UNRAVELING THE COMPLEXITY OF MEMORY IN RL AGENTS: AN APPROACH FOR CLASSIFICATION AND EVALUATION

Anonymous authors
Paper under double-blind review

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.

1 INTRODUCTION

Reinforcement Learning (RL) effectively addresses problems within the Markov Decision Process (MDP) framework. However, applying RL to tasks with partial observability remains challenging, requiring agents to efficiently process their interaction history (Esslinger et al., 2022; Hausknecht & Stone, 2015; Ni et al., 2021).

In complex environments with noisy observations and long episodes, storing and retrieving important information becomes crucial (Goyal et al., 2022; Graves et al., 2016). Yet, the concept of *"memory"* in RL literature lacks unified definition. Some works define it as the ability to **handle dependencies within a fixed context** (Esslinger et al., 2022; Ni et al., 2023), others as the ability to **use out-of-context information** (Parisotto et al., 2020), and in Meta-RL, as the ability to **adapt to new environments** (Team et al., 2023).

However, in the absence of clear definitions and standardized evaluation protocols, claims about memory capacity in RL agents remain vague and often misleading. Memory is frequently attributed to architectural features like recurrence or attention, yet without proper isolation of memory effects, such assumptions can be incorrect. For instance, an agent might appear to exhibit long-term memory simply due to task configurations that allow shortcuts or overlap with short-term context. As a result, many empirical evaluations risk conflating different memory mechanisms or failing to detect architectural limitations. This hinders progress in developing truly memory-capable agents and comparing models in a fair and reproducible manner.

In this work, we attempt to unify and clarify the concept of memory in RL agents by treating memory as an intrinsic attribute of memory-enhanced agents, directly linking memory type classification to the agent's internal mechanisms. These specific memory types - short-term vs. long-term and declarative vs. procedural - can be rigorously assessed through experiments in memory-intensive environments. Our classification, based on temporal dependencies and the nature of the recalled information, provides a structured framework for distinguishing memory types, enabling fair comparisons, diagnosing architectural limitations, and guiding principled improvements.

It is important to clarify that our goal is not to replicate the full complexity of human memory. Rather, we selectively adapt well-established memory concepts from neuroscience - such as short-term, long-term, declarative, and procedural memory - that are already informally used in RL, but lack precise definitions and formal grounding (Fortunato et al., 2020; Kang et al., 2024b; Ni et al., 2023).

In summary, our main contributions are as follows:

1. We provide formal definitions of key memory types in RL – specifically, *short-term (STM) vs. long-term (LTM)* and *declarative vs. procedural* memory – grounded in neuroscience and formalized for RL settings – Section 4.
2. We introduce a task-level decoupling of *Memory Decision-Making (Memory DM)* and *Meta-RL*, clarifying the behavioral role of memory in each category – Section 4.
3. We propose a principled experimental methodology for evaluating STM and LTM in Memory DM tasks, including precise criteria for identifying memory boundaries – Section 4.2.
4. We show that neglecting the proposed methodology can mislead conclusions about agent memory capabilities, highlighting the importance of proper experimental configuration – Section 5.

2 Background

2.1 Memory of Humans and Agents

Many RL studies reference memory types from cognitive science, such as long-term (Lampinen et al., 2021; Ni et al., 2023), working (Graves et al., 2014), associative (Polson, 1975), and episodic memory (Pritzel et al., 2017), but often reduce them to vague temporal categories (e.g., short vs. long-term), with short-term spanning a few steps and long-term hundreds. This oversimplification, ignoring the relative nature of memory, complicates meaningful evaluation. To resolve this, we formalize agent memory types and introduce a principled evaluation framework.

2.1.1 Memory in Cognitive Science

Human adaptive behavior depends heavily on memory, which governs how knowledge and skills are acquired, retained, and reused (Parr et al., 2020; 2022). Memory exists in many forms, each of which relies on different neural mechanisms. Neuroscience and cognitive psychology distinguish memory by the temporal scales at which information is stored and accessed, and by the type of information that is stored. Abstracting from this distinction, a high-level definition of human memory is as follows: ***"memory – is the ability to retain information and recall it at a later time"***.

This definition aligns with how memory is typically understood in RL, and we adopt it to define types of RL agent memory. In neuroscience, memory is classified by timescale and behavioral function, distinguishing *short-term* memory, lasting seconds, from *long-term* memory, which can persist for a lifetime (Davis & Squire, 1984). It is also divided into *declarative* (explicit) and *procedural* (implicit) forms (Graf & Schacter, 1985): the former involves consciously recalled facts and events, while the latter relates to unconscious skills like riding a bike or skiing. Though these distinctions are well-established in neuroscience, RL requires precise, testable definitions. In what follows, we adapt these cognitive categories into a formal framework suitable for RL agents.

2.1.2 Memory in RL

Memory in RL encompasses diverse agent capabilities, but its definition varies across studies. In many POMDPs, agents must retain key information to act effectively later within the same environment. This typically involves two kinds of temporal dependencies: 1) within a bounded time window (e.g., transformer context (Esslinger et al., 2022; Ni et al., 2023; Yue et al., 2024)); 2) beyond the current context, requiring persistent storage or recall (Parisotto et al., 2020; Sorokin et al., 2022). As noted in Section 2.1.1, STM and LTM correspond to different temporal scopes of declarative memory (see Figure 1). In contrast, Meta-RL involves procedural memory, enabling agents to reuse skills across tasks (Team et al., 2023) (see Figure 2). However, many works conflate these types, evaluating "long-term memory" solely in Meta-RL settings based on MDPs (Kang et al., 2024a), without isolating decision-making from past information. To resolve this, we formalize RL memory types by task structure and temporal dependencies. In this work, we focus on **declarative memory**, guiding decisions from past observations in the same environment, emphasizing its short- and long-term forms.

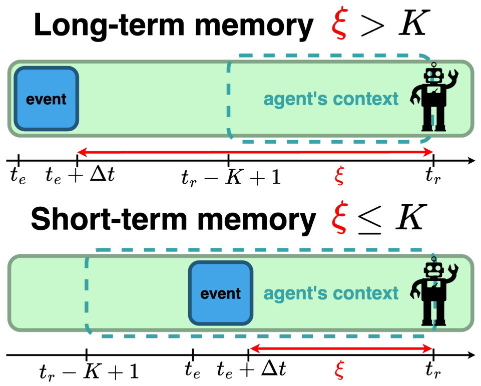


Figure 1: STM vs. LTM. t_e - event start, t_r - recall time; K - context length, ξ – correlation horizon. If the event lies beyond K, LTM is needed; if within, STM is enough.

2.1.3 Memory and Credit Assignment

Papers on agent memory, especially declarative memory, often distinguish between two forms of temporal reasoning: *memory* and *credit assignment* (Mesnard et al., 2020; Ni et al., 2023; Osband et al., 2019). In Ni et al. (2023), *memory* is defined as recalling a past event at the current time, while *credit assignment* is identifying when reward-relevant actions occurred. Though distinct, both concepts describe temporal dependencies between events. Here, we focus on the agent's ability to form such dependencies, treating memory and credit assignment as one. We adopt the general definition from Section 2.1.1, which applies to both, as it captures their shared temporal nature.

3 Related Works

Interest in memory-enhanced RL has led to numerous architectures (Oh et al., 2016; Lampinen et al., 2021; Fortunato et al., 2020) and benchmarks (Morad et al., 2023a; Cherepanov et al., 2025; Osband et al., 2019; Pleines et al., 2023), yet the term "memory" remains inconsistently defined and often misaligned with what experiments actually test. Some works define memory as retaining recent observations within the same episode – either via recurrent states (Hausknecht & Stone, 2015), transformer contexts (Esslinger et al., 2022; Grigsby et al., 2024), or external stores (Lampinen et al., 2021; Le et al., 2024). Others expand it to long-range dependencies through learned state compression (Morad et al., 2023b), key-value recurrent updates (Pramanik et al., 2023; Cherepanov et al., 2024), or spatial memory maps (Parisotto & Salakhutdinov, 2017b). A separate line views memory as cross-episode knowledge transfer, e.g., in Meta-RL (Kang et al., 2024a; Bauer et al., 2023). This variety – from within-episode recall to multi-task adaptation – reflects the lack of a shared definition. Our work addresses this gap by introducing a unified taxonomy grounded in temporal dependencies and task structure.

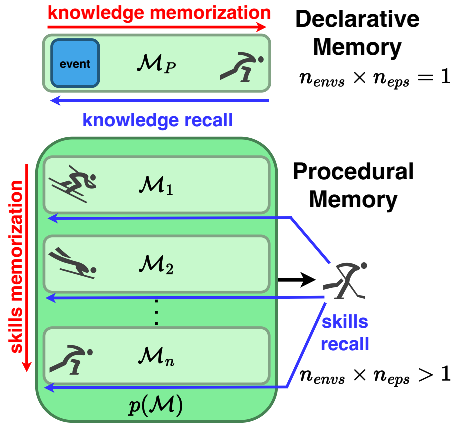


Figure 2: Illustration of declarative and procedural memory. Red arrows represent memorization steps, blue arrows indicate the recall of task-relevant information.

Among concrete instantiations, Esslinger et al. (2022) introduced Deep Transformer Q-Networks (DTQN), which leverage transformer context for partially observable RL. Ni et al. (2023) extended this line with GPT-2 based agents, including DQN-GPT-2 and SAC-GPT-2, which apply attention-based memory to online RL control. In the offline setting, Decision Transformer (DT) (Chen et al., 2021) uses a sequence model trained on trajectories for return-conditioned planning, while recurrent baselines such as BC-LSTM provide a contrast between attention-based and recurrent memory mechanisms. These models constitute the baselines we evaluate in our experiments.

Ni et al. (2023) further distinguish between memory – the ability to recall past events – and credit assignment – identifying when reward-relevant actions occurred. Kang et al. (2024b) build on reconstructive memory (Bartlett & Kintsch, 1995), emphasizing reflection grounded in interaction.

These varied interpretations underscore the need for a unified definition of memory in RL. We address this by formalizing memory types via temporal dependencies and task structure, and proposing a framework for empirical evaluation. Concurrently with our study, Yue et al. (2024) introduced memory dependency pairs (p, q) to model recall in demonstrations. While insightful for imitation learning, their approach lacks a theoretical treatment of RL memory and does not address the broader taxonomy of agent memory types.

4 Memory in RL

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.*

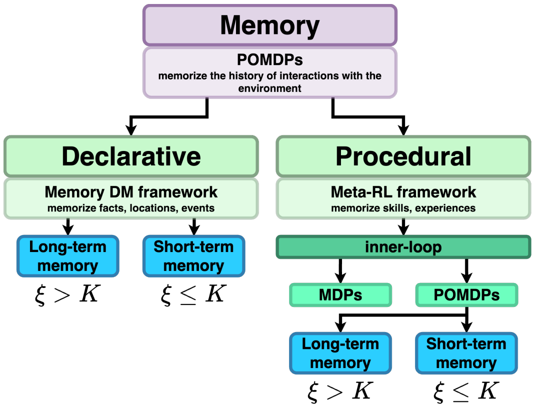


Figure 3: Classification of memory types of RL agents. While the Memory DM framework contrasts with Meta-RL, its formalism can also describe inner-loop tasks when they are POMDPs.

In 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: $J^\theta = \mathbb{E}_{\mathcal{M}_i \sim p(\mathcal{M})}\left[\mathbb{E}_{\mathcal{D}}\left[\sum\limits_{\tau \in \mathcal{D}_{I:H}} G_i(\tau) \middle| f_\theta, \mathcal{M}_i\right]\right]$, 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 (Beck et al., 2024).*

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 (see Figure 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 Figure 1.

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 zero-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 STM or LTM (Figure 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 from interaction histories in the inner-loop, and within the inner-loop can be considered a Memory DM. In the blue case, memory is needed only for skill transfer. While this paper focuses on Memory DM tasks, the terminology enables further classification of Meta-RL tasks, with POMDP sub-classes highlighted in green. The proposed classification of tasks requiring agent memory is shown in Table 1.

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 (Bellemare et al., 2013) with frame stacking or MuJoCo control tasks (Fu et al., 2021) 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 $\max\limits_n \xi_n = 1$. Conversely, if $\max\limits_n \xi_n = 1$, then no decision depends on events beyond the current step, satisfying the Markov property. ∎

Table 1: Classification of tasks requiring agent memory based on our definitions: green indicates tasks described by the proposed definitions of LTM and STM, while blue indicates those that are not. Meta-RL tasks with a POMDP inner-loop are marked green as they can be classified as Memory DM tasks. POMDP† indicates a Memory DM task considered 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 zero-shot	Multi-task zero-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 zero-shot	Multi-task zero-shot
>1	>1	Meta-RL	MDP	Proc.	Multi-task	Multi-task

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 Theorem 2:

Theorem 2 (**On the context memory border**). *Let $\tilde{\mathcal{M}}_P$ be a memory-intensive environment and K be an agent's 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 $\overline{K} = \min\limits_n \Xi - 1$. 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 $K \leq \min\limits_n \Xi - 1$ generates the LTM problem exclusively. Since context length cannot be negative or zero, it turns out that $1 \leq K \leq \overline{K} = \min\limits_n \Xi - 1$, 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

$$\overline{K} = \min_n \Xi - 1 = \min_n \Big\{(t_r - t_e - \Delta t + 1)_n\Big\}_n - 1. \tag{4}$$

Using Theorem 2, we can establish the necessary conditions for validating short-term memory: **1) Weak condition to validate short-term memory**: if $\overline{K} < K < \max\limits_n \Xi$, then the memory-intensive

Algorithm 1 Algorithm for setting up an experiment to test long-term and short-term memory in Memory DM framework.

```
  Require: $\tilde{\mathcal{M}}_P$ – memory-intensive environment; $\mu(K)$ – memory mechanism.
1. Estimate the number of $n$ event-recall pairs in the environment (Definition 4.6).
    1. $n = 0 \rightarrow$ Environment is not suitable for testing long-term and short-term memory.
    2. $n \geq 1 \rightarrow$ Environment is suitable for testing long-term and short-term memory.
2. Estimate context memory border $\overline{K}$ (4).
    1. $\forall$ event-recall pair $(\beta(\alpha), \alpha)_i$ find corresponding $\xi_i, i \in [1..n]$.
    2. Determine $\overline{K}$ as $\overline{K} = \min \Xi - 1 = \min_n \{\xi_n\}_n - 1 = \min_n \left\{(t_r - t_e - \Delta t + 1)_n\right\}_n - 1$
3. Conduct an appropriate experiment (Definition 4.5).
    1. To test short-term memory set $K > \overline{K}$.
    2. To test long-term memory set $K \leq \overline{K} \leq K_{eff} = \mu(K)$.
4. Analyze the results.
```

environment $\tilde{M}_P$ is used to validate both short-term and long-term memory. **2) Strong condition to validate short-term memory**: if $\max_n \Xi < K$, 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 $\overline{K} < K < \max_n \Xi \leq T - 1$, 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 $K \geq \max_n \Xi$, 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) $K \in (\overline{K}, \max_n \Xi) \Rightarrow$ **validating both STM and LTM**; (iii) $K \in [\max_n \Xi, \infty) \Rightarrow$ **validating STM only**.

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 (Esslinger et al., 2022; Grigsby et al., 2024; Ni et al., 2023). 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** $\mu(K) : \mathbb{N} \rightarrow \mathbb{N}$ 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.*

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

Memory mechanisms are key to LTM tasks by recalling out-of-context information 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 Hausknecht & Stone (2015)), 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:**

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

According to our definitions, agents with memory mechanisms in the Memory DM framework that solve LTM tasks can also handle STM tasks, but not vice versa. The algorithm for setting up experiments to test an agent's STM or LTM is outlined in Algorithm 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 Table 2), including *Passive T-Maze* (Ni et al., 2023), *Minigrid-Memory* (Chevalier-Boisvert et al., 2023), *ViZDoom-Two-Colors* (Sorokin et al., 2022), *Memory Maze* (Pasukonis et al., 2022), *Memory Cards* (Esslinger et al., 2022), *Mortar Mayhem* and *Mystery Path* (Pleines et al., 2025), *POPGym–Autoencode* and *POPGym-RepeatPrevious* (Morad et al., 2023a).

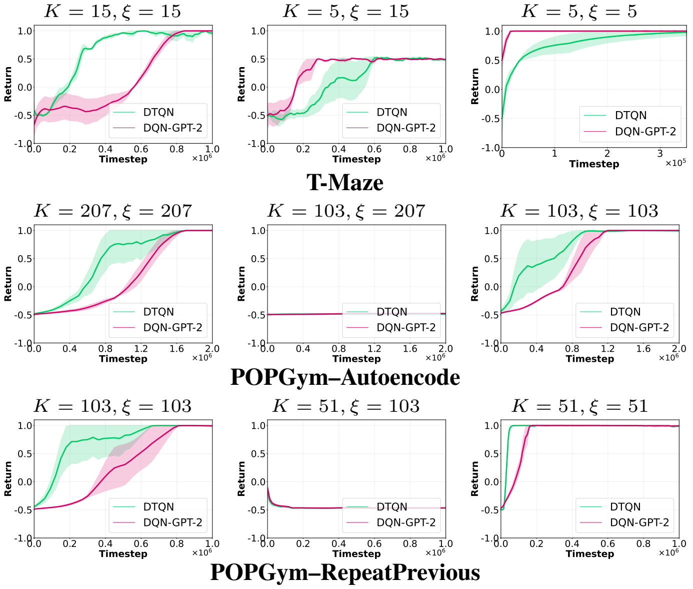


Figure 4: Performance of Online RL agents "with memory" across different memory configurations. Each row shows a specific environment: T-Maze, POPGym-Autoencode, and POPGym-RepeatPrevious, with varying agent context length K and correlation horizons ξ. The STM ⇆ LTM transitions reflect the relative nature of the settings to test memory, depending on both agent and environment parameters.

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 $T = L+1$, where L is the corridor length. 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 using the Memory DM framework to distinguish short- vs. long-term memory. Our experiments stress the need for proper methodology (Algorithm 1) and show how poor setups can misrepresent memory use. We test four memory-intensive tasks: Passive T-Maze and Minigrid-Memory (cue recall), and POPGym–Autoencode and RepeatPrevious (observation reconstruction and action repeat), all requiring recall over time. In the online setting, we evaluate DTQN (Esslinger et al., 2022), DQN-GPT-2, and SAC-GPT-2 (Ni et al., 2023) with attention-based memory. Offline, we test DT (Chen et al., 2021) and BC-LSTM to compare attention vs. recurrence. In all cases, we vary agent context K and task horizon ξ to isolate memory types and reveal model limitations.

5.1 Pitfalls of Naive Memory Tests

Proper evaluation of memory in RL agents requires distinguishing STM from LTM by accounting for correlation horizons ξ. Without this, STM and LTM effects blur, misrepresenting agent capacity. We illustrate this with SAC-GPT-2 in Minigrid-Memory under two setups: (i) fixed $L = 21$ ($\xi = 22$), and (ii) variable L ($\xi \in [7, 22]$), testing STM ($K = 22$) and LTM ($K = 14$) settings. As shown in Figure 5, the *variable* setup gives high success for both settings, implying good memory.

But in the *fixed* case, LTM fails, revealing the agent's true limit. Mixed-horizon tasks can hide LTM deficits - only fixed $\xi > K$ setups expose them. Proper LTM evaluation requires controlling the correlation horizon ξ relative to the agent's context K. Without this, STM effects may dominate and misrepresent the agent's memory type. Our methodology provides a principled way to avoid this confusion.

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

Task	$\boldsymbol{\Xi}$	$\boldsymbol{\xi}$	**LTM if** $K <$
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

5.2 The Relative Nature of an Agent's Memory

According to Algorithm 1, the experimental setup for testing agent memory types (LTM and STM) depends on two parameters: the agent's context length K and the context memory border $\overline{K}$, which in turn is determined by the environment's correlation horizon ξ. Verifying LTM or STM requires adjusting K or ξ while keeping the other fixed. This section outlines how these parameters interact in memory testing. An agent's memory cannot be defined in isolation – it arises from the interplay between its context K and the environment's horizons ξ. Thus, the same agent may exhibit either STM or LTM behavior depending on the task setup.

We test DTQN and DQN-GPT-2 in three memory-intensive tasks: Passive T-Maze, POPGym Autoencode, and RepeatPrevious – by varying K and ξ to simulate STM ($\xi \leq K$) and LTM ($\xi > K$); as shown in Figure 4, performance is high when $\xi \leq K$ but drops sharply for $\xi > K$, confirming that long-range dependencies require explicit memory mechanisms. These results confirm memory is relative: LTM depends on both temporal distance and agent design. Without controlling K and ξ, memory claims are unreliable. Our K-ξ framework ensures consistent, interpretable evaluation.

5.3 Generalization Across Sequence Lengths

Evaluating memory in RL agents requires distinguishing true long-term memory (LTM) from memorization within fixed context. To demonstrate this, we test DT and BC-LSTM on T-Maze: both are trained on specific corridor lengths and evaluated on both seen and longer, unseen ones. This setup tests whether agents can recall cue information beyond training range - an LTM indicator. Figure 6 shows success heatmaps across training and validation lengths. The diagonal indicates in-distribution performance; extrapolation lies to the right.

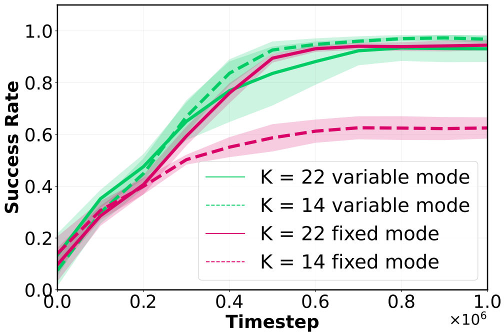
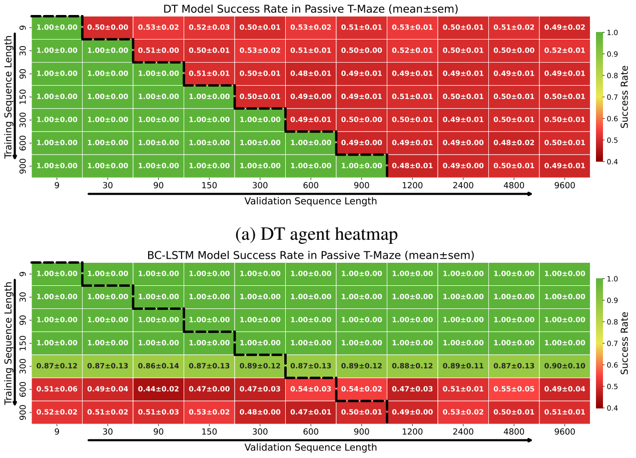


Figure 5: SAC-GPT-2 in Minigrid-Memory ($L = 21$) with short- ($K = 22$) and long-term ($K = 14$) memory setups. Variable mode (green) masks memory limits; fixed mode (red) reveals failure at $K = 14$, demonstrating lack of long-term memory – made evident by our evaluation method.

While both models process sequences and are labeled as memory-enhanced, our framework reveals key differences. DT relies on a fixed attention window and operates with short-term memory, while LSTM uses a recurrent state, enabling true LTM. T-Maze results expose this gap: DT performs well when validation lengths stay within context but fails for $L > 90$, whereas BC-LSTM generalizes to much longer sequences, demonstrating effective LTM. If evaluated only on shorter validation lengths, DT may appear stronger, masking memory limitations. Both perform well for lengths ≤ 150, but at training length 300, DT scores 100%, while BC-LSTM drops to 0.87. For longer training (600, 900), BC-LSTM collapses, DT remains high – misleadingly favoring STM.

DT succeeds only when the correlation horizon satisfies $\xi \leq K$, but fails whenever $\xi > K$, which demonstrates that DT lacks LTM. In contrast, BC-LSTM, on the horizons where it trains reliably, can exploit dependencies beyond the training range and therefore exhibits LTM within those regimes. Our framework makes this distinction precise by separating architectural limits from memory function, ensuring that DT is correctly identified as an STM agent, while BC-LSTM is capable of LTM on the ranges where it learns effectively (i.e. without vanishing gradients problem (Trinh et al., 2018)).



(a) DT agent heatmap

(b) BC-LSTM agent heatmap

Figure 6: 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.

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 differences between architectures: transformers like DTQN or DT rely mainly on short-term memory, while recurrent models such as BC-LSTM exhibit long-term memory. Experiments on T-Maze, MiniGrid, and POPGym confirm the need for proper setups to avoid misleading conclusions. The framework clarifies how memory mechanisms shape behavior and could be extended to include additional systems from cognitive science, such as working or episodic memory, and to explore whether new types emerge in complex RL tasks. As a direction for future work, it would be valuable to study adaptive dynamic updating of memory representations, since most existing work focuses primarily on memorization and retention rather than on how agents revise stored information over time.

Reproducibility Statement

We have taken several measures to ensure the reproducibility of our results. **Model details:** The formalization of our framework – covering Memory DM, STM/LTM, and the correlation horizon – is provided in Section 4, with precise definitions in Definition 4.1, Definition 4.2, and Definition 4.5, and the experimental configuration procedure in Algorithm 1. **Theoretical results:** Assumptions and complete statements (including the definition of memory-intensive environments) are given in Definition 4.6, and key results with proofs appear in Theorem 2 and its accompanying discussion. **Experimental setup:** Tasks, training procedures, and evaluation protocols are reported in Section 5, with validation protocol details in Section F.1, hyperparameters in Section F.1, and environment descriptions in Appendix F. **Baselines:** Baseline selections and configurations are documented in Section 5, with their hyperparameters listed in Section F.1. **Code and data:** An anonymous repository containing source code, training scripts, and configuration files submitted as supplementary material. Together, these resources allow for full replication of our theoretical analyses and empirical results.

References

Frederic C. Bartlett and Walter Kintsch. *Remembering: A Study in Experimental and Social Psychology*. Cambridge University Press, 2 edition, 1995.

Jakob Bauer, Kate Baumli, Feryal Behbahani, Avishkar Bhoopchand, Nathalie Bradley-Schmieg, Michael Chang, Natalie Clay, Adrian Collister, Vibhavari Dasagi, Lucy Gonzalez, Karol Gregor, Edward Hughes, Sheleem Kashem, Maria Loks-Thompson, Hannah Openshaw, Jack Parker-Holder, Shreya Pathak, Nicolas Perez-Nieves, Nemanja Rakicevic, Tim Rocktäschel, Yannick Schroecker, Satinder Singh, Jakub Sygnowski, Karl Tuyls, Sarah York, Alexander Zacherl, and Lei M Zhang. Human-timescale adaptation in an open-ended task space. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 1887–1935. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/bauer23a.html.

Jacob Beck, Risto Vuorio, Evan Zheran Liu, Zheng Xiong, Luisa Zintgraf, Chelsea Finn, and Shimon Whiteson. A survey of meta-reinforcement learning, 2024. URL https://arxiv.org/abs/2301.08028.

Philipp Becker, Niklas Freymuth, and Gerhard Neumann. Kalmamba: Towards efficient probabilistic state space models for rl under uncertainty, 2024. URL https://arxiv.org/abs/2406.15131.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47: 253–279, 2013.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Egor Cherepanov, Alexey Staroverov, Dmitry Yudin, Alexey K. Kovalev, and Aleksandr I. Panov. Recurrent action transformer with memory. *arXiv preprint arXiv:2306.09459*, 2024. URL https://arxiv.org/abs/2306.09459.

Egor Cherepanov, Nikita Kachaev, Alexey Kovalev, and Aleksandr 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.

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. *CoRR*, abs/2306.13831, 2023.

Hasker Davis and Larry 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.

Ben Deverett, Ryan Faulkner, Meire Fortunato, Gregory Wayne, and Joel Z Leibo. Interval timing in deep reinforcement learning agents. *Advances in Neural Information Processing Systems*, 32, 2019.

Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning, 2016. URL https://arxiv.org/abs/1611.02779.

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

Meire Fortunato, Melissa Tan, Ryan Faulkner, Steven Hansen, Adrià Puigdomènech Badia, Gavin Buttimore, Charlie Deck, Joel Z Leibo, and Charles Blundell. Generalization of reinforcement learners with working and episodic memory, 2020. URL https://arxiv.org/abs/1910.13406.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2021.

Anirudh Goyal, Abram L. Friesen, Andrea Banino, Theophane Weber, Nan Rosemary Ke, Adria Puigdomenech Badia, Arthur Guez, Mehdi Mirza, Peter C. Humphreys, Ksenia Konyushkova, Laurent Sifre, Michal Valko, Simon Osindero, Timothy Lillicrap, Nicolas Heess, and Charles Blundell. Retrieval-augmented reinforcement learning, 2022. URL https://arxiv.org/abs/2202.08417.

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

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines, 2014. URL https://arxiv.org/abs/1410.5401.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez, Edward Grefenstette, Tiago Ramalho, John Agapiou, Adrià Badia, Karl Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, Christopher Summerfield, Phil Blunsom, Koray Kavukcuoglu, and Demis Hassabis. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538, 10 2016. doi: 10.1038/nature20101.

Jake Grigsby, Linxi Fan, and Yuke Zhu. Amago: Scalable in-context reinforcement learning for adaptive agents, 2024. URL https://arxiv.org/abs/2310.09971.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution, 2018. URL https://arxiv.org/abs/1809.01999.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2555–2565. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/hafner19a.html.

Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps, 2015.

Jikun Kang, Romain Laroche, Xingdi Yuan, Adam Trischler, Xue Liu, and Jie Fu. Think before you act: Decision transformers with working memory, 2024a. URL https://arxiv.org/abs/2305.16338.

Yongxin Kang, Enmin Zhao, Yifan Zang, Lijuan Li, Kai Li, Pin Tao, and Junliang Xing. Sample efficient reinforcement learning using graph-based memory reconstruction. *IEEE Transactions on Artificial Intelligence*, 5(2):751–762, 2024b. doi: 10.1109/TAI.2023.3268612.

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.

Hung Le, Kien Do, Dung Nguyen, Sunil Gupta, and Svetha Venkatesh. Stable hadamard memory: Revitalizing memory-augmented agents for reinforcement learning. *arXiv preprint arXiv:2410.10132*, 2024.

Chris Lu, Yannick Schroecker, Albert Gu, Emilio Parisotto, Jakob Foerster, Satinder Singh, and Feryal Behbahani. Structured state space models for in-context reinforcement learning, 2023. URL https://arxiv.org/abs/2303.03982.

Luckeciano C. Melo. Transformers are meta-reinforcement learners, 2022. URL https://arxiv.org/abs/2206.06614.

Thomas Mesnard, Théophane Weber, Fabio Viola, Shantanu Thakoor, Alaa Saade, Anna Harutyunyan, Will Dabney, Tom Stepleton, Nicolas Heess, Arthur Guez, et al. Counterfactual credit assignment in model-free reinforcement learning. *arXiv preprint arXiv:2011.09464*, 2020.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner, 2018. URL https://arxiv.org/abs/1707.03141.

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. *Advances in Neural Information Processing Systems*, 36:72008–72029, 2023b.

Steven D. Morad, Stephan Liwicki, Ryan Kortvelesy, Roberto Mecca, and Amanda Prorok. Graph convolutional memory using topological priors, 2021. URL https://arxiv.org/abs/2106.14117.

Tianwei Ni, Benjamin Eysenbach, and Ruslan Salakhutdinov. Recurrent model-free rl can be a strong baseline for many pomdps. *arXiv preprint arXiv:2110.05038*, 2021.

Tianwei Ni, Michel Ma, Benjamin Eysenbach, and Pierre-Luc 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.

Junhyuk Oh, Valliappa Chockalingam, Satinder Singh, and Honglak Lee. Control of memory, active perception, and action in minecraft, 2016. URL https://arxiv.org/abs/1605.09128.

Ian Osband, Yotam Doron, Matteo Hessel, John Aslanides, Eren Sezener, Andre Saraiva, Katrina McKinney, Tor Lattimore, Csaba Szepesvari, Satinder Singh, et al. Behaviour suite for reinforcement learning. *arXiv preprint arXiv:1908.03568*, 2019.

Emilio Parisotto and Ruslan Salakhutdinov. Neural map: Structured memory for deep reinforcement learning, 2017a. URL https://arxiv.org/abs/1702.08360.

Emilio Parisotto and Ruslan Salakhutdinov. Neural map: Structured memory for deep reinforcement learning. *arXiv preprint arXiv:1702.08360*, 2017b.

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.

Thomas Parr, Rajeev Vijay Rikhye, Michael M Halassa, and Karl J Friston. Prefrontal computation as active inference. *Cerebral Cortex*, 30(2):682–695, 2020.

Thomas Parr, Giovanni Pezzulo, and Karl J Friston. *Active inference: the free energy principle in mind, brain, and behavior*. MIT Press, 2022.

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. Memory gym: Partially observable challenges to memory-based agents in endless episodes. *arXiv preprint arXiv:2309.17207*, 2023.

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.

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

Subhojeet Pramanik, Esraa Elelimy, Marlos C Machado, and Adam White. Recurrent linear transformers. *arXiv preprint arXiv:2310.15719*, 2023.

Alexander Pritzel, Benigno Uria, Sriram Srinivasan, Adrià Puigdomènech, Oriol Vinyals, Demis Hassabis, Daan Wierstra, and Charles Blundell. Neural episodic control, 2017. URL https://arxiv.org/abs/1703.01988.

Jan Robine, Marc Höftmann, Tobias Uelwer, and Stefan 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.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986. URL https://api.semanticscholar.org/CorpusID:205001834.

Mohammad Reza Samsami, Artem Zholus, Janarthanan Rajendran, and Sarath Chandar. Mastering memory tasks with world models, 2024. URL https://arxiv.org/abs/2403.04253.

Gresa Shala, André Biedenkapp, and Josif Grabocka. Hierarchical transformers are efficient meta-reinforcement learners, 2024. URL https://arxiv.org/abs/2402.06402.

Jimmy T. H. Smith, Andrew Warrington, and Scott W. Linderman. Simplified state space layers for sequence modeling, 2023. URL https://arxiv.org/abs/2208.04933.

Doo Re Song, Chuanyu Yang, Christopher McGreavy, and Zhibin Li. Recurrent deterministic policy gradient method for bipedal locomotion on rough terrain challenge, November 2018. URL http://dx.doi.org/10.1109/ICARCV.2018.8581309.

Artyom Sorokin, Nazar Buzun, Leonid Pugachev, and Mikhail Burtsev. Explain my surprise: Learning efficient long-term memory by predicting uncertain outcomes. *Advances in Neural Information Processing Systems*, 35:36875–36888, 2022.

Ivan Sorokin, Alexey Seleznev, Mikhail Pavlov, Aleksandr Fedorov, and Anastasiia Ignateva. Deep attention recurrent q-network, 2015. URL https://arxiv.org/abs/1512.01693.

Adaptive Agent Team, Jakob Bauer, Kate Baumli, Satinder Baveja, Feryal Behbahani, Avishkar Bhoopchand, Nathalie Bradley-Schmieg, Michael Chang, Natalie Clay, Adrian Collister, Vibhavari Dasagi, Lucy Gonzalez, Karol Gregor, Edward Hughes, Sheleem Kashem, Maria Loks-Thompson, Hannah Openshaw, Jack Parker-Holder, Shreya Pathak, Nicolas Perez-Nieves, Nemanja Rakicevic, Tim Rocktäschel, Yannick Schroecker, Jakub Sygnowski, Karl Tuyls, Sarah York, Alexander Zacherl, and Lei Zhang. Human-timescale adaptation in an open-ended task space, 2023. URL https://arxiv.org/abs/2301.07608.

Trieu Trinh, Andrew Dai, Thang Luong, and Quoc Le. Learning longer-term dependencies in rnns with auxiliary losses. In *International Conference on Machine Learning*, pp. 4965–4974. PMLR, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Daan Wierstra, Alexander Förster, Jan Peters, and Jürgen Schmidhuber. Recurrent policy gradients. *Logic Journal of the IGPL*, 18:620–634, 10 2010. doi: 10.1093/jigpal/jzp049.

William Yue, Bo Liu, and Peter Stone. Learning memory mechanisms for decision making through demonstrations. *arXiv preprint arXiv:2411.07954*, 2024.

Tony Duan YuXuan Liu and Wesley Hsieh. Temporal convolutional policy networks, 2016. URL https://yuxuanliu.com/files/tcpn.pdf.

Wojciech Zaremba and Ilya Sutskever. Reinforcement learning neural turing machines - revised, 2016. URL https://arxiv.org/abs/1505.00521.

Deyao Zhu, Li Erran Li, and Mohamed Elhoseiny. Value memory graph: A graph-structured world model for offline reinforcement learning, 2023. URL https://arxiv.org/abs/2206.04384.

Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: A very good method for bayes-adaptive deep rl via meta-learning, 2020. URL https://arxiv.org/abs/1910.08348.

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) (Rumelhart et al., 1986), which are capable of handling sequential dependencies by maintaining a hidden state that captures information about previous time steps (Wierstra et al., 2010; Hausknecht & Stone, 2015; Sorokin et al., 2015; Duan et al., 2016; Song et al., 2018; Zintgraf et al., 2020) (pure LTM, according to our taxonomy). Another popular way to implement memory is to use Transformers (Vaswani et al., 2017), which use self-attention mechanisms to capture dependencies inside the context window (Parisotto et al., 2020; Lampinen et al., 2021; Esslinger et al., 2022; Melo, 2022; Team et al., 2023; Pramanik et al., 2023; Robine et al., 2023; Ni et al., 2023; Grigsby et al., 2024; Shala et al., 2024) (STM in case of classical transformers without additional memory mechanisms or LTM if we use recurrent memory, activation caching, etc.). State-space models (SSMs) (Gu et al., 2021; Smith et al., 2023; Gu & Dao, 2023) combine the strengths of RNNs and Transformers and can also serve to implement memory through preservation of system state (Hafner et al., 2019; Lu et al., 2023; Becker et al., 2024; Samsami et al., 2024) (LTM, according to our taxonomy). 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 (YuXuan Liu & Hsieh, 2016; Mishra et al., 2018) (STM, since memory is represented as a fixed-size temporal convolution, analogous to an attention window). A world model (Ha & Schmidhuber, 2018) 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 (Morad et al., 2021; Zhu et al., 2023; Kang et al., 2024b).

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) (Oh et al., 2016; Lampinen et al., 2021; Goyal et al., 2022; Cherepanov et al., 2024) and read/write access (Graves et al., 2016; Zaremba & Sutskever, 2016; Parisotto & Salakhutdinov, 2017a).

Memory can also be implemented without architectural mechanisms, relying instead on agent policy. For instance, in the work of Deverett et al. (2019), 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 $P_0(s_0)$ (Beck et al., 2024).

F Appendix – Experiment Details

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

Passive-T-Maze (Ni et al., 2023). 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 (Ni et al., 2023) 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 (Chevalier-Boisvert et al., 2023). 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.1 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 3: Online RL baselines hyperparameters used in the Minigrid-Memory and Passive T-Maze experiments.

Table 4: 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 5: 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 6: 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 7: Offline RL baselines hyperparameters used for Decision Transformer and BC-LSTM in T-Maze experiments.

Table 8: 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 9: 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