memory DM framework.

Require: $\tilde{\mathcal{M}}_P$ – memory-intensive environment; $\mu(K)$ – memory mechanism. =0

1. Count n event–recall pairs (Def. 4.6).
1. $n = 0$: environment unsuitable.
2. $n \geq 1$: environment suitable.

2. Estimate context border $\overline{K}$ (4).
1. For each pair $(\beta(\alpha), \alpha)_i$, compute ξ_i.
2. Set $\overline{K} = \min \Xi - 1$.

3. Run experiment (Def. 4.5).
1. STM: $K > \overline{K}$.
2. LTM: $K \leq \overline{K} \leq K_{eff} = \mu(K)$.

4. Analyze results.

4.2 Long-term memory in Memory DM

As defined in Definition 4.5, short-term Memory DM tasks arise when event-recall pairs in $\tilde{\mathcal{M}}_P$ fall within the agent's context ($\xi \leq K$), allowing decisions based on local correlations. This holds regardless of how large K is. Examples include [1, 28, 6]. Validating STM is simple: increase K. In contrast, testing long-term memory requires more care and is typically more informative.

Memory DM tasks requiring long-term memory occur when event-recall pairs in the memory-intensive environment $\tilde{\mathcal{M}}_P$ are outside the agent's context ($\xi > K$). In this case, memory involves the agent's ability to connect information beyond its context, necessitating memory mechanisms (Definition 4.7) that can manage interaction histories h longer than the agent's base model can handle.

Definition 4.7 (**Memory mechanisms**). *Let the agent process histories $h_{t-K+1:t}$ of length K at the current time t, where $K \in \mathbb{N}$ is agents context length. Then, a **memory mechanism** $\mu(K) : \mathbb{N} \to \mathbb{N}$ is defined as a function that, for a fixed K, allows the agent to process sequences of length $K_{eff} \geq K$, i.e., to establish global correlations out of context, where K_{eff} is the effective context.*

$$\mu(K) = K_{eff} \geq K. \quad (5)$$

Memory mechanisms are key to solving LTM tasks by accessing out-of-context data in Memory DM.

Example of memory mechanism. Consider an agent based on an RNN architecture that can process $K = 1$ triplets of tokens (o_t, a_t, r_t) at all times t. By using memory mechanisms $\mu(K)$ (e.g., as in [2]), the agent can increase the number of tokens processed in a single step without expanding the context size of its RNN architecture. Therefore, if initially in a memory-intensive environment $\tilde{\mathcal{M}}_P : \xi > K = 1$, it can now be represented as $\tilde{\mathcal{M}}_P : \xi \leq K_{eff} = \mu(K)$. Here, the memory mechanism $\mu(K)$ refers to the RNNs recurrent updates to its hidden state.

Thus, validating an agent's ability to solve long-term memory problems in the Memory DM framework reduces to validating the agent's memory mechanisms $\mu(K)$. **To design correct experiments in such a case, the following condition must be met:**

$$\tilde{\mathcal{M}}_P : K \leq \overline{K} < \xi \leq K_{eff} = \mu(K) \quad (6)$$

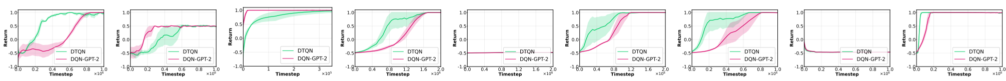


Figure 5: Performance of Online RL agents “with memory” across different memory configurations. Each environment (T-Maze, POPGym-Autoencode, POPGym-RepeatPrevious) is shown with varying agent context length K and correlation horizons ξ. The STM $\leftrightarrows$ LTM transitions reflect the relative nature of the settings to test memory, depending on both agent and environment parameters.

According to our definitions, agents with memory mechanisms within the Memory DM framework that can solve long-term memory tasks can also handle short-term memory tasks, but not vice versa. The algorithm for setting up experiments to test an agent’s short-term or long-term memory is outlined in Section 1.

4.3 Example of Ξ and ξ estimates

Following the proposed methodology (Algorithm 1), we estimated the sets of correlation horizons Ξ and minimal recall distances ξ for a range of popular memory-intensive tasks (Table 4), including *Passive T-Maze* [6], *Minigrid-Memory* [39], *ViZDoom-Two-Colors* [20], *Memory Maze* [40], *Memory Cards* [1], *Mortar Mayhem* and *Mystery Path* [41], *POPGym–Autoencode* and *POPGym-RepeatPrevious* [25].

Figure 4: Correlation horizons ξ and LTM thresholds K for popular memory-intensive tasks. L is corridor length, T is episode length. (f) and (r) denote fixed and variable setups. POPGym entries show values for the easy setting; for easy / medium / hard, Ξ becomes $\{2, 4, \ldots, 104/208/312\}$ for Autoencode and $\{5/33/65\}$ for RepeatPrevious.

Task	Ξ	ξ	**LTM:** $K <$
Passive T-Maze	$\{L+1\}$	$L+1$	$L+1$
Minigrid-Memory (f)	$\{L+1\}$	$L+1$	$L+1$
Minigrid-Memory (v)	$[7, L+1]$	7	7
ViZDoom-Two-Colors	$[2, 2055]$	2	2
Memory Maze 9x9	$[28, 1000]$	28	28
Memory Maze 15x15	$[45, 4000]$	45	45
Memory Cards	$[2, T]$	2	2
Mortar Mayhem (finite)	$[38, 218]$	38	38
Mystery Path (finite)	$[8, 26]$	8	8
POPGym–Autoencode	$[2, 104]$	2	2
POPGym–RepeatPrevious	$\{5\}$	5	5

Example: Testing Memory in Passive T-Maze In Passive T-Maze, the agent sees a cue at the start of a corridor and must turn correctly at the junction. The episode lasts $T = L+1$. Using Algorithm 1: **1)** There’s one event-recall pair ($n = 1$), so the task suits both STM and LTM. **2)** The event lasts one step ($\Delta t = 0$), so $\xi = T$, and $\overline{K} = T - 1$. **3)** Varying T or context size K lets us test STM (if $K > \overline{K}$) or LTM (if $K \le \overline{K} \le \mu(K)$). While $K = \overline{K}$ is enough in theory, choosing smaller K better reveals memory mechanism effects.

5 Experiments

We evaluate memory-enhanced RL agents with the Memory DM framework to distinguish STM and LTM. Our study highlights the importance of proper methodology (Section 1) and shows how poor setups can misrepresent memory. We test four tasks: Passive T-Maze and Minigrid-Memory (cue recall), and POPGym–Autoencode and RepeatPrevious (reconstruction and repetition), all requiring temporal recall. In the online setting, we assess DTQN [1], DQN-GPT-2, and SAC-GPT-2 [6] with attention-based memory. Offline, we compare DT [42] and BC-LSTM, contrasting attention with recurrence. Across settings, we vary agent context K and task horizon ξ to isolate memory types and expose model limits.

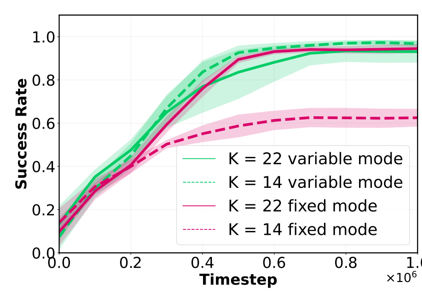


Figure 6: SAC-GPT-2 in Minigrid-Memory ($L = 21$) with short- ($K = 22$) and long-term ($K = 14$) memory setups. Variable mode (green) masks memory limits; fixed mode (red) reveals failure at $K = 14$, demonstrating lack of long-term memory—made evident by our evaluation method.

5.1 Pitfalls of Naive Memory Evaluation

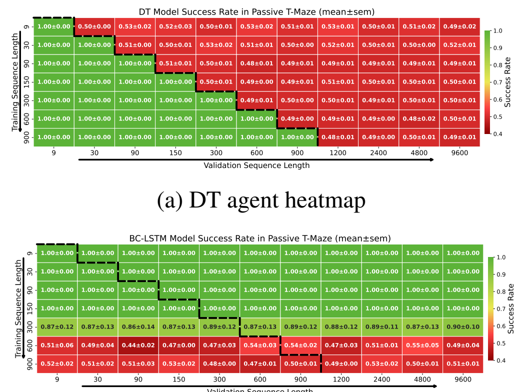
Proper memory evaluation in RL requires separating STM from LTM by controlling correlation horizons ξ. Otherwise, STM and LTM effects blur. We illustrate this with SAC-GPT-2 in Minigrid-Memory under (i) fixed $L = 21$ ($\xi = 22$) and (ii) variable L ($\xi \in [7, 22]$), testing STM ($K = 22$) and LTM ($K = 14$). As shown in Section 6, the variable setup yields high success in both cases, suggesting good memory. Yet in the fixed case, LTM fails, exposing the true limit. Mixed-horizon tasks thus mask LTM deficits; only fixed $\xi > K$ reveals them. Accurate LTM evaluation therefore requires aligning ξ with K, which our methodology ensures.

5.2 The Relative Nature of an Agent's Memory

According to Algorithm 1, testing STM vs. LTM depends on two parameters: agent context K and the environment's correlation horizon ξ (with border $\overline{K}$). Verifying memory requires varying one while fixing the other. Thus, memory is not intrinsic to the agent but emerges from the K–ξ interplay, so the same agent may show STM or LTM depending on setup. We evaluate DTQN and DQN-GPT-2 on Passive T-Maze, POPGym Autoencode, and RepeatPrevious by varying K and ξ. As shown in Figure 5, performance is high for $\xi \leq K$ but collapses for $\xi > K$, demonstrating that long-range dependencies demand explicit memory. These findings confirm that LTM is relative to both task horizon and agent design. Without controlling K and ξ, memory claims are unreliable. Our framework provides consistent, interpretable evaluation.

5.3 Generalization Across Sequence Lengths

Evaluating memory in RL requires separating true long-term memory (LTM) from fixed-context recall. We illustrate this using DT and BC-LSTM on T-Maze: agents are trained on specific corridor lengths and tested on both seen and longer ones. Success heatmaps (Figure 7) show in-distribution along the diagonal and extrapolation to the right.



(a) DT agent heatmap

(b) BC-LSTM agent heatmap

Figure 7: Generalization on Passive T-Maze. Each heatmap shows success rates for (a) DT and (b) BC-LSTM across training (vertical) and validation (horizontal) sequence lengths. DT succeeds only when validation $\leq$ training, showing short-term memory limits. BC-LSTM generalizes beyond training, indicating strong long-term memory.

Although both are labeled memory-based, our framework highlights clear differences. DT relies on a fixed attention window, supporting only short-term memory, while LSTM's recurrent state enables LTM. DT performs well within context but fails for $L > 90$; BC-LSTM generalizes further, though performance degrades at long training lengths (600, 900). If evaluated only on shorter lengths, DT may appear stronger, masking its lack of LTM.

Thus, DT exhibits only STM, whereas BC-LSTM demonstrates LTM despite gradient challenges [43, 44]. Our framework prevents such misinterpretations, showing that DT is suited to STM tasks via attention, while BC-LSTM supports LTM through recurrence.

6 Conclusion

We propose a unified framework for classifying and evaluating memory in RL agents, grounded in neuroscience-inspired definitions of short- and long-term declarative memory. By introducing the concept of correlation horizon and formalizing memory-intensive environments, we enable precise evaluation of agent memory. Our methodology reveals key architectural differences: transformers such as DTQN or DT rely mainly on short-term memory, while recurrent models like BC-LSTM demonstrate true long-term memory. Experiments on T-Maze, MiniGrid, and POPGym highlight the need for proper setup to avoid misleading conclusions. Overall, our framework clarifies how different memory mechanisms shape agent behavior. Future work may extend it to other cognitive memory systems (e.g., working, episodic) and investigate whether new types emerge in complex RL tasks.

References

[1] K. Esslinger, R. Platt, and C. Amato. Deep transformer q-networks for partially observable reinforcement learning. *arXiv preprint arXiv:2206.01078*, 2022.

[2] M. Hausknecht and P. Stone. Deep recurrent q-learning for partially observable mdps, 2015.

[3] T. Ni, B. Eysenbach, and R. Salakhutdinov. Recurrent model-free rl can be a strong baseline for many pomdps. *arXiv preprint arXiv:2110.05038*, 2021.

[4] A. Goyal, A. L. Friesen, A. Banino, T. Weber, N. R. Ke, A. P. Badia, A. Guez, M. Mirza, P. C. Humphreys, K. Konyushkova, L. Sifre, M. Valko, S. Osindero, T. Lillicrap, N. Heess, and C. Blundell. Retrieval-augmented reinforcement learning, 2022. URL https://arxiv.org/abs/2202.08417.

[5] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. Gómez, E. Grefenstette, T. Ramalho, J. Agapiou, A. Badia, K. Hermann, Y. Zwols, G. Ostrovski, A. Cain, H. King, C. Summerfield, P. Blunsom, K. Kavukcuoglu, and D. Hassabis. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538, 10 2016. doi:10.1038/nature20101.

[6] T. Ni, M. Ma, B. Eysenbach, and P.-L. Bacon. When do transformers shine in RL? decoupling memory from credit assignment. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=APGXBNkt6h.

[7] E. Parisotto, F. Song, J. Rae, R. Pascanu, C. Gulcehre, S. Jayakumar, M. Jaderberg, R. L. Kaufman, A. Clark, S. Noury, et al. Stabilizing transformers for reinforcement learning. In *International conference on machine learning*, pages 7487–7498. PMLR, 2020.

[8] A. A. Team, J. Bauer, K. Baumli, S. Baveja, F. Behbahani, A. Bhoopchand, N. Bradley-Schmieg, M. Chang, N. Clay, A. Collister, V. Dasagi, L. Gonzalez, K. Gregor, E. Hughes, S. Kashem, M. Loks-Thompson, H. Openshaw, J. Parker-Holder, S. Pathak, N. Perez-Nieves, N. Rakicevic, T. Rocktäschel, Y. Schroecker, J. Sygnowski, K. Tuyls, S. York, A. Zacherl, and L. Zhang. Human-timescale adaptation in an open-ended task space, 2023. URL https://arxiv.org/abs/2301.07608.

[9] M. Fortunato, M. Tan, R. Faulkner, S. Hansen, A. P. Badia, G. Buttimore, C. Deck, J. Z. Leibo, and C. Blundell. Generalization of reinforcement learners with working and episodic memory, 2020. URL https://arxiv.org/abs/1910.13406.

[10] Y. Kang, E. Zhao, Y. Zang, L. Li, K. Li, P. Tao, and J. Xing. Sample efficient reinforcement learning using graph-based memory reconstruction. *IEEE Transactions on Artificial Intelligence*, 5(2):751–762, 2024. doi:10.1109/TAI.2023.3268612.

[11] A. Lampinen, S. Chan, A. Banino, and F. Hill. Towards mental time travel: a hierarchical memory for reinforcement learning agents. *Advances in Neural Information Processing Systems*, 34:28182–28195, 2021.

[12] A. Graves, G. Wayne, and I. Danihelka. Neural turing machines, 2014. URL https://arxiv.org/abs/1410.5401.

[13] P. G. Polson. *The American Journal of Psychology*, 88(1):131–140, 1975. ISSN 00029556. URL http://www.jstor.org/stable/1421672.

[14] A. Pritzel, B. Uria, S. Srinivasan, A. Puigdomènech, O. Vinyals, D. Hassabis, D. Wierstra, and C. Blundell. Neural episodic control, 2017. URL https://arxiv.org/abs/1703.01988.

[15] T. Parr, R. V. Rikhye, M. M. Halassa, and K. J. Friston. Prefrontal computation as active inference. *Cerebral Cortex*, 30(2):682–695, 2020.

[16] T. Parr, G. Pezzulo, and K. J. Friston. *Active inference: the free energy principle in mind, brain, and behavior*. MIT Press, 2022.

[17] H. Davis and L. Squire. Davis hp, squire lr. protein synthesis and memory: a review. psychol bull 96: 518-559. *Psychological bulletin*, 96:518–59, 11 1984. doi:10.1037/0033-2909.96.3.518.

[18] P. Graf and D. Schacter. Implicit and explicit memory for new associations in normal and amnesic subjects. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, 11: 501–518, 1985.

[19] W. Yue, B. Liu, and P. Stone. Learning memory mechanisms for decision making through demonstrations. *arXiv preprint arXiv:2411.07954*, 2024.

[20] A. Sorokin, N. Buzun, L. Pugachev, and M. Burtsev. Explain my surprise: Learning efficient long-term memory by predicting uncertain outcomes. *Advances in Neural Information Processing Systems*, 35:36875–36888, 2022.

[21] J. Kang, R. Laroche, X. Yuan, A. Trischler, X. Liu, and J. Fu. Think before you act: Decision transformers with working memory, 2024. URL `https://arxiv.org/abs/2305.16338`.

[22] T. Mesnard, T. Weber, F. Viola, S. Thakoor, A. Saade, A. Harutyunyan, W. Dabney, T. Stepleton, N. Heess, A. Guez, et al. Counterfactual credit assignment in model-free reinforcement learning. *arXiv preprint arXiv:2011.09464*, 2020.

[23] I. Osband, Y. Doron, M. Hessel, J. Aslanides, E. Sezener, A. Saraiva, K. McKinney, T. Lattimore, C. Szepesvari, S. Singh, et al. Behaviour suite for reinforcement learning. *arXiv preprint arXiv:1908.03568*, 2019.

[24] J. Oh, V. Chockalingam, S. Singh, and H. Lee. Control of memory, active perception, and action in minecraft, 2016. URL `https://arxiv.org/abs/1605.09128`.

[25] S. Morad, R. Kortvelesy, M. Bettini, S. Liwicki, and A. Prorok. Popgym: Benchmarking partially observable reinforcement learning, 2023. URL `https://arxiv.org/abs/2303.01859`.

[26] E. Cherepanov, N. Kachaev, A. Kovalev, and A. Panov. Memory, benchmark & robots: A benchmark for solving complex tasks with reinforcement learning. In *7th Robot Learning Workshop: Towards Robots with Human-Level Abilities*, 2025.

[27] M. Pleines, M. Pallasch, F. Zimmer, and M. Preuss. Memory gym: Partially observable challenges to memory-based agents in endless episodes. *arXiv preprint arXiv:2309.17207*, 2023.

[28] J. Grigsby, L. Fan, and Y. Zhu. Amago: Scalable in-context reinforcement learning for adaptive agents, 2024. URL `https://arxiv.org/abs/2310.09971`.

[29] H. Le, K. Do, D. Nguyen, S. Gupta, and S. Venkatesh. Stable hadamard memory: Revitalizing memory-augmented agents for reinforcement learning. *arXiv preprint arXiv:2410.10132*, 2024.

[30] S. Morad, R. Kortvelesy, S. Liwicki, and A. Prorok. Reinforcement learning with fast and forgetful memory. *Advances in Neural Information Processing Systems*, 36:72008–72029, 2023.

[31] S. Pramanik, E. Elelimy, M. C. Machado, and A. White. Recurrent linear transformers. *arXiv preprint arXiv:2310.15719*, 2023.

[32] E. Cherepanov, A. Staroverov, D. Yudin, A. K. Kovalev, and A. I. Panov. Recurrent action transformer with memory. *arXiv preprint arXiv:2306.09459*, 2024. URL `https://arxiv.org/abs/2306.09459`.

[33] E. Parisotto and R. Salakhutdinov. Neural map: Structured memory for deep reinforcement learning. *arXiv preprint arXiv:1702.08360*, 2017.

[34] J. Bauer, K. Baumli, F. Behbahani, A. Bhoopchand, N. Bradley-Schmieg, M. Chang, N. Clay, A. Collister, V. Dasagi, L. Gonzalez, K. Gregor, E. Hughes, S. Kashem, M. Loks-Thompson, H. Openshaw, J. Parker-Holder, S. Pathak, N. Perez-Nieves, N. Rakicevic, T. Rocktäschel, Y. Schroecker, S. Singh, J. Sygnowski, K. Tuyls, S. York, A. Zacherl, and L. M. Zhang. Human-timescale adaptation in an open-ended task space. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 1887–1935. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/bauer23a.html.

[35] F. C. Bartlett and W. Kintsch. *Remembering: A Study in Experimental and Social Psychology*. Cambridge University Press, 2 edition, 1995.

[36] J. Beck, R. Vuorio, E. Z. Liu, Z. Xiong, L. Zintgraf, C. Finn, and S. Whiteson. A survey of meta-reinforcement learning, 2024. URL https://arxiv.org/abs/2301.08028.

[37] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[38] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2021.

[39] M. Chevalier-Boisvert, B. Dai, M. Towers, R. de Lazcano, L. Willems, S. Lahlou, S. Pal, P. S. Castro, and J. Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *CoRR*, abs/2306.13831, 2023.

[40] J. Pasukonis, T. Lillicrap, and D. Hafner. Evaluating long-term memory in 3d mazes, 2022. URL https://arxiv.org/abs/2210.13383.

[41] M. Pleines, M. Pallasch, F. Zimmer, and M. Preuss. Memory gym: Towards endless tasks to benchmark memory capabilities of agents. *Journal of Machine Learning Research*, 26(6):1–40, 2025.

[42] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[43] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. Pmlr, 2013.

[44] T. Trinh, A. Dai, T. Luong, and Q. Le. Learning longer-term dependencies in rnns with auxiliary losses. In *International Conference on Machine Learning*, pages 4965–4974. PMLR, 2018.

[45] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986. URL https://api.semanticscholar.org/CorpusID:205001834.

[46] D. Wierstra, A. Förster, J. Peters, and J. Schmidhuber. Recurrent policy gradients. *Logic Journal of the IGPL*, 18:620–634, 10 2010. doi:10.1093/jigpal/jzp049.

[47] I. Sorokin, A. Seleznev, M. Pavlov, A. Fedorov, and A. Ignateva. Deep attention recurrent q-network, 2015. URL https://arxiv.org/abs/1512.01693.

[48] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning, 2016. URL https://arxiv.org/abs/1611.02779.

[49] D. R. Song, C. Yang, C. McGreavy, and Z. Li. Recurrent deterministic policy gradient method for bipedal locomotion on rough terrain challenge, Nov. 2018. URL http://dx.doi.org/10.1109/ICARCV.2018.8581309.

[50] L. Zintgraf, K. Shiarlis, M. Igl, S. Schulze, Y. Gal, K. Hofmann, and S. Whiteson. Varibad: A very good method for bayes-adaptive deep rl via meta-learning, 2020. URL https://arxiv.org/abs/1910.08348.

[51] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[52] L. C. Melo. Transformers are meta-reinforcement learners, 2022. URL https://arxiv.org/abs/2206.06614.

[53] J. Robine, M. Höftmann, T. Uelwer, and S. Harmeling. Transformer-based world models are happy with 100k interactions. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=TdBaDGCpjly.

[54] G. Shala, A. Biedenkapp, and J. Grabocka. Hierarchical transformers are efficient meta-reinforcement learners, 2024. URL https://arxiv.org/abs/2402.06402.

[55] A. Gu, K. Goel, and C. Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

[56] J. T. H. Smith, A. Warrington, and S. W. Linderman. Simplified state space layers for sequence modeling, 2023. URL https://arxiv.org/abs/2208.04933.

[57] A. Gu and T. Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[58] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2555–2565. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/hafner19a.html.

[59] C. Lu, Y. Schroecker, A. Gu, E. Parisotto, J. Foerster, S. Singh, and F. Behbahani. Structured state space models for in-context reinforcement learning, 2023. URL https://arxiv.org/abs/2303.03982.

[60] P. Becker, N. Freymuth, and G. Neumann. Kalmamba: Towards efficient probabilistic state space models for rl under uncertainty, 2024. URL https://arxiv.org/abs/2406.15131.

[61] M. R. Samsami, A. Zholus, J. Rajendran, and S. Chandar. Mastering memory tasks with world models, 2024. URL https://arxiv.org/abs/2403.04253.

[62] T. D. YuXuan Liu and W. Hsieh. Temporal convolutional policy networks, 2016. URL https://yuxuanliu.com/files/tcpn.pdf.

[63] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel. A simple neural attentive meta-learner, 2018. URL https://arxiv.org/abs/1707.03141.

[64] D. Ha and J. Schmidhuber. Recurrent world models facilitate policy evolution, 2018. URL https://arxiv.org/abs/1809.01999.

[65] S. D. Morad, S. Liwicki, R. Kortvelesy, R. Mecca, and A. Prorok. Graph convolutional memory using topological priors, 2021. URL https://arxiv.org/abs/2106.14117.

[66] D. Zhu, L. E. Li, and M. Elhoseiny. Value memory graph: A graph-structured world model for offline reinforcement learning, 2023. URL https://arxiv.org/abs/2206.04384.

[67] W. Zaremba and I. Sutskever. Reinforcement learning neural turing machines - revised, 2016. URL https://arxiv.org/abs/1505.00521.

[68] E. Parisotto and R. Salakhutdinov. Neural map: Structured memory for deep reinforcement learning, 2017. URL https://arxiv.org/abs/1702.08360.

[69] B. Deverett, R. Faulkner, M. Fortunato, G. Wayne, and J. Z. Leibo. Interval timing in deep reinforcement learning agents. *Advances in Neural Information Processing Systems*, 32, 2019.

A Appendix – Glossary

In this section, we provide a comprehensive glossary of key terms and concepts used throughout this paper. The definitions are intended to clarify the terminology proposed in our research and to ensure that readers have a clear understanding of the main elements underpinning our work.

1. $\mathcal{M}$ – MDP environment
2. $\mathcal{M}_P$ – POMDP environment
3. $\tilde{\mathcal{M}}_P$ – memory-intensive environment
4. $h_{0:t-1} = \{(o_i, a_i, r_i)\}_{i=0}^{t-1}$ – agent history of interactions with environment
5. K – agent base model context length
6. $\overline{K}$ – context memory border of the agent, such that $K \in [1, \overline{K}] \Leftrightarrow$ strictly LTM problem
7. $\mu(K)$ – memory mechanism that increases number of steps available to the agent to process
8. $K_{eff} = \mu(K)$ – the agent effective context after applying the memory mechanism
9. $\alpha_{t_e}^{\Delta t} = \{(o_i, a_i, r_i)\}_{i=t_e}^{t_e+\Delta t}$ – an event starting at time t_e and lasting Δt, which the agent should recall when making a decision in the future
10. $\beta_{t_r} = \beta_{t_r}(\alpha_{t_e}^{\Delta t}) = a_t \mid (o_t, \alpha_{t_e}^{\Delta t})$ – the moment of decision making at time t_r according to the event $\alpha_{t_e}^{\Delta t}$
11. $\xi = t_r - t_a - \Delta t + 1$ – an event's correlation horizon

B Appendix – Additional notes on the motivation for the article

B.1 Why use definitions from neuroscience?

Definitions from neuroscience and cognitive science, such as short-term and long-term memory, as well as declarative and procedural memory, are already well-established in the RL community, but do not have common meanings and are interpreted in different ways. We strictly formalize these definitions to avoid possible confusion that may arise when introducing new concepts and redefine them with clear, quantitative meanings to specify the type of agent memory, since the performance of many algorithms depends on their type of memory.

In focusing exclusively on memory within RL, we do not attempt to exhaustively replicate the full spectrum of human memory. Instead, our goal is to leverage the intuitive understanding of neuroscience concepts already familiar to RL researchers. This approach avoids the unnecessary introduction of new terminology into the already complex Memory RL domain. By refining and aligning existing definitions, we create a robust framework that facilitates clear communication, rigorous evaluation, and practical application in RL research.

B.2 On practical applications of our framework

The primary goal of our framework is to address practical challenges in RL by providing a robust classification of memory types based on temporal dependencies and the nature of memorized information. This classification is essential for standardizing memory testing and ensuring that RL agents are evaluated under conditions that accurately reflect their capabilities.

In RL, memory is interpreted in various ways, such as transformers with large context windows, recurrent networks, or models capable of skill transfer across tasks. However, these approaches often vary fundamentally in design, making comparisons unreliable and leading to inconsistencies in testing. Our framework resolves this by providing a clear structure to evaluate memory mechanisms under uniform and practical conditions.

The proposed definitions of declarative and procedural memory use two straightforward numerical parameters: the number of environments (n_{envs}) and episodes (n_{eps}). These parameters allow researchers to reliably determine the type of memory required for a task. This simplicity and alignment with numerical parameters make the framework practical and widely applicable across diverse RL problems.

Moreover, the division of declarative memory into long-term and short-term memory, as well as the need to use a balance between the agent's context length K and the correlation horizons of the environment ξ when conducting the experiment, allows us to unambiguously determine which type of memory is present in the agent. This clarity ensures fair comparisons between agents with similar memory mechanisms and highlights specific limitations in an agent's design. By aligning memory definitions with practical testing requirements, the framework provides actionable insights to guide the development of memory-enhanced RL agents.

C Appendix – Memory Mechanisms

In RL, memory has several meanings, each of which is related to a specific class of different tasks. To solve these tasks, the authors use various memory mechanisms. The most prevalent approach to incorporating memory into an agent is through the use of Recurrent Neural Networks (RNNs) [45], which are capable of handling sequential dependencies by maintaining a hidden state that captures information about previous time steps [46, 2, 47, 48, 49, 50]. Another popular way to implement memory is to use Transformers [51], which use self-attention mechanisms to capture dependencies inside the context window [7, 11, 1, 52, 8, 31, 53, 6, 28, 54]. State-space models (SSMs) [55, 56, 57] combine the strengths of RNNs and Transformers and can also serve to implement memory through preservation of system state [58, 59, 60, 61]. Temporal convolutions may be regarded as an effective memory mechanism, whereby information is stored implicitly through the application of learnable filters across the time axis [62, 63]. A world model [64] which builds an internal environment representation can also be considered as a form of memory. One method for organizing this internal representation is through the use of a graph, where nodes represent observations within the environment and edges represent actions [65, 66, 10].

A distinct natural realization of memory is the utilization of an external memory buffer, which enables the agent to retrieve pertinent information. This approach can be classified into two categories: read-only (writeless) [24, 11, 4, 32] and read/write access [5, 67, 68].

Memory can also be implemented without architectural mechanisms, relying instead on agent policy. For instance, in the work of Deverett et al. [69], the agent learns to encode temporal intervals by generating specific action patterns. This approach allows the agent to implicitly represent timing information within its behavior, showcasing that memory can emerge as a result of policy adaptations rather than being explicitly embedded in the underlying neural architecture.

Using these memory mechanisms, both decision-making tasks based on information from the past within a single episode and tasks of fast adaptation to new tasks are solved. However, even in works using the same underlying base architectures to solve the same class of problems, the concepts of memory may differ.

D Appendix – POMDP

D.1 POMDP

The Partially Observable Markov Decision Process (POMDP) is a generalization of the Markov Decision Process (MDP) that models sequential decision-making problems where the agent has incomplete information about the environment's state. POMDP can be represented as a tuple

$\mathcal{M}_P = \langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{P}, \mathcal{R}, \mathcal{Z} \rangle$, where $\mathcal{S}$ denotes the set of states, $\mathcal{A}$ is the set of actions, $\mathcal{O}$ is the set of observations and $\mathcal{Z} = \mathcal{P}(o_{t+1} \mid s_{t+1}, a_t)$ is an observation function such that $o_{t+1} \sim \mathcal{Z}(s_{t+1}, a_t)$. An agent takes an action $a_t \in \mathcal{A}$ based on the observed history $h_{0:t-1} = \{(o_i, a_i, r_i)\}_{i=0}^{t-1}$ and receives a reward $r_t = \mathcal{R}(s_t, a_t)$. It is important to note that state s_t is not available to the agent at time t. In the case of POMDPs, a policy is a function $\pi(a_t \mid o_t, h_{0:t-1})$ that uses the agent history $h_{0:t-1}$ to obtain the probability of the action a_t. Thus, in order to operate effectively in a POMDPs, an agent must have memory mechanisms to retrieve a history $h_{0:t-1}$. Partial observability arises in a variety of real-world situations, including robotic navigation and manipulation tasks, autonomous vehicle tasks, and complex decision-making problems.

E Appendix – Meta Reinforcement Learning

In this section, we explore the concept of Meta-Reinforcement Learning (Meta-RL), a specialized domain within POMDPs that focuses on equipping agents with the ability to learn from their past experiences across multiple tasks. This capability is particularly crucial in dynamic environments where agents must adapt quickly to new challenges. By recognizing and memorizing common patterns and structures from previous interactions, agents can enhance their efficiency and effectiveness when facing unseen tasks.

Meta-RL is characterized by the principle of "*learning to learn*", where agents are trained not only to excel at specific tasks but also to generalize their knowledge and rapidly adjust to new tasks with minimal additional training. This adaptability is achieved through a structured approach that involves mapping data collected from various tasks to policies that guide the agent's behavior.

Meta-RL algorithm is a function f_θ parameterized with *meta-parameters* that maps the data $\mathcal{D}$, obtained during the process of training of RL agent in MDPs (tasks) $\mathcal{M}_i \sim p(\mathcal{M})$, to a policy $\pi_\phi : \phi = f_\theta(\mathcal{D})$. The process of learning the function f is typically referred to as the *outer-loop*, while the resulting function f is called the *inner-loop*. In this context, the parameters θ are associated with the outer-loop, while the parameters ϕ are associated with the inner-loop. Meta-training proceeds by sampling a task from the task distribution, running the inner-loop on it, and optimizing the inner-loop to improve the policies it produces. The interaction of the inner-loop with the task, during which the adaptation happens, is called a *lifetime* or a *trial*. In Meta-RL, it is common for $\mathcal{S}$ and $\mathcal{A}$ to be shared between all of the tasks and the tasks to only differ in the reward $\mathcal{R}(s, a)$ function, the dynamics $\mathcal{P}(s^{'} \mid s, a)$, and initial state distributions $P_0(s_0)$ [36].

F Appendix – Experiment Details

F.1 Appendix – Environments description

This section provides an extended description of the environments used in this work.

Passive-T-Maze [6]. In this T-shaped maze environment, the agent's goal is to move from the starting point to the junction and make the correct turn based on an initial signal. The agent can select from four possible actions: $a \in left, up, right, down$. The signal, denoted by the variable $clue$, is provided only at the beginning of the trajectory and indicates whether the agent should turn up ($clue = 1$) or down ($clue = -1$). The episode duration is constrained to $T = L + 1$, where L is the length of the corridor leading to the junction, which adds complexity to the task. To facilitate navigation, a binary variable called $flag$ is included in the observation vector. This variable equals 1 one step before reaching the junction and 0 at all other times, indicating the agent's proximity to the junction. Additionally, a noise channel introduces random integer values from the set $-1, 0, +1$ into the observation vector, further complicating the task. The observation vector is defined as $o = [y, clue, flag, noise]$, where y represents the vertical coordinate.

The agent receives a reward only at the end of the episode, which depends on whether it makes a correct turn at the junction. A correct turn yields a reward of 1, while an incorrect turn results in a reward of 0. This configuration differs from the conventional Passive T-Maze environment [6] by featuring distinct observations and reward structures, thereby presenting a more intricate set of conditions for the agent to navigate and learn within a defined time constraint. To transition from a sparse reward function to a dense reward function, the environment is parameterized using a penalty defined as $penalty = -\frac{1}{T-1}$, which imposes a penalty on the agent for each step taken within the environment. Thus, this environment has a 1D vector space of observations, a discrete action space, and sparse and dense configurations of the reward function.

Minigrid-Memory [39]. Minigrid-Memory is a two-dimensional grid-based environment specifically crafted to evaluate an agent's long-term memory and credit assignment capabilities. The layout consists of a T-shaped maze featuring a small room at the corridor's outset, which contains an object. The agent is instantiated at a random position within the corridor. Its objective is to navigate to the chamber, observe and memorize the object, then proceed to the junction at the maze's terminus and turn towards the direction where the object, identical to that in the initial chamber, is situated. A reward function defined as $r = 1 - 0.9 \times \frac{t}{T}$ is awarded upon successful completion, while failure results in a reward of zero. The episode concludes when the agent either makes a turn at a junction or exhausts a predefined time limit of 95 steps. To implement partial observability, observational constraints are imposed on the agent, limiting its view to a 3×3 frame size. Thus, this environment has a 2D space of image observations, a discrete action space, and sparse reward function.

F.2 Experimental Protocol

For each experiment, we conducted three runs of the agents with different initializations and performed validation during training using 100 random seeds ranging from 0 to 99. The results are presented as the mean success rate (or reward) ± the standard error of the mean (SEM).

Table 2: Online RL baselines hyperparameters used in the Minigrid-Memory and Passive T-Maze experiments.

Table 3: SAC-GPT-2

Hyperparameter	**Value**
Number of layers	2
Number of attention heads	2
Hidden dimension	256
Batch size	64
Optimizer	Adam
Learning rate	3e-4
Dropout	0.1
Replay buffer size	1e6
Discount (γ)	0.99
Entropy temperature	0.1

Table 4: DQN-GPT-2

Hyperparameter	**Value**
Number of layers	2
Number of attention heads	2
Hidden dimension	256
Batch size	64
Optimizer	Adam
Learning rate	3e-4
Dropout	0.1
Replay buffer size	1e6
Discount (γ)	0.99

Table 5: DTQN

Hyperparameter	**Value**
Number of layers	4
Number of attention heads	8
Hidden dimension	128
Batch size	32
Optimizer	Adam
Learning rate	3e-4
Dropout	0.1
Replay buffer size	5e5
Discount (γ)	0.99

Table 6: Offline RL baselines hyperparameters used for Decision Transformer and BC-LSTM in T-Maze experiments.

Table 7: Decision Transformer (DT)

Hyperparameter	**Value**
Number of layers	8
Number of attention heads	4
Hidden dimension (d_{model})	128
Feedforward dimension (d_{inner})	128
Head dimension (d_{head})	128
Context length (K)	$3T$
Dropout	0.0
DropAttention	0.0
Optimizer	AdamW
Learning rate	1e-4
Weight decay	0.1
Adam betas	(0.9, 0.999)
Batch size	64
Warmup steps	1000
Epochs	200

Table 8: BC-LSTM

Hyperparameter	**Value**
Number of layers	1
Hidden dimension (d_{model})	64
Bidirectional	False
Effective Context length (K_{eff})	$3T$
Dropout	0.0
Optimizer	AdamW
Learning rate	3e-4
Weight decay	0.01
Adam betas	(0.9, 0.999)
Batch size	64
Warmup steps	100
Epochs	100

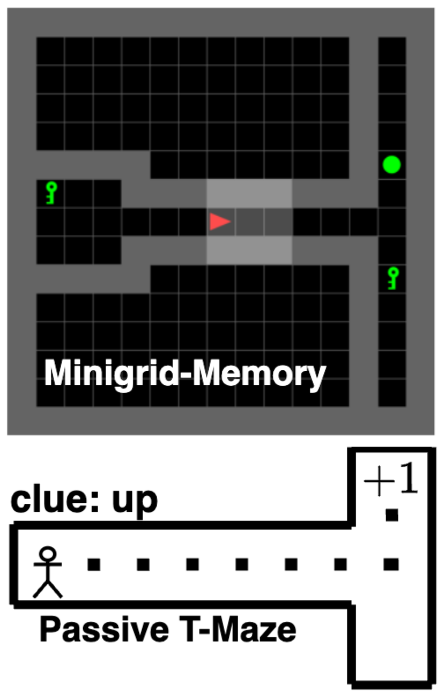


Figure 8: Memory-intensive environments for testing STM and LTM in Memory DM.