MEMORY, BENCHMARK & ROBOTS: A BENCHMARK FOR SOLVING COMPLEX TASKS WITH REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Memory is crucial for enabling agents to tackle complex tasks with temporal and spatial dependencies. While many reinforcement learning (RL) algorithms incorporate memory, the field lacks a universal benchmark to assess an agent's memory capabilities across diverse scenarios. This gap is particularly evident in tabletop robotic manipulation, where memory is essential for solving tasks with partial observability and ensuring robust performance, yet no standardized benchmarks exist. In this work, we address these challenges through three key contributions: (1) we propose a comprehensive classification framework for memory-intensive RL tasks, (2) we collect **MIKASA** – a unified benchmark that enables systematic evaluation of memory-enhanced agents across diverse scenarios, and (3) we develop **ManiSkill-Memory** – a novel benchmark of 32 carefully designed memory-intensive tasks that assess memory capabilities in tabletop robotic manipulation. Our contributions establish a unified framework for advancing memory RL research, driving the development of more reliable systems for real-world applications.

1 INTRODUCTION

Many real-world problems involve partial observability (Kaelbling et al., 1998), where an agent lacks full access to the environment's state. These tasks often include sequential decision-making (Chen et al., 2021), delayed or sparse rewards, and long-term information retention (Parisotto et al., 2020; Lampinen et al., 2021). One approach to tackling these challenges is to equip the agent with memory, allowing it to utilize historical information (Meng et al., 2021; Ni et al., 2021).

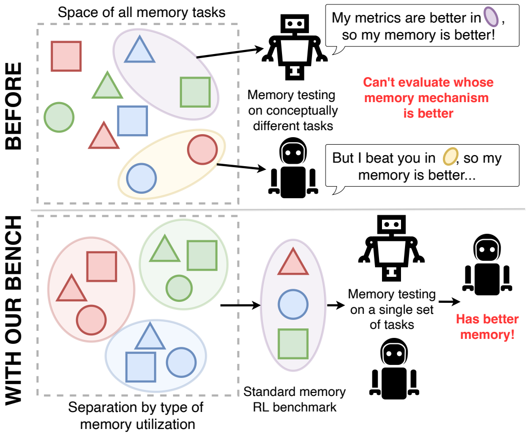
Space of all memory tasks
BEFORE
My metrics are better in , so my memory is better!
Memory testing on conceptually different tasks
Can't evaluate whose memory mechanism is better
But I beat you in , so my memory is better...
WITH OUR BENCH
Memory testing on a single set of tasks
Has better memory!
Standard memory RL benchmark
Separation by type of memory utilization

Figure 1: Systematic classification of problems with memory in RL reveals distinct memory utilization patterns and enables objective evaluation of memory mechanisms across different agents.

While there are well-established benchmarks in Natural Language Processing (Bai et al., 2023; An et al., 2023), the evaluation of memory in reinforcement learning (RL) remains fragmented. Existing benchmarks, such as POPGym (Morad et al., 2023), DMLab-30 (Hung et al., 2018) and MemoryGym (Pleines et al., 2023), focus on specific aspects of memory utilization, as they are designed around particular problem domains.

In contrast to classical RL, where benchmarks like Atari (Bellemare et al., 2013) and MuJoCo (Todorov et al., 2012) serve as universal standards, memory-enhanced agents are typically evaluated on custom environments developed alongside their proposals Table 2. This fragmented evaluation landscape obscures important performance variations across different memory tasks. For instance, an agent might excel at maintaining object attributes over extended periods while struggling

Table 1: **ManiSkill-Memory**: a benchmark of 32 memory-intensive tasks for robotic manipulation, organized into 12 categories with varying difficulty levels. See Appendix E for details.

Memory Task	Mode	Brief description of the task	T	Oracle Info	Prompt	Memory Task Type
ShellGame[Mode]-v0	Touch Push Pick	Memorize the position of the ball after some time being covered by the cups and then interact with the cup the ball is under.	90	cup_with_ball_number	—	Object
Intercept[Mode]-v0	Slow Medium Fast	Memorize the positions of the rolling ball, estimate its velocity through those positions, and then aim the ball at the target.	90	initial_velocity	—	Spatial
InterceptGrab[Mode]-v0	Slow Medium Fast	Memorize the positions of the rolling ball, estimate its velocity through those positions, and then catch the ball with the gripper and lift it up.	90	initial_velocity	—	Spatial
RotateLenient[Mode]-v0	Pos PosNeg	Memorize the initial position of the peg and rotate it by a given angle.	90	y_angle_diff	target_angle	Spatial
RotateStrict[Mode]-v0	Pos PosNeg	Memorize the initial position of the peg and rotate it to a given angle without shifting its center.	90	y_angle_diff	target_angle	Object
TakeItBack-v0	—	Memorize the initial position of the cube, move it to the target region, and then return it to its initial position.	180	xyz_initial	—	Spatial
RememberColor[Mode]-v0	3 \ 5 \ 9	Memorize the color of the cube and choose among other colors.	60	true_color_indices	—	Object
RememberShape[Mode]-v0	3 \ 5 \ 9	Memorize the shape of the cube and choose among other shapes.	60	true_shape_indices	—	Object
RememberShapeAndColor [Mode]-v0	3×2\3×3 5×3	Memorize the shape and color of the cube and choose among other shapes and colors.	60	true_shapes_info true_colors_info	—	Object
BunchOfColors[Mode]-v0	3 \ 5 \ 7	Remember the colors of the set of cubes shown simultaneously in the bunch and touch them in any order.	120	true_color_indices	—	Capacity
SeqOfColors[Mode]-v0	3 \ 5 \ 7	Remember the colors of the set of cubes shown sequentially and then select them in any order.	120	true_color_indices	—	Capacity
ChainOfColors[Mode]-v0	3 \ 5 \ 7	Remember the colors of the set of cubes shown sequentially and then select them in the same order.	120	true_color_indices	—	Sequential
Total: 32 tabletop robotic manipulation memory-intensive tasks in 12 groups						

with sequential recall challenges. Such task-specific strengths and limitations often remain hidden due to narrow evaluation scopes, underscoring the need for a comprehensive benchmark that spans diverse memory-intensive scenarios.

The challenge of memory evaluation becomes particularly evident in robotics. While some robotic tasks naturally involve partial observability, e.g. navigation tasks (Ai et al., 2022; Yadav et al., 2023), many studies artificially create partially observable scenarios from Markov Decision Processes (MDPs) (Åström, 1965) by introducing observation noise or masking parts of the state space (Spaan, 2012; Meng et al., 2021; Kurniawati, 2022; Lauri et al., 2023). However, these approaches do not fully capture the complexity of real-world robotic challenges (Lauri et al., 2023), where tasks may require the agent to recall past object configurations, manipulate occluded objects, or perform multi-step procedures that depend heavily on memory.

In this paper, we aim to address these challenges with the following three contributions:

1. **Memory Tasks Classification**: We develop a comprehensive yet practically simple classification of memory-intensive tasks. Our classification framework distills the complex landscape of memory challenges into four essential categories, enabling systematic evaluation while avoiding unnecessary complexity (Figure 1). This approach provides a clear, actionable framework for categorizing and selecting environments that capture fundamental memory challenges in RL and robotics (section 4).
2. **Unified Benchmark**: We introduce **MIKASA** (**M**emory-**I**ntensive **Sk**ills **A**ssessment **S**uite for **A**gents), a framework designed to evaluate memory capabilities of RL agents. Built upon the Gymnasium (Towers et al., 2024) interface, MIKASA provides a common platform for comparing and evaluating memory-enhanced RL agents (section 5).
3. **Robotic Manipulation Tasks**: We develop **ManiSkill-Memory**, a suite of 32 carefully designed robotic manipulation tasks that isolate and evaluate specific memory-dependent skills in realistic scenarios (section 6).

2 Related Works

Multiple RL benchmarks are designed to assess agents' memory capabilities. DMLab-30 (Hung et al., 2018) provides 3D navigation and puzzle tasks, focusing on long-horizon exploration and spatial recall. PsychLab (Leibo et al., 2018) extends DMLab by incorporating tasks that probe cognitive processes, including working memory. MiniGrid and MiniWorld (Chevalier-Boisvert et al., 2023) emphasize partial observability in lightweight 2D and 3D environments, while MiniHack (Samvelyan et al., 2021) builds on NetHack (Küttler et al., 2020), offering small roguelike scenarios that require both short- and long-term memory. BabyAI (Chevalier-Boisvert et al., 2019) combines natural language instructions with grid-based tasks, requiring memory for multi-step command execution. POPGym (Morad et al., 2023) standardizes memory evaluation with tasks ranging from pattern-matching puzzles to complex sequential decision-making. BSuite (Osband et al., 2020) offers a suite of carefully designed experiments that test core RL capabilities, including memory, through controlled tasks on exploration, credit assignment, and scalability. Memory Gym (Pleines et al., 2023) offers a

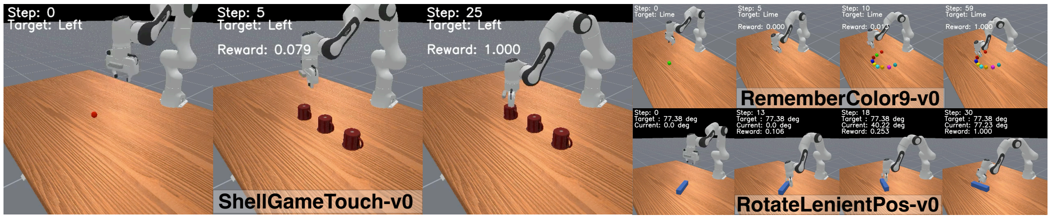

Figure 2: Illustration of demonstrative memory-intensive tasks execution from the proposed ManiSkill-Memory benchmark. The `ShellGameTouch-v0` task requires the agent to memorize the ball's location under mugs and touch the correct one. In `RememberColor9-v0`, the agent must memorize a cube's color and later select the matching one. In `RotateLenientPos-v0`, the agent must rotate a peg while keeping track of its previous rotations.

suite of 2D grid environments with partial observability, designed to benchmark memory capabilities in decision-making agents, including endless versions of tasks for evaluating memory over extremely long time intervals. Memory Maze (Pasukonis et al., 2022) presents 3D maze navigation tasks that require memory to solve efficiently.

While these benchmarks offer valuable insights into memory mechanisms, they generally focus on abstract puzzles or navigation tasks. However, none of them fully encompass the broad range of memory utilization scenarios an agent may encounter, and the tasks themselves often differ fundamentally across benchmarks, making direct comparison of memory-enhanced agents difficult.

In the robotics domain, memory requirements become particularly challenging due to the physical nature of manipulation tasks. Unlike abstract environments, robotic manipulation involves complex physical interactions and multi-step procedures that demand both spatial and temporal memory. Existing memory-intensive benchmarks, while useful for diagnostic purposes, struggle to capture these domain-specific challenges. The physical control and object interaction inherent in manipulation tasks introduce additional complexities that are not addressed by traditional memory evaluation frameworks.

Additionally, efforts have been made to classify memory-intensive environments based on specific attributes. For instance, Ni et al. (2023) categorizes these environments into memory/credit assignment, distinguishing them by temporal horizons. Yue et al. (2024) introduces memory dependency pairs, which capture the influence of past events on current decisions, enabling agents to leverage historical context for improved imitation learning in partially observable tasks. Cherepanov et al. (2024a) provides a formal division of agent memory into long-term and short-term depending on the agents' context length, as well as into declarative and procedural memory depending on the number of environments and episodes, and formalizes the notion of memory-intensive environments. Leibo et al. (2018) takes a different approach by directly adapting established tasks from cognitive psychology and visual psychophysics, providing a standardized way to evaluate agents on well-studied human cognitive benchmarks.

While these classification approaches offer insights into aspects of memory, they overlook physical dimensions in robotics. The interplay between physical interaction and memory remains unexplored, motivating the need for a framework that addresses spatio-temporal aspects in real-world tasks.

3 Background

3.1 Partially Observable Markov Decision Process

Partially Observable Markov Decision Process (POMDP) (Åström, 1965) extend MDP to account for partial observability, where an agent observes only noisy or incomplete information about the true environments state. POMDP defined by a tuple $(S, A, T, R, \Omega, O, \gamma)$, where: S is the set of states representing the complete environment configuration; A is the action space; $T(s'|s,a) : S \times A \times S \to [0,1]$ is the transition function defining the probability of reaching state s' from state s after taking action a; $R(s,a) : S \times A \to \mathbb{R}$ is the reward function specifying the immediate reward for taking action a in state s; Ω is the observation space containing all possible observations; $O(o|s,a) : S \times A \times \Omega \to [0,1]$ is the observation function defining the probability of observing o after taking action a and reaching state s; $\gamma \in [0,1)$ is the discount factor determining the importance of future rewards.

The objective is to find a policy π that maximizes the expected discounted cumulative reward: $\mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^{t}R(s_t,a_t)\right]$, where $a_t \sim \pi(\cdot|o_{1:t})$ depends on the history of observations rather than the true state. Relying on partial observations makes POMDPs harder to solve than MDPs.

Table 2: Key memory-intensive environments from the reviewed studies for evaluating agent memory. The Atari (Bellemare et al., 2013) environment with frame stacking is included to illustrate that many memory-enhanced agents are tested solely in MDP. Benchmark first introduced in the same work. Benchmark is open-sourced.

	DRQN (Hausknecht & Stone, 2015)	DTQN (Esslinger et al., 2022)	HCAM (Lampinen et al., 2021)	AMAGO (Grigsby et al., 2024)	GTrXL (Parisotto et al., 2020)	R2I (Samsami et al., 2024)	RATE (Cherepanov et al., 2024b)	R2A (Goyal et al., 2022)	Modified S5 (Lu et al., 2023)	Neural Map (Parisotto & Salakhutdinov, 2017)	GBMR (Kang et al., 2024)	EMDQN (Lin et al., 2018)	MRA (Fortunato et al., 2020)	FMRQN (Oh et al., 2016)	ADRQN (Zhu et al., 2018)	DCEM (Hill et al., 2020)	R2D2 (Kapturowski et al., 2018)	ERLAM (Zhu et al., 2020a)	AdaMemento (Yan et al., 2024)
Atari w/o FrameStack	✓														✓				✓
Atari with FrameStack						✓	✓	✓			✓	✓					✓	✓	
gym-gridverse		✓																	
car flag		✓																	
memory card		✓																	
Hallway		✓																	
HeavenHell		✓																	
Ballet			✓																
Object Permanence			✓																
DMLab-30			✓		✓												✓		
POPGym				✓		✓			✓							✓			
Passive T-Maze				✓			✓												
ViZDoom-Two-Colors							✓												
Numpad					✓														
Memory Maze						✓	✓												
Memory Maze (apples)					✓														
Minigrid-Memory							✓												
BSuite						✓			✓										
Goal-Search										✓									
Doom Maze										✓									
PsychLab													✓						
Spot the Difference													✓						
Goal Navigation													✓						
Transitive Inference													✓						
I-Maze														✓					
Pattern Matching														✓					
Random Maze														✓					
Unity Fast-Mapping Task																✓			
Action Associative Retrieval							✓												
BabyAI								✓											

3.2 Memory-intensive environments

Memory-intensive environment is an environment where agents must leverage past experiences to make decisions, often in problems with long-term dependencies or delayed rewards. More formally, following Cherepanov et al. (2024a), a memory-intensive task is a POMDP where there exists a correlation horizon $\xi > 1$, representing the minimum number of timesteps between an event critical for decision-making and when that information must be recalled. Popular memory-intensive environments in RL are listed in Table 2. One way to solving memory-intensive environments is to augment agents with memory mechanisms (see Appendix C).

3.3 Robotic Tabletop Manipulation

Robotic tabletop manipulation (Shridhar et al., 2022) involves robots manipulating objects on flat surfaces through actions like grasping, pushing, and picking. While crucial for real-world applications (Levine et al., 2018), most existing simulators treat these tasks as MDPs without memory requirements, failing to capture the spatio-temporal dependencies present in real scenarios. This limitation hinders the development of memory-enhanced agents for practical applications.

4 Classification of memory-intensive tasks

The evaluation of memory capabilities in RL faces two major challenges. First, as shown in Table 2, research studies use different sets of environments with minimal overlap, making it difficult to compare memory-enhanced agents across studies. Second, even within individual studies, benchmarks may focus on testing similar memory aspects (e.g., remembering object locations) while neglecting others (e.g., reconstructing sequential events), leading to incomplete evaluation of agents' memory.

Different architectures may exhibit varying performance across memory tasks. For instance, an architecture optimized for long-term object property recall might struggle with sequential memory tasks, yet these limitations often remain undetected due to the narrow focus of existing evaluation approaches.

To address these challenges, we propose a systematic approach to memory evaluation in RL. Given the impracticality of testing agents on every possible memory-intensive environment, **we aim to identify a minimal diagnostic set that comprehensively covers different memory requirements**. Drawing from established research in developmental psychology and cognitive science, where similar memory challenges have been extensively studied in humans, we develop a categorization framework consisting of four distinct memory task classes, detailed in subsection 4.2.

4.1 Memory: From Cognitive Science to RL

In developmental psychology and cognitive science, memory is classified into categories based on cognitive processes. Key concepts include object permanence (Piaget, 1952), which involves remem-

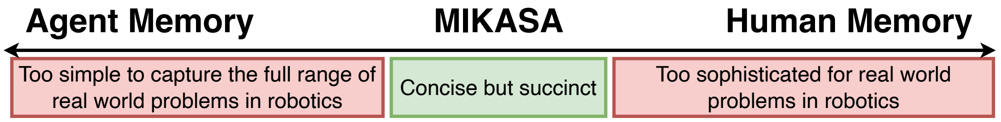
Agent Memory
MIKASA
Human Memory
Too simple to capture the full range of real world problems in robotics
Concise but succinct
Too sophisticated for real world problems in robotics

Figure 3: MIKASA bridges the gap between human-like memory complexity and robotic task requirements. While robotic tasks don't require the full spectrum of human memory capabilities, they can't be reduced to simple spatio-temporal dependencies. MIKASA provides a balanced framework that captures essential memory aspects for robotic tasks while maintaining practical simplicity.

bering the existence of objects out of sight, and categorical perception (Liberman et al., 1957), where objects are grouped based on attributes like color or shape. Working memory (Baddeley, 1992) and memory span (Daneman & Carpenter, 1980) refer to the ability to hold and manipulate information over time, while causal reasoning (Kuhn, 2012) and transitive inference (Heckers et al., 2004) involve understanding cause-and-effect relationships and deducing hidden relationships, respectively.

The RL field has attempted to utilize these concepts in the design of specific memory-intensive environments Fortunato et al. (2020); Lampinen et al. (2021), but these have been limited at the task design level. Of particular interest, however, is how existing memory-intensive tasks can be categorized using these concepts to develop a benchmark on which to test the greatest number of memory capabilities of memory-enhanced agents, and it is this problem that we address in this paper. Thus, we aim to provide a balanced framework that covers important aspects of memory for real-world applications while maintaining practical simplicity (see Figure 3).

4.2 Taxonomy of Memory Tasks

We introduce a comprehensive task classification framework for evaluating memory mechanisms in RL. Our framework categorizes memory-intensive tasks into four fundamental types, each targeting distinct aspects of memory capabilities:

1. **Object Memory.** Tasks that evaluate an agent's ability to maintain object-related information over time, particularly when objects become temporarily unobservable. These tasks align with the cognitive concept of object permanence, requiring agents to track object properties when occluded, maintain object state representations, and recognize encountered objects.
2. **Spatial Memory.** Tasks focused on environmental awareness and navigation, where agents must remember object locations, maintain mental maps of environment layouts, and navigate based on previously observed spatial information.
3. **Sequential Memory.** Tasks that test an agent's ability to process and utilize temporally ordered information, similar to human serial recall and working memory. These tasks require remembering action sequences, maintaining order-dependent information, and using past decisions to inform future actions.
4. **Memory Capacity.** Tasks that challenge an agent's ability to manage multiple pieces of information simultaneously, analogous to human memory span. These tasks evaluate information retention limits and multi-task information processing.

This classification framework enables systematic evaluation of memory-enhanced RL agents across diverse scenarios. By providing a structured approach to memory task categorization, we establish a foundation for comprehensive benchmarking that spans the wide spectrum of memory requirements. In the following section, we present a carefully curated set of tasks based on this classification, forming the basis of our proposed MIKASA benchmark.

5 MIKASA: Memory-Intensive Skills Assessment Suite for Agents

Motivation and Overview. The RL domain currently lacks standardized benchmarks for evaluating agents' memory capabilities. While numerous memory-intensive environments exist, their dispersion across different research projects makes systematic comparison challenging. Moreover, existing frameworks often focus on narrow aspects of memory, failing to capture the diverse memory requirements found in real-world applications. To address these limitations, we introduce MIKASA

Table 4: Recommended memory-intensive environments for comprehensive agent evaluation.

Memory Type	Diagnostic Tasks	Complex Tasks
Object Memory	Passive T-Maze (Ni et al., 2023)	ViZDoom-Two-Colors (Sorokin et al., 2022)
Spatial Memory	POPGym Labyrinth (Morad et al., 2023)	Memory Maze (Pasukonis et al., 2022)
Sequential Memory	POPGym Autoencode (Morad et al., 2023)	Ballet (Lampinen et al., 2021)
Memory Capacity	Memory Cards (Esslinger et al., 2022)	–

(Memory-Intensive Skills Assessment Suite for Agents), a unified benchmark that systematically evaluates memory capabilities across diverse tasks while maintaining practical simplicity.

Benchmark Design Principles. Our benchmark follows key design principles that ensure comprehensive evaluation of memory capabilities. To isolate memory mechanisms from other learning challenges, MIKASA implements a two-tiered task structure. The first tier consists of **diagnostic** vector-based environments, enabling direct validation of specific memory mechanisms in atomic tasks. The second tier comprises **complex** image-based environments that introduce additional challenges through 2D observation processing, more closely approximating real-world scenarios. This hierarchical approach allows researchers to first validate fundamental memory capabilities before progressing to more sophisticated tasks.

Table 3: Analysis of established robotics frameworks with manipulation tasks, comparing their support for memory-intensive tasks. † – excluding Franka Kitchen.

Robotics Framework with Manipulation Tasks	**Memory Tasks**		
	Manipulation	**Atomic**	**Low-level actions**
`ManiSkill-Memory` (**Ours**)	✓	✓	✓
`ManiSkill3` (Tao et al., 2024)	✗	✗	✗
`ManiSkill-HAB` (Shukla et al., 2024)	✗	✗	✗
`RoboCasa` (Nasiriany et al., 2024)	✗	✗	✗
`Gymnasium-Robotics`† (de Lazcano et al., 2024)	✗	✗	✗
`BEHAVIOR-1K` (Li et al., 2024)	✓	✗	✗
`ARNOLD` (Gong et al., 2023)	✗	✗	✗
`iGibson 2.0` (Li et al., 2022)	✓	✗	✗
`VIMA` (Jiang et al., 2022)	✓	✓	✗
`Isaac Sim` (Makoviychuk et al., 2021)	✗	✗	✗
`panda-gym` (Gallouédec et al., 2021)	✗	✗	✗
`Habitat 2.0` (Szot et al., 2021)	✗	✗	✗
`Meta-World` (Yu et al., 2020)	✗	✗	✗
`CausalWorld` (Ahmed et al., 2020)	✗	✗	✗
`RLBench` (James et al., 2020)	✗	✗	✗
`robosuite` (Zhu et al., 2020b)	✗	✗	✗
`dm_control` (Tunyasuvunakool et al., 2020)	✗	✗	✗
`Franka Kitchen` (Gupta et al., 2019)	✗	✗	✗
`SURREAL` (Fan et al., 2018)	✗	✗	✗
`AI2-THOR` (Kolve et al., 2017)	✗	✗	✗

Task Classification and Selection. Building upon our taxonomy presented in subsection 4.2, we conducted a systematic analysis of existing open-source memory-intensive environments from Table 2. Our analysis, detailed in Appendix, Table 5, revealed four distinct classes of memory tasks. This classification enabled us to identify a minimal yet representative set of environments that spans the large spectrum of memory utilization patterns, from object permanence to sequential decision-making, while maintaining practical simplicity. Comprehensive descriptions of all considered environments are provided in Appendix G.

We have unified these environments under the Gymnasium API (Towers et al., 2024), ensuring seamless integration with existing RL tools and workflows (see Table 4). This standardization simplifies access to scattered environments and facilitates direct architectural comparisons. For detailed implementation guidelines and example usage of our MIKASA framework, we refer readers to Appendix B.

To evaluate agents in realistic memory-intensive scenarios, we introduce our ManiSkill-Memory benchmark (section 6). This benchmark provides a suite of robotic manipulation tasks that systematically assess all four memory types in practical, real-world-inspired contexts.

MIKASA represents a significant advancement in standardizing memory evaluation for RL. Through its carefully curated environment selection and hierarchical structure, MIKASA enables systematic evaluation of memory-enhanced architectures, facilitates direct comparison between different memory mechanisms, and provides a clear progression path from fundamental to complex memory tasks. This structured approach allows precise identification of memory-related limitations in RL agents while maintaining practical utility.

6 MANISKILL-MEMORY

The landscape of robotic manipulation frameworks reveals significant limitations in addressing memory-intensive tasks. First, while partial observability is extensively studied in navigation tasks, manipulation scenarios are predominantly evaluated under full observability, with memory requirements receiving limited attention (see Table 3). Second, among the few frameworks that incorporate memory-intensive manipulation tasks, significant limitations exist. BEHAVIOR-1k (Li et al., 2024) and iGibson 2.0 (Li et al., 2022) employ highly complex, non-atomic tasks that obscure the evaluation

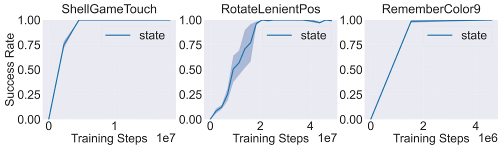

Figure 5: Performance of PPO-MLP trained in `state` mode, i.e., in MDP mode without the need for memory. These results suggest that the proposed tasks are inherently solvable with a success rate of 100%.

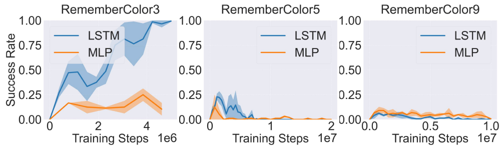

Figure 6: PPO with MLP and LSTM backbones trained in `RGB+joints` mode on the `RememberColor-v0` environment with dense rewards. Both architectures fail to solve medium and high complexity tasks.

of specific memory mechanisms. Similarly, VIMA (Jiang et al., 2022) relies on high-level actions that inadequately capture memory performance over extended time horizons. To the best of our knowledge, there are no benchmarks specifically designed to evaluate memory in RL in the robotic manipulation domain. To fill this gap, we introduce the ManiSkill-Memory framework for the RL.

6.1 MANISKILL-MEMORY BENCHMARK

ManiSkill-Memory is a benchmark designed for memory-intensive robotic tabletop manipulation tasks, simulating real-world challenges commonly encountered by robots. These tasks include locating occluded objects, recalling previous configurations, and executing complex sequences of actions over extended time horizons. By incorporating meaningful partial observability, this framework offers a systematic approach to test an agent's memory mechanisms.

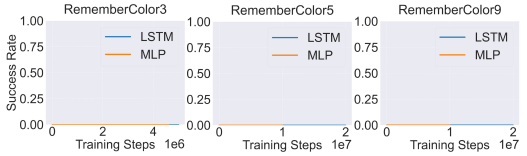

Figure 4: PPO with MLP and LSTM backbones trained in `RGB+joints` mode on the `RememberColor-v0` environment with sparse rewards. Both LSTM and MLP cannot solve this task, which emphasizes their limitations in such scenarios and the need to develop new memory-enhanced agents.

Building upon the robust foundation of ManiSkillv3 framework (Tao et al., 2024), our benchmark leverages its efficient parallel GPU-based training capabilities to create and evaluate these tasks.

6.2 MANISKILL-MEMORY MANIFESTATION

In designing the tasks, we drew inspiration from the four memory types identified in our classification framework (subsection 4.2). We developed 32 tasks across 12 categories of robotic tabletop manipulation, each targeting specific aspects of object memory, spatial memory, sequential memory, and memory capacity. These tasks feature varying levels of complexity, allowing for systematic evaluation of different memory mechanisms. For instance, some tasks test object permanence by requiring the agent to track occluded objects, while others challenge sequential memory by requiring the reproduction of a strict order of actions. A summary of these tasks and their corresponding memory types is provided in Table 1, with detailed descriptions in Appendix E.

To illustrate the concept of our memory-intensive framework, we present `ShellGameTouch-v0`, `RememberColor-v0`, and `RotateLenientPos-v0` tasks in Figure 2. In the `ShellGameTouch-v0` task, the agent observes a red ball placed in one of three positions over the first 5 steps ($t \in [0, 4]$). At $t = 5$, the ball and the three positions are covered by mugs. The agent must then determine the location of the ball by interacting with the correct mug. In the simplest mode (`Touch`), the agent only needs to touch the correct mug, whereas in other modes, it must either push or lift the mug. In the `RememberColor-v0` task, the agent observes a cube of a specific color for 5 steps ($t \in [0, 4]$). After the cube disappears for 5 steps, 3, 5, or 9 (depending on task mode) cubes of different colors appear at $t = 10$. The agent's task is to identify and select the same cube it initially saw. In the `RotateLenientPos-v0` task, the agent must rotate a randomly oriented peg by a specified clockwise angle.

The ManiSkill-Memory benchmark supports several training modes: `state` mode: the agent receives all necessary information in vector form, including oracle data and the Tool Center Point (TCP) pose. This mode treats the task as a pure MDP; `RGB` mode: the agent receives two images (a top view of the table and a view from the camera on the gripper) with TCP position information; `joints` mode:

the agent gets joint positions, velocities, and TCP pose, but no environmental data; `oracle` mode: provides task-specific environment information, such as target cap number or ball velocity, useful for memory mechanisms debugging; `promptt` mode: supplies static information to the agent at each step, such as prompted rotation angles in `RotateLenient-v0` and `RotateStrict-v0`. Any combination of these modes is allowed, though **`RGB + joints` is the standard for testing memory**. `State` mode is used for MDP-based tasks.

The ManiSkill-Memory benchmark implements two types of reward functions: dense and sparse. The dense reward provides continuous feedback based on the agent's progress towards the goal, while the sparse reward only signals task completion. While dense rewards facilitate faster learning in our experiments, sparse rewards better reflect real-world scenarios where intermediate feedback is often unavailable, making them crucial for evaluating practical applicability of memory-enhanced agents.

6.3 Performance of classic baselines on ManiSkill-Memory benchmark

For our experimental evaluation, we selected PPO (Schulman et al., 2017) with two backbone architectures: Multilayer Perceptron (MLP) and Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997). The MLP variant serves as a memory-less baseline, while LSTM represents a widely-adopted memory mechanism in RL, known for its effectiveness in solving POMDPs (Ni et al., 2021). This choice of architectures enables direct comparison between memory-less and memory-enhanced agents while validating our benchmark's ability to assess memory. We focus specifically on these fundamental architectures as they align with our primary goal of benchmark validation rather than comprehensive algorithm comparison.

To demonstrate that all proposed environments are solvable with 100% success rate (SR), we trained a PPO-MLP agent using the `state` mode, where it had full access to the system information. Results for the demo environments are presented in Figure 5, with additional results for all tasks available in Appendix D.

Training under the `RGB+joints` mode with dense rewards reveals the memory-intensive nature of our tasks. Using the `RememberColor-v0` task as an example, PPO-LSTM demonstrates superior performance compared to PPO-MLP when distinguishing between three colors (see Figure 6). However, both agents' success rates drop dramatically to near-zero as the task complexity increases to five or nine colors. Moreover, under sparse reward conditions, both architectures fail to solve even the three-color variant (see Figure 4). These results validate our benchmark's effectiveness in evaluating agents' memory capabilities, showing clear performance degradation as memory demands increase.

Our baseline experiments reveal several key insights: (1) the proposed tasks are inherently solvable, as demonstrated by the perfect performance in `state` mode; (2) the tasks effectively challenge memory capabilities, evidenced by the performance gap between memory-less (MLP) and memory-enhanced (LSTM) architectures; and (3) primitive memory mechanisms show clear limitations as task complexity increases, particularly under sparse rewards. These findings validate ManiSkill-Memory as an effective benchmark for evaluating and developing memory-enhanced RL agents in robotic manipulation tasks.

7 Conclusion

In this work, we addressed the critical gap in memory-intensive RL research through three key contributions. First, we developed a comprehensive classification framework that categorizes memory tasks into four distinct classes: object memory, spatial memory, sequential memory, and memory capacity. This taxonomy provides a structured approach to understanding and evaluating different aspects of memory in RL agents. Second, we introduced a unified benchmark that consolidates diverse memory-intensive environments into a single, standardized framework. By carefully selecting representative tasks from each memory category, our benchmark enables systematic comparison and evaluation of memory-enhanced RL agents across a broad spectrum of memory challenges. Third, we presented ManiSkill-Memory, a novel benchmark comprising 23 carefully designed memory-intensive tasks for robotic manipulation, which bridges the gap between abstract memory challenges and practical robotics applications.

References

Ossama Ahmed, Frederik Träuble, Anirudh Goyal, Alexander Neitz, Yoshua Bengio, Bernhard Schölkopf, Manuel Wüthrich, and Stefan Bauer. Causalworld: A robotic manipulation benchmark for causal structure and transfer learning. *arXiv preprint arXiv:2010.04296*, 2020.

Bo Ai, Wei Gao, David Hsu, et al. Deep visual navigation under partial observability. In *2022 International Conference on Robotics and Automation (ICRA)*, pp. 9439–9446. IEEE, 2022.

Chenxin An, Shansan Gong, Ming Zhong, Xingjian Zhao, Mukai Li, Jun Zhang, Lingpeng Kong, and Xipeng Qiu. L-eval: Instituting standardized evaluation for long context language models. *arXiv preprint arXiv:2307.11088*, 2023.

Karl Johan Åström. Optimal control of markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, 10:174–205, 1965. URL `https://api.semanticscholar.org/CorpusID:121222106`.

Alan Baddeley. Working memory. *Science*, 255(5044):556–559, 1992.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*, 2023.

Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5):834–846, 1983. doi: 10.1109/TSMC.1983.6313077.

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, Nikita Kachaev, Artem Zholus, Alexey K. Kovalev, and Aleksandr I. Panov. Unraveling the complexity of memory in rl agents: an approach for classification and evaluation, 2024a. URL `https://arxiv.org/abs/2412.06531`.

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

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: A platform to study the sample efficiency of grounded language learning, 2019. URL `https://arxiv.org/abs/1810.08272`.

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

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Meredyth Daneman and Patricia A Carpenter. Individual differences in working memory and reading. *Journal of verbal learning and verbal behavior*, 19(4):450–466, 1980.

Rodrigo de Lazcano, Kallinteris Andreas, Jun Jet Tai, Seungjae Ryan Lee, and Jordan Terry. Gymnasium robotics, 2024. URL `http://github.com/Farama-Foundation/Gymnasium-Robotics`.

Kenji Doya. Temporal difference learning in continuous time and space. In *Neural Information Processing Systems*, 1995. URL `https://api.semanticscholar.org/CorpusID:1170136`.

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

Linxi Fan, Yuke Zhu, Jiren Zhu, Zihua Liu, Orien Zeng, Anchit Gupta, Joan Creus-Costa, Silvio Savarese, and Li Fei-Fei. Surreal: Open-source reinforcement learning framework and robot manipulation benchmark. In Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto (eds.), *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pp. 767–782. PMLR, 29–31 Oct 2018. URL https://proceedings.mlr.press/v87/fan18a.html.

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.

Quentin Gallouédec, Nicolas Cazin, Emmanuel Dellandréa, and Liming Chen. panda-gym: Open-Source Goal-Conditioned Environments for Robotic Learning. *4th Robot Learning Workshop: Self-Supervised and Lifelong Learning at NeurIPS*, 2021.

Hector Geffner and Blai Bonet. Solving large pomdps using real time dynamic programming. 1998. URL https://api.semanticscholar.org/CorpusID:14063723.

Ran Gong, Jiangyong Huang, Yizhou Zhao, Haoran Geng, Xiaofeng Gao, Qingyang Wu, Wensi Ai, Ziheng Zhou, Demetri Terzopoulos, Song-Chun Zhu, et al. Arnold: A benchmark for language-grounded task learning with continuous states in realistic 3d scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 20483–20495, 2023.

Anirudh Goyal, Abram Friesen, Andrea Banino, Theophane Weber, Nan Rosemary Ke, Adria Puigdomenech Badia, Arthur Guez, Mehdi Mirza, Peter C Humphreys, Ksenia Konyushova, et al. Retrieval-augmented reinforcement learning. In *International Conference on Machine Learning*, pp. 7740–7765. PMLR, 2022.

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.

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019.

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.

Stephan Heckers, Martin Zalesak, Anthony P Weiss, Tali Ditman, and Debra Titone. Hippocampal activation during transitive inference in humans. *Hippocampus*, 14(2):153–162, 2004.

Felix Hill, Olivier Tieleman, Tamara von Glehn, Nathaniel Wong, Hamza Merzic, and Stephen Clark. Grounded language learning fast and slow, 2020. URL https://arxiv.org/abs/2009.01719.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8): 1735–1780, nov 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Jan Humplik, Alexandre Galashov, Leonard Hasenclever, Pedro A. Ortega, Yee Whye Teh, and Nicolas Heess. Meta reinforcement learning as task inference, 2019. URL https://arxiv.org/abs/1905.06424.

Chia-Chun Hung, Timothy Lillicrap, Josh Abramson, Yan Wu, Mehdi Mirza, Federico Carnevale, Arun Ahuja, and Greg Wayne. Optimizing agent behavior over long time scales by transporting value, 2018. URL https://arxiv.org/abs/1810.06721.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.

Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. Vima: General robot manipulation with multimodal prompts. *arXiv preprint arXiv:2210.03094*, 2(3):6, 2022.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1):99–134, 1998. ISSN 0004-3702. doi: https://doi.org/10.1016/S0004-3702(98)00023-X. URL https://www.sciencedirect.com/science/article/pii/S000437029800023X.

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, 2024. doi: 10.1109/TAI.2023.3268612.

Steven Kapturowski, Georg Ostrovski, John Quan, Rémi Munos, and Will Dabney. Recurrent experience replay in distributed reinforcement learning. In *International Conference on Learning Representations*, 2018. URL https://api.semanticscholar.org/CorpusID:59345798.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, et al. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.

Deanna Kuhn. The development of causal reasoning. *Wiley Interdisciplinary Reviews: Cognitive Science*, 3(3):327–335, 2012.

Hanna Kurniawati. Partially observable markov decision processes and robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 5(1):253–277, 2022.

Heinrich Küttler, Nantas Nardelli, Alexander H. Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The nethack learning environment, 2020. URL https://arxiv.org/abs/2006.13760.

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

Mikko Lauri, David Hsu, and Joni Pajarinen. Partially observable markov decision processes in robotics: A survey. *IEEE Transactions on Robotics*, 39(1):21–40, February 2023. ISSN 1941-0468. doi: 10.1109/tro.2022.3200138. URL http://dx.doi.org/10.1109/TRO.2022.3200138.

Joel Z. Leibo, Cyprien de Masson d'Autume, Daniel Zoran, David Amos, Charles Beattie, Keith Anderson, Antonio García Castañeda, Manuel Sanchez, Simon Green, Audrunas Gruslys, Shane Legg, Demis Hassabis, and Matthew M. Botvinick. Psychlab: A psychology laboratory for deep reinforcement learning agents, 2018. URL https://arxiv.org/abs/1801.08116.

Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International journal of robotics research*, 37(4-5):421–436, 2018.

Chengshu Li, Fei Xia, Roberto Martín-Martín, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Elliott Vainio, Cem Gokmen, Gokul Dharan, Tanish Jain, Andrey Kurenkov, Karen Liu, Hyowon Gweon, Jiajun Wu, Li Fei-Fei, and Silvio Savarese. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. In Aleksandra Faust, David Hsu, and Gerhard Neumann (eds.), *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 455–465. PMLR, 08–11 Nov 2022. URL `https://proceedings.mlr.press/v164/li22b.html`.

Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Wensi Ai, Benjamin Martinez, et al. Behavior-1k: A human-centered, embodied ai benchmark with 1,000 everyday activities and realistic simulation. *arXiv preprint arXiv:2403.09227*, 2024.

Alvin M Liberman, Katherine Safford Harris, Howard S Hoffman, and Belver C Griffith. The discrimination of speech sounds within and across phoneme boundaries. *Journal of experimental psychology*, 54(5):358, 1957.

Zichuan Lin, Tianqi Zhao, Guangwen Yang, and Lintao Zhang. Episodic memory deep q-networks, 2018. URL `https://arxiv.org/abs/1805.07603`.

Michael L. Littman, Anthony R. Cassandra, and Leslie Pack Kaelbling. Learning policies for partially observable environments: Scaling up. In Armand Prieditis and Stuart Russell (eds.), *Machine Learning Proceedings 1995*, pp. 362–370. Morgan Kaufmann, San Francisco (CA), 1995. ISBN 978-1-55860-377-6. doi: https://doi.org/10.1016/B978-1-55860-377-6.50052-9. URL `https://www.sciencedirect.com/science/article/pii/B9781558603776500529`.

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`.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

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

Steven Morad, Ryan Kortvelesy, Matteo Bettini, Stephan Liwicki, and Amanda Prorok. POPGym: Benchmarking partially observable reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=chDrutUTs0K`.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. *arXiv preprint arXiv:2406.02523*, 2024.

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, 2023. URL `https://arxiv.org/abs/2307.03864`.

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 Szepesvári, Satinder Singh, Benjamin Van Roy, Richard Sutton, David Silver, and Hado van Hasselt. Behaviour suite for reinforcement learning. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=rygf-kSYwH`.

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

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

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

John Piaget. The origins of intelligence in children. *International University*, 1952.

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.

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.

Mikayel Samvelyan, Robert Kirk, Vitaly Kurin, Jack Parker-Holder, Minqi Jiang, Eric Hambro, Fabio Petroni, Heinrich Küttler, Edward Grefenstette, and Tim Rocktäschel. Minihack the planet: A sandbox for open-ended reinforcement learning research, 2021. URL https://arxiv.org/abs/2109.13202.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on robot learning*, pp. 894–906. PMLR, 2022.

Arth Shukla, Stone Tao, and Hao Su. Maniskill-hab: A benchmark for low-level manipulation in home rearrangement tasks. *arXiv preprint arXiv:2412.13211*, 2024.

Aleksandrs Slivkins. Introduction to multi-armed bandits, 2024. URL https://arxiv.org/abs/1904.07272.

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.

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.

Matthijs T. J. Spaan. Partially observable Markov decision processes. In Marco Wiering and Martijn van Otterlo (eds.), *Reinforcement Learning: State of the Art*, pp. 387–414. Springer Verlag, 2012.

Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, et al. Habitat 2.0: Training home assistants to rearrange their habitat. *Advances in neural information processing systems*, 34:251–266, 2021.

Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse-kai Chan, et al. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. *arXiv preprint arXiv:2410.00425*, 2024.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020.

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.

Karmesh Yadav, Jacob Krantz, Ram Ramrakhya, Santhosh Kumar Ramakrishnan, Jimmy Yang, Austin Wang, John Turner, Aaron Gokaslan, Vincent-Pierre Berges, Roozbeh Mootaghi, Oleksandr Maksymets, Angel X Chang, Manolis Savva, Alexander Clegg, Devendra Singh Chaplot, and Dhruv Batra. Habitat challenge 2023. `https://aihabitat.org/challenge/2023/`, 2023.

Renye Yan, Yaozhong Gan, You Wu, Junliang Xing, Ling Liangn, Yeshang Zhu, and Yimao Cai. Adamemento: Adaptive memory-assisted policy optimization for reinforcement learning, 2024. URL `https://arxiv.org/abs/2410.04498`.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020.

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

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020.

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`.

Guangxiang Zhu, Zichuan Lin, Guangwen Yang, and Chongjie Zhang. Episodic reinforcement learning with associative memory. In *International Conference on Learning Representations*, 2020a. URL `https://api.semanticscholar.org/CorpusID:212799813`.

Pengfei Zhu, Xin Li, Pascal Poupart, and Guanghui Miao. On improving deep reinforcement learning for pomdps, 2018. URL `https://arxiv.org/abs/1704.07978`.

Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Kevin Lin, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning. In *arXiv preprint arXiv:2009.12293*, 2020b.

Table of Contents

A MANISKILL-MEMORY IMPLEMENTATION DETAILS

An example of running the environment from the ManiSkill-Memory benchmark is shown in Code 1. For ease of debugging, we also added various wrappers (found in `ManiSkillMemory/utils/wrappers/`) that display useful information about the episode on the video. Thus, `RenderStepInfoWrapper()` displays the current step in the environment; `DebugRewardWrapper()` displays information about the full reward at the current step in the environment; `DebugRewardWrapper()` displays information about each component that generates the reward function at the current step. In addition, we also added task-specific wrappers for each environment. For example, `RememberColorInfoWrapper()` displays the target color of the cube in the `RememberColor-v0` task, and `ShellGameRenderCupInfoWrapper()` displays which mug the ball is actually under in the `ShellGame-v0` task.

Code 1: Example code for running `RememberColor9-v0` environment from ManiSkill-Memory.

```python
# Import ManiSkill-Memory tasks
import ManiSkillMemory
# Import ManiSkill-Memory wrappers
from ManiSkillMemory.utils.wrappers import *
# Import RecordEpisode from the original ManiSkill3
from mani_skill.utils.wrappers import RecordEpisode

num_envs, seed = 512, 123

# Create the environment via gym.make()
# obs_mode="rgb" for modes "RGB", "RGB+joint", "RGB+oracle" etc.
# obs_mode="state" for mode "state"
env = gym.make("RememberColor9-v0", num_envs=num_envs,
                obs_mode="rgb", render_mode="all")

# [use always] to generate required observation keys
env = StateOnlyTensorToDictWrapper(env)
# [use for debug] to show specific env info on video
env = RememberColorInfoWrapper(env)
# [use for debug] to show env step on video
env = RenderStepInfoWrapper(env)
# [use for debug] to show agent total reward on video
env = RenderRewardInfoWrapper(env)
# [use for debug] show each component of the reward function on video
env = DebugRewardWrapper(env)
# [use to record video]
env = RecordEpisode(env, "./videos/demo_remember-color-9")

obs, _ = env.reset(seed)
for i in tqdm(range(89)):
    action = env.action_space.sample()
    obs, reward, terminated, truncated, info = env.step(torch.from_numpy(action))
env.close()

Video("./videos/demo_remember-color-9/0.mp4", embed=True, width=1240)
```

B MIKASA Implementation Details

An example of running an environment from the MIKASA benchmark is shown in Code 2. MIKASA supports the standard Gymnasium API and is fully compatible with all its wrappers. This allows users to leverage various functionalities, including parallelization using `SyncVectorEnv` to improve training efficiency. MIKASA provides a predefined set of environments with different levels of difficulty. However, users can customize the environment parameters by passing specific arguments (see Code 2).

Code 2: Example code for running `MemoryLength-v0` environment.

```
# Import necessary libraries
import membench
import gymnasium as gym

def make_env(env_id, idx, capture_video, run_name, env_kwargs):
    def thunk():
        if capture_video and idx == 0:
            env = gym.make(env_id, render_mode="rgb_array", **
                env_kwargs)
            env = gym.wrappers.RecordVideo(env, f"videos/{run_name
                }")
        else:
            env = gym.make(env_id, **env_kwargs)
        env = gym.wrappers.RecordEpisodeStatistics(env)
        return env
    return thunk

# Setup environment with custom parameters
num_envs = 8
env_id = ’MemoryLength-v0’
env_kwargs = {’memory_length’: 10, ’num_bits’: 1}

# Setup environment from our task set
# env_id = ’MemoryLengthEasy-v0’
# env_kwargs = None

envs = gym.vector.SyncVectorEnv(
    [make_env(env_id, i, False, ’test’, env_kwargs) for i in range
        (num_envs)]
)

obs, _ = envs.reset(seed=1)

for i in range(11):
    action = envs.action_space.sample()
    next_obs, reward, terminations, truncations, infos = envs.step
        (action)
```

C Memory Mechanisms in RL

In RL, memory mechanisms are techniques or models used to enable agents to retain and recall information from past interactions with the environment.

There are several approaches to incorporating memory into RL, including recurrent neural networks (RNNs) (Rumelhart et al., 1986; Hochreiter & Schmidhuber, 1997; Chung et al., 2014) which uses hidden states to store information from previous steps (Wierstra et al., 2010; Hausknecht & Stone, 2015), state-space models (SSMs) (Gu et al., 2021; Smith et al., 2023; Gu & Dao, 2023) which uses system state to store historical information (Hafner et al., 2019; Samsami et al., 2024), transformers (Vaswani et al., 2017) which uses attention mechanism to capture sequential dependencies inside the context window (Parisotto et al., 2020; Lampinen et al., 2021; Ni et al., 2023), graph neural networks (GNNs) (Zhou et al., 2020) which uses graphs to store information Zhu et al. (2023); Kang et al. (2024) etc. Popular agents with memory mechanisms are summarized in Table 2.

D CLASSIC BASELINES PERFORMANCE ON THE MANISKILL-MEMORY BENCHMARK

In this section, we present a comprehensive evaluation of PPO-MLP and PPO-LSTM baselines on our ManiSkill-Memory benchmark. Our experiments with PPO-MLP in `state` mode using dense rewards demonstrate perfect performance across all tasks, consistently achieving 100% success rate, as shown in Figure 7 and Figure 8. This remarkable performance serves as a crucial validation of our benchmark design: when an agent has access to complete state information and receives dense rewards, it can master these tasks completely. Therefore, any performance degradation in `RGB+joints` mode observed with other algorithms or training configurations must stem from the algorithmic limitations or learning challenges rather than any inherent flaws in the task design. This empirical evidence confirms that our environments are well-calibrated and properly designed, establishing a solid foundation for evaluating memory-enhanced algorithms. All results are presented as mean ± standard error of the mean (SEM), where the mean is computed across three independent training runs, and each trained agent is evaluated on 16 different random seeds to ensure robust performance assessment.

The performance evaluation of PPO-MLP and PPO-LSTM with dense rewards in the `RGB+joints` mode is presented in Figure 9. This mode specifically tests the agents' memory capabilities, as it requires remembering and utilizing historical information to solve the tasks. Our results demonstrate a clear distinction between memory-less and memory-enhanced architectures, while also revealing the limitations of conventional memory mechanisms.

Consider the `RememberColor-v0` environment as an illustrative example. In its simplest configuration with three cubes, the memory-less PPO-MLP achieves only 25% success rate. In contrast, PPO-LSTM, leveraging its memory mechanism, achieves perfect performance with 100% success rate. However, as task complexity increases to five or nine cubes, even the LSTM's memory capabilities prove insufficient, with performance degrading significantly.

These results validate two key aspects of our benchmark: first, its effectiveness in distinguishing between memory-less and memory-enhanced architectures, and second, its ability to challenge even sophisticated memory mechanisms as task complexity increases. This demonstrates that ManiSkill-Memory provides a competitive yet meaningful evaluation framework for developing and testing advanced memory-enhanced agents.

Our evaluation of PPO-MLP and PPO-LSTM baselines under sparse reward conditions in `RGB+joints` mode reveals the true challenge of our benchmark tasks. As shown in Figure 10, both architectures – even the memory-enhanced LSTM – consistently fail to achieve any meaningful success rate across nearly all considered environments. This striking result underscores the extreme difficulty of memory-intensive manipulation tasks when only terminal rewards are available, highlighting the substantial gap between current algorithms and the level of memory capabilities required for real-world robotic applications.

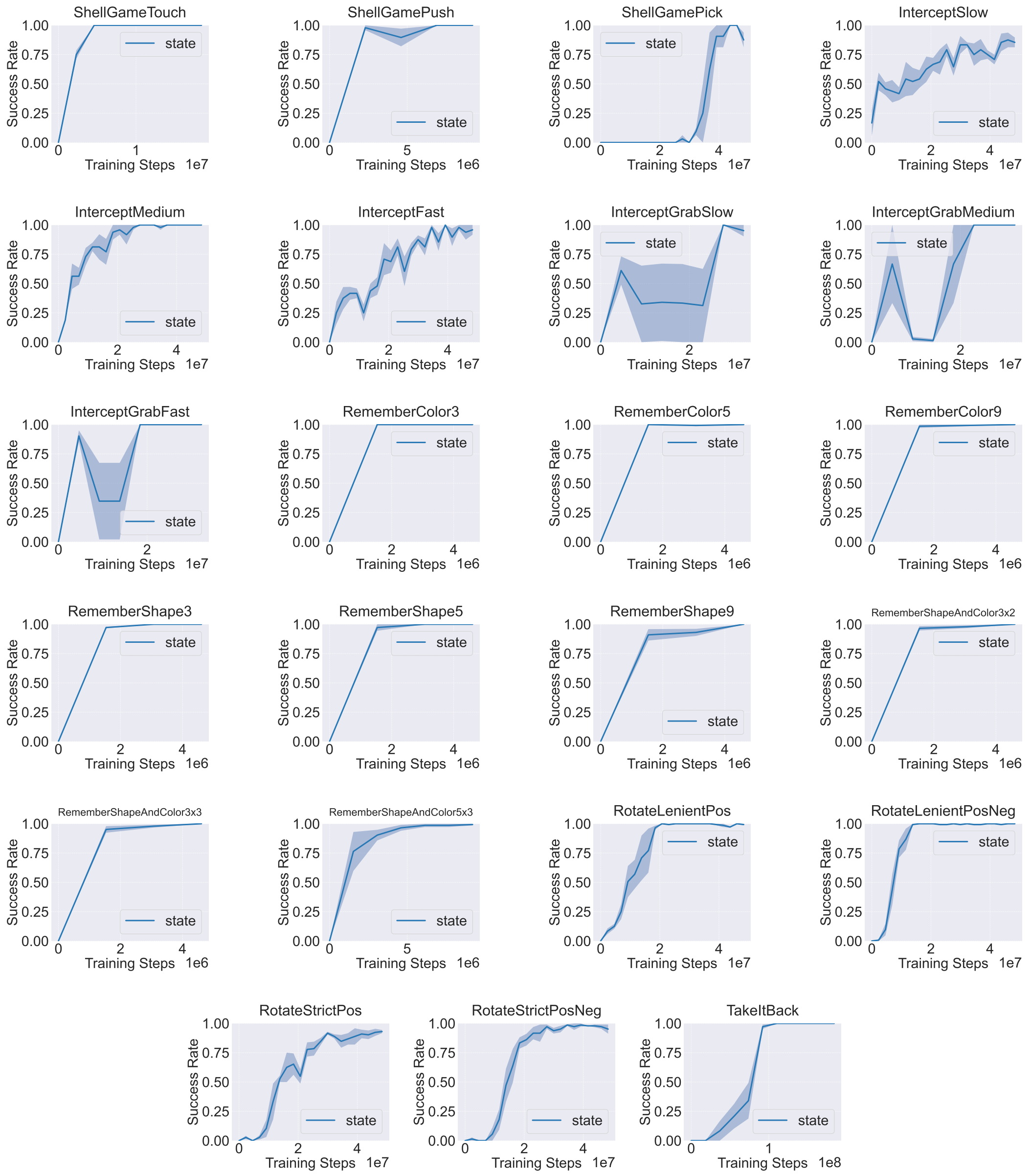

Figure 7: Demonstration of PPO-MLP performance on ManiSkill-Memory benchmark when trained with oracle-level `state` information. In this learning mode, MDP problem formulation is considered, i.e. memory is not required for successful problem solving. At the same time, the obtained results show that it is possible to solve these problems and obtain 100% Success Rate.

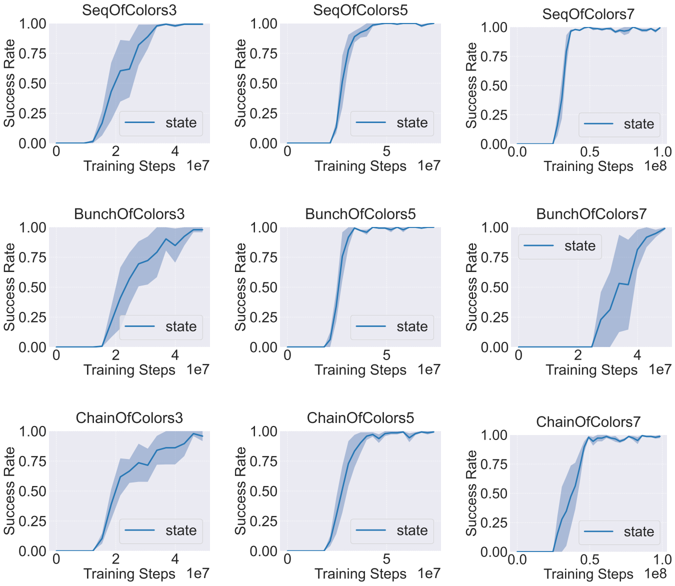

Figure 8: Demonstration of PPO-MLP performance on ManiSkill-Memory benchmark when trained with oracle-level `state` information. Results are shown for memory capacity (`SeqOfColors[3,5,7]-v0`, `BunchOfColors[3,5,7]-v0`) and sequential memory (`ChainOfColors[3,5,7]-v0`).

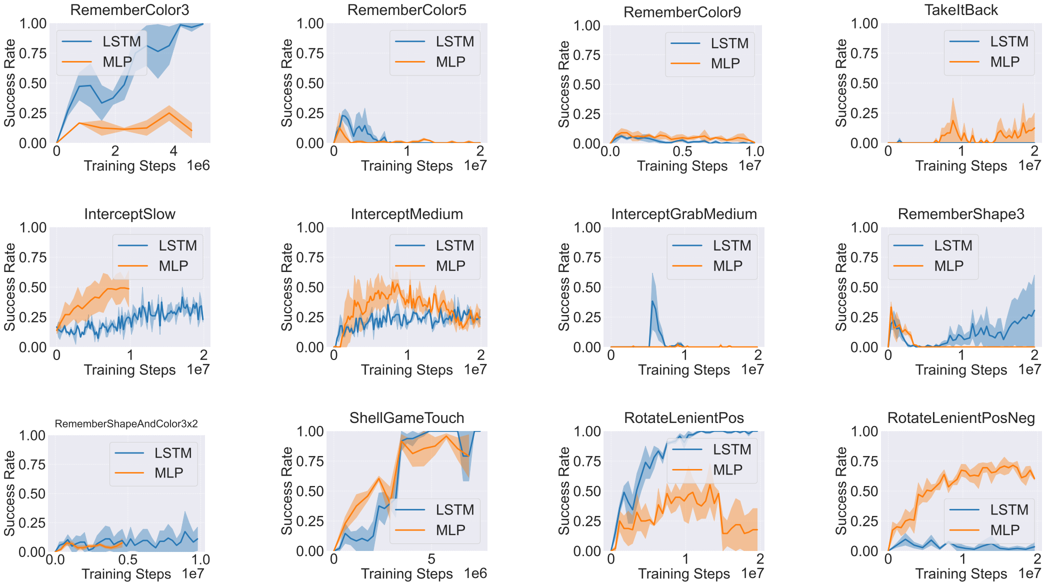

Figure 9: Performance evaluation of PPO-MLP and PPO-LSTM on the ManiSkill-Memory benchmark using the "RGB+joints" training mode with dense reward function, where the agent only receives images from the camera (from above and from the gripper) and information about the state of the joints (position and velocity). The results demonstrate that numerous tasks pose significant challenges even for PPO-LSTM agents with memory, establishing these environments as effective benchmarks for evaluating advanced memory-enhanced architectures.

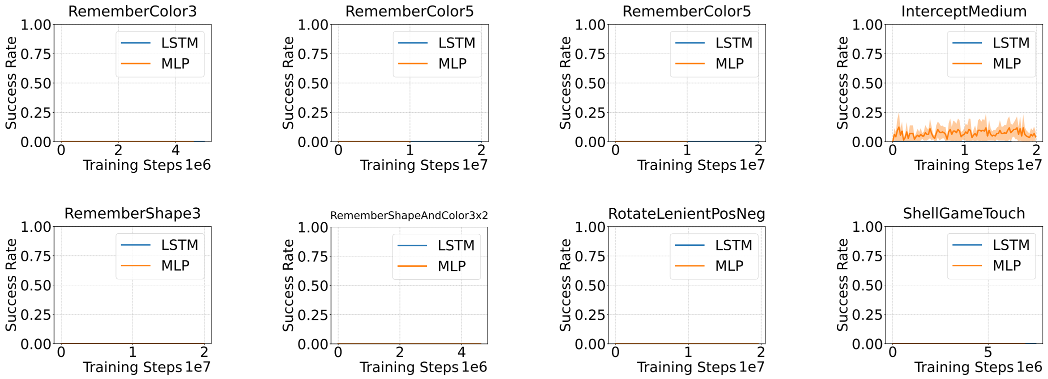

Figure 10: Performance evaluation of PPO-MLP and PPO-LSTM on the ManiSkill-Memory benchmark using the "RGB+joints" with sparse reward function training mode, where the agent only receives images from the camera (from above and from the gripper) and information about the state of the joints (position and velocity). This training mode with sparse reward function causes even more difficulty for the agent to learn, making this mode even more challenging for memory-enhanced agents.

E ManiSkill-Memory Detailed Tasks Description

In this section, we provide comprehensive descriptions of the 32 memory-intensive tasks that comprise the ManiSkill-Memory benchmark. Each task is designed to evaluate specific aspects of memory capabilities in robotic manipulation, ranging from object tracking and spatial memory to sequential decision-making. For each task, we detail its objective, memory requirements, observation space, reward structure, and success criteria. Additionally, we explain how task complexity increases across different variants and discuss the specific memory challenges they present. The following subsections describe each task category and its variants in detail.

Each of the proposed environment supports multiple observation modes:

- **State**: Full state information including ball position
- **RGB+joints**: Two camera views (top-down and gripper) plus robot joint states
- **RGB**: Only visual information from two cameras

In the case of `RotateLenient-v0` and `RotateStrict-v0`, the prompt information available at each step is additionally added to each observation.

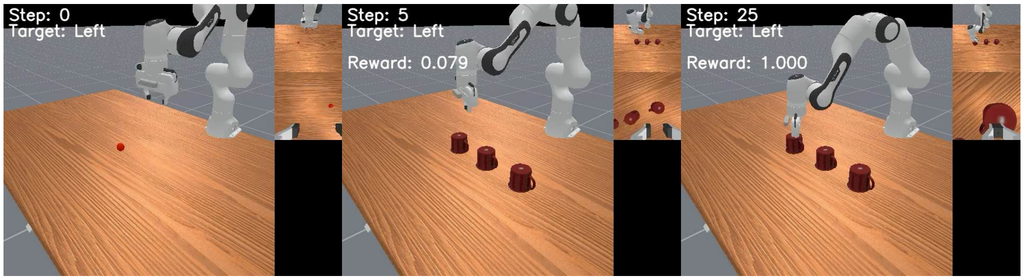

Figure 11: `ShellGameTouch-v0`: The robot observes a ball in front of it. next, this ball is covered by a mug and then the robot has to touch the mug with the ball underneath.

E.1 SHELLGAME-V0

The `ShellGame-v0` task (Figure 11) is inspired by a simplified version of the classic shell game, which tests a person's ability to remember object locations when they become occluded. This task evaluates an agent's capacity for object permanence and spatial memory, crucial skills for real-world robotic manipulation where objects frequently become temporarily hidden from view.

Environment Description The environment consists of three identical mugs placed on a table and a red ball. The task proceeds in three phases:

1. **Observation Phase** (steps 0-4): The ball is placed at one of three positions, and the agent can observe its location.
2. **Occlusion Phase** (step 5): The ball and positions are covered by three identical mugs.
3. **Action Phase** (steps 6+): The agent must interact with the mug covering the ball's location. The type of target interaction depends on the selected mode: `Touch`, `Push` and `Pick`.

Task Modes The task includes three variants of increasing difficulty:

- `Touch`: The agent only needs to touch the correct mug
- `Push`: The agent must push the correct mug to a designated area
- `Pick`: The agent must pick and lift the correct mug above a specified height

Success Criteria Success is determined by:

- `Touch`: Contact between the gripper and the correct mug
- `Push`: Moving forward the correct mug to the target zone
- `Pick`: Elevating the correct mug above 0.1m

Reward Structure The environment provides both sparse and dense reward variants:

- **Sparse**: Binary reward (1.0 for success, 0.0 otherwise)
- **Dense**: Continuous reward based on:
  - Distance between gripper and target mug (reaching reward with tanh scaling)
  - Robot's motion smoothness (static reward based on joint velocities)
  - Task completion status (additional reward when correct mug is reached)

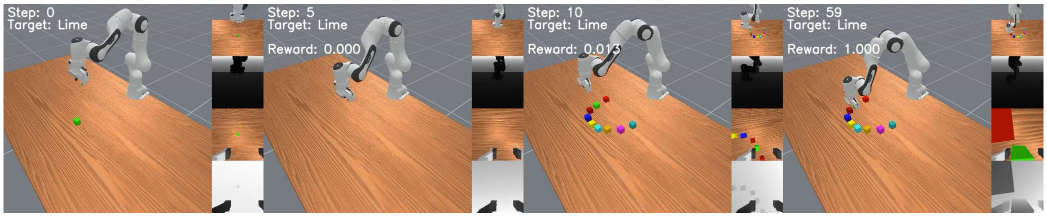

Figure 12: `RememberColor9-v0`: The robot observes a colored cube in front of it, then this cube disappears and an empty table is shown. Then 9 cubes appear on the table, and the agent must touch a cube of the same color as the one it observed at the beginning of the episode.

E.2 REMEMBERCOLOR-V0

The `RememberColor-v0` task (Figure 12) tests an agent's ability to remember and identify objects based on their visual properties. This capability is essential for real-world robotics applications where agents must recall and match object characteristics across time intervals.

Environment Description The environment presents a sequence of colored cubes on a table. The task proceeds in three phases:

1. **Observation Phase** (steps 0-4): A cube of a specific color is displayed, and the agent must memorize its color.
2. **Delay Phase** (steps 5-9): The cube disappears, leaving an empty table.
3. **Selection Phase** (steps 10+): Multiple cubes of different colors appear (3, 5, or 9 depending on difficulty), and the agent must identify and interact with the cube matching the original color.

Task Modes The task includes three complexity levels:

- `3` (easy): Choose from 3 different colors (red, lime, blue)
- `5` (Medium): Choose from 5 different colors (red, lime, blue, yellow, magenta)
- `9` (Hard): Choose from 9 different colors (red, lime, blue, yellow, magenta, cyan, maroon, olive, teal)

Success Criteria Success is determined by:

- Correctly identifying and touching the cube that matches the color shown in the observation phase
- Maintaining contact with the correct cube for at least 0.1 seconds

Reward Structure The environment provides both sparse and dense reward variants:

- **Sparse**: Binary reward (1.0 for success, 0.0 otherwise)
- **Dense**: Continuous reward based on:
  - Reaching reward
  - Static reward
  - Additional reward for robot being static while touching the correct cube

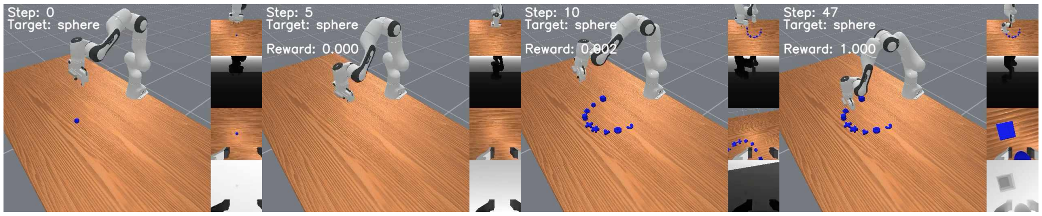

Figure 13: RememberShape9-v0: The robot observes an object with specific shape in front of it, then the object disappears and an empty table appears. Then 9 objects of different shapes appear on the table, and the agent must touch an object of the same shape as the one it observed at the beginning of the episode.

E.3 REMEMBERSHAPE-V0

The RememberShape-v0 task (Figure 13) evaluates an agent's ability to remember and identify objects based on their geometric properties. This capability is crucial for robotic applications where shape recognition and recall are essential for successful manipulation.

Environment Description The environment presents a sequence of geometric shapes on a table. The task proceeds in three phases:

1. **Observation Phase** (steps 0-4): A shape (cube, sphere, cylinder, etc.) is displayed, and the agent must memorize its geometry.
2. **Delay Phase** (steps 5-9): The shape disappears, leaving an empty table.
3. **Selection Phase** (steps 10+): Multiple shapes appear (3, 5, or 9 depending on difficulty), and the agent must identify and interact with the shape matching the original geometry.

Task Modes The task includes three complexity levels:

- 3 (Easy): Choose from 3 different shapes (cube, sphere, cylinder)
- 5 (Medium): Choose from 5 different shapes (cube, sphere, cylinder cross, torus)
- 9 (Hard): Choose from 9 different shapes (cube, sphere, cylinder cross, torus, star, pyramid, t-shape, crescent)

Success Criteria Success is determined by:

- Correctly identifying and touching the object with the same shape shown in the observation phase
- Maintaining contact with the correct shape for at least 0.1 seconds

Reward Structure The environment provides both sparse and dense reward variants:

- **Sparse**: Binary reward (1.0 for success, 0.0 otherwise)
- **Dense**: Continuous reward based on:
  - Reaching reward
  - Static reward
  - Additional reward for maintaining static position when touching correct object

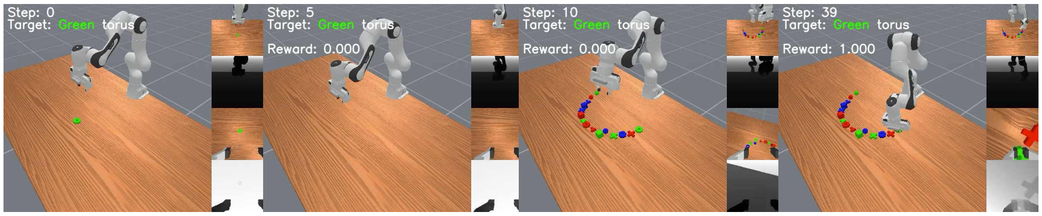

Figure 14: `RememberShapeAndColor5x3-v0`: An object of a certain shape and color appears in front of the agent. Then the object disappears and the agent sees an empty table. Then objects of 5 different shapes and 3 different colors appear on the table and the agent has to touch what it observed at the beginning of the episode.

E.4 REMEMBERSHAPEANDCOLOR-V0

The `RememberShapeAndColor-v0` task (Figure 14) evaluates an agent's ability to remember and identify objects based on multiple visual properties simultaneously. This task combines shape and color recognition, testing the agent's capacity to maintain and match multiple object features across time intervals.

Environment Description The environment presents a sequence of colored geometric shapes on a table. The task proceeds in three phases:

1. **Observation Phase** (steps 0-4): An object with specific shape and color is displayed, and the agent must memorize both properties.
2. **Delay Phase** (steps 5-9): The object disappears, leaving an empty table.
3. **Selection Phase** (steps 10+): Multiple objects with different combinations of shapes and colors appear, and the agent must identify and interact with the object matching both the original shape and color.

Task Modes The task includes three complexity levels based on the number of shape-color combinations:

- `3x2` (Easy): Choose from 6 objects (3 shapes × 2 colors); shapes: cube, sphere, t-shape; colors: red, green
- `3x3` (Medium): Choose from 9 objects (3 shapes × 3 colors); shapes: cube, sphere, t-shape; colors: red, green, blue
- `5x3` (Hard): Choose from 15 objects (5 shapes × 3 colors); shapes: cube, sphere, t-shape, cross, torus; colors: red, green, blue

Success Criteria Success is determined by:

- Correctly identifying and touching the object that matches both the shape and color shown in the observation phase
- Maintaining contact with the correct object for at least 0.1 seconds

Reward Structure The environment provides both sparse and dense reward variants:

- **Sparse**: Binary reward (1.0 for success, 0.0 otherwise)
- **Dense**: Continuous reward based on:
  - Reaching reward
  - Static reward
  - Additional reward for maintaining static position while touching correct object

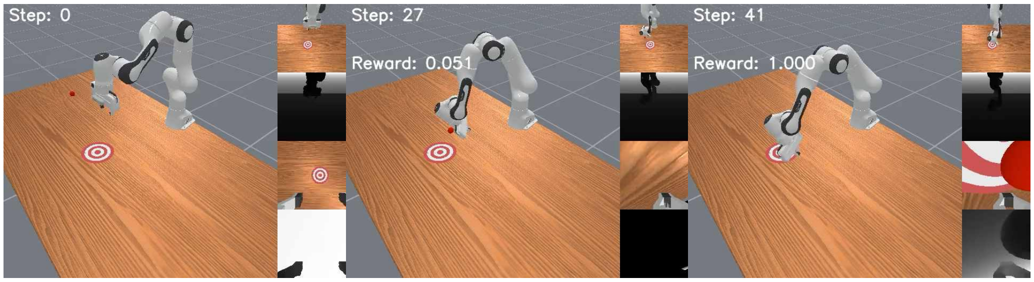

Figure 15: `InterceptMedium-v0`: A ball rolls on the table in front of the agent with a random initial velocity, and the agent's task is to intercept this ball and direct it at the target zone.

E.5 Intercept-v0

The `Intercept-v0` task (Figure 16) evaluates an agent's ability to predict and intercept a moving object based on its initial trajectory. This task tests the agent's capacity for motion prediction and spatial-temporal reasoning, which are essential skills for dynamic manipulation tasks in robotics.

Environment Description The environment consists of a red ball moving across a table and a target zone. The task requires the agent to:

1. Observe the ball's initial position and velocity
2. Predict the ball's trajectory
3. Guide the ball to reach a designated target zone

Task Modes The task includes three variants with increasing ball velocities:

- `Slow`: Ball velocity range of 0.25-0.5 m/s
- `Medium`: Ball velocity range of 0.5-0.75 m/s
- `Fast`: Ball velocity range of 0.75-1.0 m/s

Success Criteria Success is determined by:

- Guiding the ball to enter the target zone
- The ball must come to rest within the target area (radius 0.1m)

Reward Structure The environment provides both sparse and dense reward variants:

- **Sparse**: Binary reward (1.0 for success, 0.0 otherwise)
- **Dense**: Continuous reward based on:
  - Distance between ball and target zone
  - Static reward based on robot joint velocities

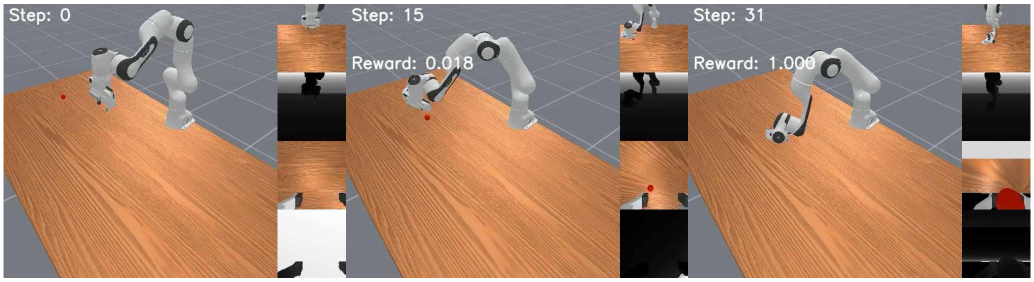

Figure 16: `InterceptGrabMedium-v0`: A ball rolls on the table in front of the agent with a random initial velocity, and the agent's task is to intercept this ball with a gripper and lift it up.

E.6 INTERCEPTGRAB-V0

The `InterceptGrab-v0` task (Figure 16) extends the `Intercept-v0` task by requiring the agent to not only predict the trajectory of a moving object but also grasp it while in motion. This task evaluates the agent's ability to combine motion prediction with precise manipulation timing, simulating real-world scenarios where robots must catch or intercept moving objects.

Environment Description The environment consists of a red ball moving across a table. The task requires the agent to:

1. Observe the ball's initial position and velocity
2. Predict the ball's trajectory
3. Position the gripper to intercept the ball's path
4. Time the grasping action correctly to catch the ball
5. Maintain a stable grasp while bringing the ball to rest

Task Modes The task includes three variants with increasing ball velocities:

- `Slow`: Ball velocity range of 0.25-0.5 m/s
- `Medium`: Ball velocity range of 0.5-0.75 m/s
- `Fast`: Ball velocity range of 0.75-1.0 m/s

Success Criteria Success is determined by:

- Successfully grasping the moving ball
- Maintaining a stable grasp until the ball comes to rest
- The robot must be static with the ball firmly grasped

Reward Structure The environment provides both sparse and dense reward variants:

- **Sparse**: Binary reward (1.0 for success, 0.0 otherwise)
- **Dense**: Continuous reward based on:
  - Reaching reward
  - Grasping reward
  - Static reward

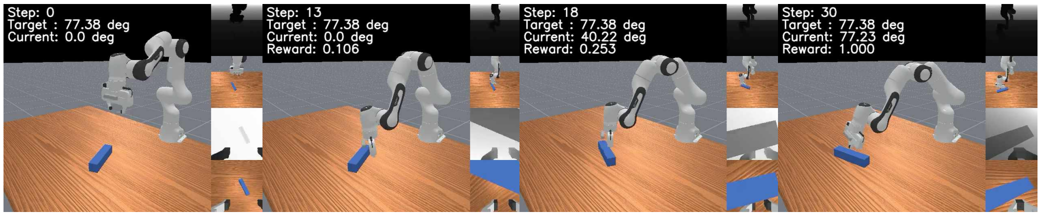

Figure 17: `RotateLenientPos-v0`: A randomly oriented peg is placed in front of the agent. The agent's task is to rotate this peg by a certain angle (the center of the peg can be shifted).

E.7 ROTATELENIENT-V0

The `RotateLenient-v0` task (Figure 17) evaluates an agent's ability to remember and execute specific rotational movements. This task tests the agent's capacity to maintain and reproduce angular information, which is crucial for manipulation tasks requiring precise orientation control. This task tests the agent's ability to hold information in memory about how far peg has already rotated at the current step relative to its initial position.

Environment Description The environment consists of a blue-colored peg on a table that must be rotated by a specified angle. The task proceeds in one phase, but the static prompt information about the target angle is available to the agent at each timestep:

1. **Action Phase**: The agent must rotate the peg to match the target angle

Task Modes The task includes two variants with different rotation requirements:

- `Pos`: Rotate by a positive angle between 0 and $\pi/2$
- `PosNeg`: Rotate by either positive or negative angle between $-\pi/4$ and $\pi/4$

Success Criteria Success is determined by:

- Rotating the peg to within the angle threshold (±0.1 radians) of the target angle
- Maintaining the final orientation in a stable position
- The robot must be static with the peg at the correct orientation

Reward Structure The environment provides both sparse and dense reward variants:

- **Sparse**: Binary reward (1.0 for success, 0.0 otherwise)
- **Dense**: Continuous reward based on:
  - Angular distance to target rotation
  - Stability of the peg's orientation
  - Robot's motion smoothness

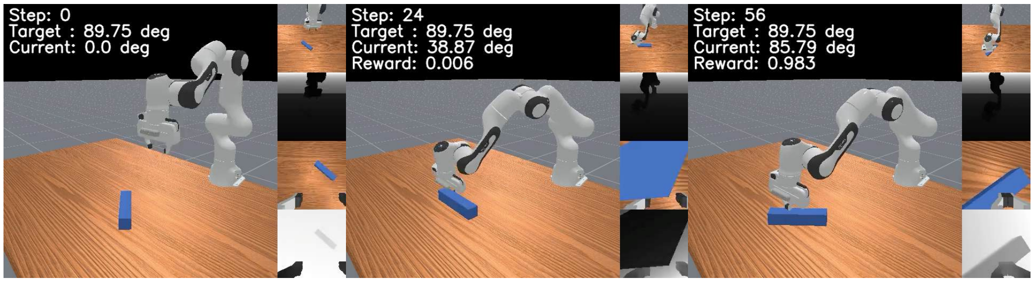

Figure 18: `RotateStrictPos-v0`: A randomly oriented peg is placed in front of the agent. The agent's task is to rotate this peg by a certain angle (it is not allowed to move the center of the peg)

E.8 RotateStrict-v0

The `RotateStrict-v0` task (Figure 18) extends the `RotateLenient-v0` task with more stringent requirements for precise rotational control.

Environment Description The environment consists of a blue-colored peg on a table that must be rotated by a specified angle while maintaining its position. The task proceeds in one phase, but the static prompt information about the target angle is available to the agent at each timestep:

1. **Action Phase**: The agent must rotate the peg to match the target angle while keeping it centered

Task Modes The task includes two variants with different rotation requirements:

- `Pos`: Rotate by a positive angle between 0 and $\pi/2$
- `PosNeg`: Rotate by either positive or negative angle between $-\pi/4$ and $\pi/4$

Success Criteria Success is determined by:

- Rotating the peg to within the angle threshold (±0.1 radians) of the target angle
- Maintaining the peg's position within 5cm of its initial XY coordinates
- The robot must be static with the peg at the correct orientation
- No significant deviation in other rotation axes

Reward Structure The environment provides both sparse and dense reward variants:

- **Sparse**: Binary reward (1.0 for success, 0.0 otherwise)
- **Dense**: Continuous reward based on:
  - Angular distance to target rotation
  - Position deviation from initial location
  - Stability of the peg's orientation
  - Robot's motion smoothness

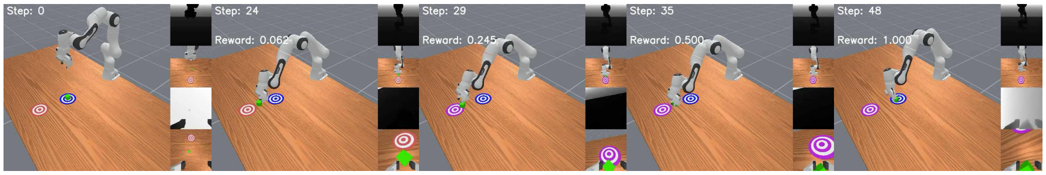

Figure 19: `TakeItBack-v0`: The agent observes a green cube in front of him. The agent's task is to move the green cube to the red target, and as soon as it lights up violet, return the cube to its original position (the agent does not observes the original position of the cube).

E.9 TakeItBack-v0

The `TakeItBack-v0` task (Figure 19) assesses the agent's ability to perform sequential tasks and memorize the starting position. This task tests the agent's capacity for sequential memory and spatial reasoning, requiring it to maintain information about past locations and achievements while executing a multi-step plan.

Environment Description The environment consists of a green cube and two target regions (initial and goal) on a table. The task proceeds in two phases:

1. **First Phase**: The agent must move the cube from its initial position to a goal region
2. **Second Phase**: After reaching the goal, goal region change it's color from red to magenta, and the agent must return the cube to its original position (marked by the initial region and invisible for the agent)

Success Criteria Success is determined by:

- First reaching the goal region with the cube
- Then returning the cube to the initial region
- Both goals must be achieved in sequence

Reward Structure The environment provides both sparse and dense reward variants:

- **Sparse**: Binary reward (1.0 for success, 0.0 otherwise)
- **Dense**: Continuous reward based on:
  - Distance to current target region
  - Progress through the task sequence
  - Stability of cube manipulation

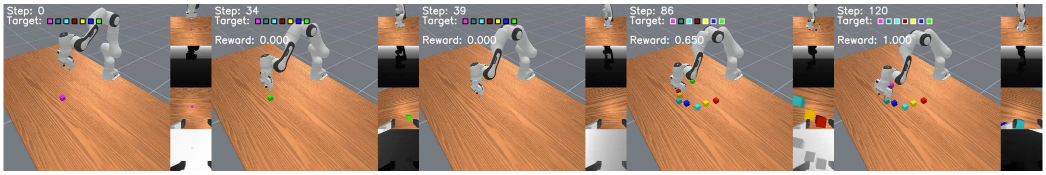

Figure 20: `SeqOfColors7-v0`: In front of the agent, 7 cubes of different colors appear sequentially. After the last cube is shown, the agent observes an empty table. Then 9 cubes of different colors appear on the table and the agent has to touch the cubes that were shown at the beginning of the episode in any order.

E.10 SeqOfColors-v0

The `SeqOfColors-v0` task (Figure 20) evaluates an agent's ability to remember and reproduce an unordered sequence of colors. This task tests memory capacity capabilities essential for robotic tasks that require following specific patterns or sequences.

Environment Description The environment presents a sequence of colored cubes that must be reproduced in any order. The task proceeds in two phases:

1. **Observation Phase** (steps 0-($5N - 1$)): A sequence of N colored cubes is shown one at a time, with each cube visible for 5 steps.
2. **Delay Phase** (steps ($5N$)-($5N + 4$)): All cubes disappear
3. **Selection Phase** (steps ($5N + 5$)+): A larger set of cubes appears, and the agent must identify and touch all previously shown cubes in any order

Task Modes The task includes three complexity levels:

- `3` (Easy): Remember 3 colors demonstrated sequentially
- `5` (Medium): Remember 5 colors demonstrated sequentially
- `7` (Hard): Remember 7 colors demonstrated sequentially

Success Criteria Success is determined by:

- Correctly identifying and touching all cubes from the observation phase
- Order of selection doesn't matter
- Each cube must be touched for at least 0.1 seconds
- The demonstrated set must be touched without any mistakes

Reward Structure The environment provides both sparse and dense reward variants:

- **Sparse**: Binary reward (1.0 for success, 0.0 otherwise)
- **Dense**: Continuous reward based on:
  - Number of correctly identified cubes
  - Reaching reward for current interaction
  - Static reward for stable touches

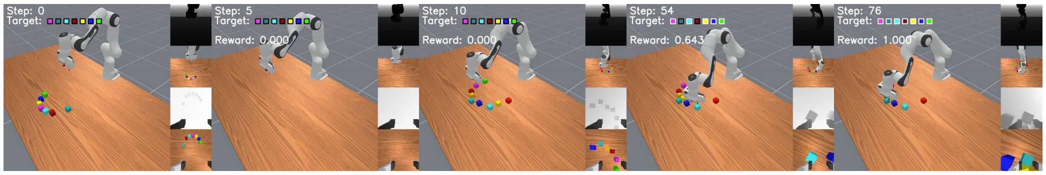

Figure 21: `BunchOfColors7-v0`: 7 cubes of different colors appear simultaneously in front of the agent. After the agent observes an empty table. Then, 9 cubes of different colors appear on the table and the agent has to touch the cubes that were shown at the beginning of the episode in any order.

E.11 BUNCHOFCOLORS-V0

The `BunchOfColors-v0` task (Figure 21) tests an agent's memory capacity by requiring it to remember multiple objects simultaneously. This capability is crucial for tasks requiring parallel processing of multiple items.

Environment Description The environment presents multiple colored cubes simultaneously. The task proceeds in three phases:

1. **Observation Phase** (steps 0-4): Multiple colored cubes are displayed simultaneously
2. **Delay Phase** (steps 5-9): All cubes disappear
3. **Selection Phase** (steps 10+): A larger set of cubes appears, and the agent must identify and touch all previously shown cubes in any order

Task Modes The task includes three complexity levels:

- `3` (Easy): Remember 3 colors demonstrated simultaneously
- `5` (Medium): Remember 5 colors demonstrated simultaneously
- `7` (Hard): Remember 7 colors demonstrated simultaneously

Success Criteria Success is determined by:

- Correctly identifying and touching all cubes from the observation phase
- Order of selection doesn't matter
- Each cube must be touched for at least 0.1 seconds
- The demonstrated set must be touched without any mistakes

Reward Structure The environment provides both sparse and dense reward variants:

- **Sparse**: Binary reward (1.0 for success, 0.0 otherwise)
- **Dense**: Continuous reward based on:
  - Number of correctly identified cubes
  - Reaching reward for current interaction
  - Static reward for stable touches

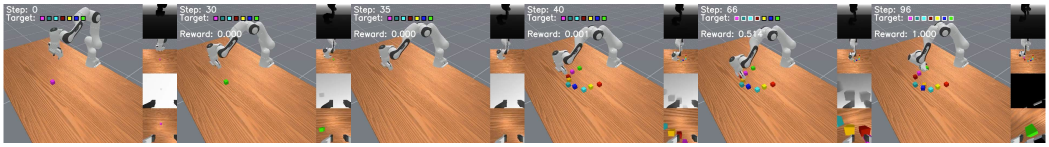

Figure 22: `ChainOfColors7-v0`: In front of the agent, 7 cubes of different colors appear sequentially. After the last cube is shown, the agent sees an empty table. Then 9 cubes of different colors appear on the table and the agent must unmistakably touch the cubes that were shown at the beginning of the episode, in the same strict order.

E.12 CHAINOFCOLORS-V0

The `ChainOfColors-v0` task (Figure 22) evaluates the agent's ability to store and retrieve ordered information. This task simulates scenarios where the agent must track changing relationships between objects over time.

Environment Description The environment presents am ordered sequence (chain) of colored cubes that must be followed. The task proceeds in multiple phases:

1. **Observation Phase** (steps 0-($5N-1$)): A sequence of N colored cubes is shown one at a time, with each cube visible for 5 steps.
2. **Delay Phase** (steps ($5N$)-($5N+4$)): All cubes disappear
3. **Selection Phase** (steps ($5N+5$)+): A larger set of cubes appears, and the agent must identify and touch all previously shown cubes in the exact order as demonstrated

Task Modes The task includes three complexity levels:

- `3` (Easy): Remember 3 colors demonstrated sequentially
- `5` (Medium): Remember 5 colors demonstrated sequentially
- `7` (Hard): Remember 7 colors demonstrated sequentially

Success Criteria Success is determined by:

- Correctly identifying and touching all cubes from the observation phase in the exact order
- Each cube must be touched for at least 0.1 seconds
- The demonstrated set must be touched without any mistakes

Reward Structure The environment provides both sparse and dense reward variants:

- **Sparse**: Binary reward (1.0 for success, 0.0 otherwise)
- **Dense**: Continuous reward based on:
  - Reaching reward for current interaction
  - Static reward for stable contact
  - Additional reward for selecting correct final cube

F CLASSIC BASELINES PERFORMANCE ON THE MIKASA BENCHMARK

In this section, we evaluate the performance of standard reinforcement learning baselines on the proposed benchmark. We utilize PPO-MLP and PPO-LSTM. PPO-MLP serves as a baseline model without memory, while PPO-LSTM incorporates recurrent layers, allowing it to retain past information and effectively handle memory intensive environments.

Figure 23 presents the results of the test of these baselines in selected benchmark environments. The performance gap between PPO-MLP and PPO-LSTM is evident, with the latter consistently achieving higher score in memory-dependent tasks. This discrepancy confirms that the benchmark effectively evaluates memory capabilities, as environments requiring information retention challenge models without memory mechanisms.

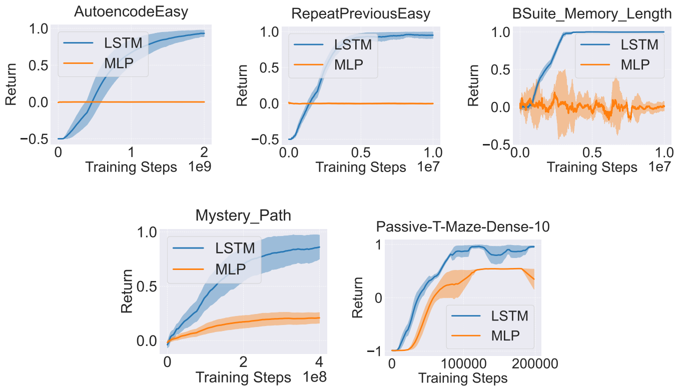

Figure 23: Performance evaluation of PPO-MLP and PPO-LSTM on the MIKASA benchmark

Table 5: Comparison of various memory-intensive tasks across different environments.

Environment	Memory Task	Brief description of the task	Observation Space	Action Space
Memory Cards	Capacity	Memorize the positions of revealed cards and correctly match pairs while minimizing incorrect guesses.	state	discrete
ViZDoom-two-colors	Object	Memorize the color of the briefly appearing pillar (green or red) and collect items of the same color to survive in the acid-filled room.	img	discrete
BSuite Memory Length	Object	Memorize the initial context signal and recall it after a given number of steps to take the correct action.	state	discrete
Gym Gridverse Memory	Spatial, Sequential, Object	Memorize the object in the starting room and use this information to select the correct path at the junction.	img	discrete
Memory Maze	Spatial	Memorize the locations of objects and the maze structure using visual clues, then navigate efficiently to find objects of a specific color and score points.	img	discrete
Ballet	Sequential, Object	Memorize the sequence of movements performed by each uniquely colored and shaped dancer, then identify and approach the dancer who executed the given pattern.	img	discrete
Numpad	Sequential	Memorize the sequence of movements and navigate the rolling ball on a 3×3 grid by following the correct order while avoiding mistakes.	img, state	discrete, continuous
MinigridMemory	Object	Memorize the object in the starting room and use this information to select the correct path at the junction.	img	discrete
Passive-T-Maze	Object	Memorize the goal's location upon initial observation, navigate through the maze with limited sensory input, and select the correct path at the junction.	state	discrete
POPGym Repeat First	Object	Memorize the initial value presented at the first step and recall it correctly after receiving a sequence of random values.	state	discrete
POPGym Repeat Previous	Sequential, Object	Memorize the value observed at each step and recall the value from k steps earlier when required.	state	discrete
POPGym Autoencode	Sequential	Memorize the sequence of cards presented at the beginning and reproduce them in the same order when required.	state	discrete
POPGym Count Recall	Object, Capacity	Memorize unique values encountered and count how many times a specific value has appeared.	state	discrete
POPGym Stateless Cartpole	Sequential	Memorize velocity data over time and integrate it to infer the position of the pole for balance control.	state	continuous
POPGym Stateless Pendulum	Sequential	Memorize angular velocity over time and integrate it to infer the pendulum's position for successful swing-up control.	state	continuous
POPGym Multiarmed Bandit	Object, Capacity	Memorize the reward probabilities of different slot machines by exploring them and identify the one with the highest expected reward.	state	discrete
POPGym Concentration	Capacity	Memorize the positions of revealed cards and match them with previously seen cards to find all matching pairs.	state	discrete
POPGym Battleship	Spatial	Memorize the coordinates of previous shots and their HIT or MISS feedback to build an internal representation of the board, avoid repeat shots, and strategically target ships for maximum rewards.	state	discrete
POPGym Mine Sweeper	Spatial	Memorize revealed grid information and use numerical clues to infer safe tiles while avoiding mines.	state	discrete
POPGym Labyrinth Explore	Spatial	Memorize previously visited cells and navigate the maze efficiently to discover new, unexplored areas and maximize rewards.	state	discrete
POPGym Labyrinth Escape	Spatial	Memorize the maze layout while exploring and navigate efficiently to find the exit and receive a reward.	state	discrete
POPGym Higher Lower	Object, Sequential	Memorize previously revealed card ranks and predict whether the next card will be higher or lower, updating the reference card after each prediction to maximize rewards.	state	discrete

G MIKASA BENCHMARK TASKS DESCRIPTION

This section provides a detailed description of all environments included in the MIKASA benchmark section 5. Understanding the characteristics and challenges of these environments is crucial for evaluating RL algorithms. Each environment presents unique tasks, offering diverse scenarios to test the memory abilities of RL agents.

G.1 MEMORY CARDS

The Memory Cards environment (Esslinger et al., 2022) is a memory game environment with 5 randomly shuffled pairs of hidden cards. At each step, the agent sees one revealed card and must find its matching pair. A correct guess removes both cards; otherwise, the card is hidden again, and a new one is revealed. The game continues until all pairs are removed. The observation space is a 10-element vector indicating card states (hidden, revealed, removed). The action space consists of selecting an index corresponding to the current revealed card. Rewards: 0 for correct guesses, -1 for mistakes. Policies that effectively remember past observations perform best.

G.2 NUMPAD

The Numpad environment (Humplik et al., 2019) consists of an $N \times N$ grid of tiles. The agent controls a ball that rolls between tiles. At the beginning of an episode, a random sequence of n neighboring tiles (excluding diagonals) is selected, and the agent must follow this sequence in the correct order. The environment is structured so that pressing the correct tile lights it up, while pressing an incorrect tile resets progress. A reward of +1 is given for the first press of each correct tile after a reset. The episode ends after a fixed number of steps. To succeed, the agent must memorize the sequence and navigate it correctly without mistakes. The ability to "jump" over tiles is not available.

G.3 HALLWAY

The Hallway environment (Littman et al., 1995) is a gridworld with four rooms aligned to the south. The agent's goal is to reach the fourth southern room despite stochastic transitions. The environment provides an integer-based observation indicating visible walls in the agent's current position. The agent has five possible actions: no-op, move forward, turn left, turn right, and turn around. To succeed, the agent must localize itself through observations and navigate effectively. Rewards are 0 for movement and 1 for reaching the goal.

G.4 HEAVEN HELL

The Heaven Hell environment (Geffner & Bonet, 1998) is a T-shaped grid with a priest at the southern end. The two northern forks represent heaven and hell, but their locations are randomized each episode. The agent must visit the priest to learn heaven's position. Observations are integers indicating position, except when consulting the priest, who provides a hint about heaven's location. The agent can move north, south, east, or west. To succeed, the agent must first visit the priest and then navigate correctly. It receives a reward of 1 for reaching heaven and -1 for reaching hell.

G.5 BSUITE MEMORYLENGTH

The MemoryLength environment (Osband et al., 2020) represents a sequence of observations, where at each step, the observation obs takes a value of either +1 or -1. The environment is structured so that a reward is given only at the final step if the agent correctly predicts the i-th value from the initial observation. The index of this i-th value is specified at the last step observation in $obs[1]$. To succeed, the agent must remember the sequence of observations and use this information to make an accurate prediction at the final step.

G.6 MINIGRID-MEMORY

Minigrid-Memory (Chevalier-Boisvert et al., 2023) is a two-dimensional grid-based environment that features a T-shaped maze with a small room at the beginning of the corridor, containing an object. The agent starts at a random position within the corridor. Its task is to reach the room, observe and memorize the object, then proceed to the junction at the maze's end and turn towards the direction where an identical object is located. The reward function is defined as $R_t = 1 - 0.9 \times \frac{t}{T}$ for a successful attempt; otherwise, the agent receives zero reward. The episode terminates when the agent makes a choice at the junction or exceeds a time limit of 95 steps. To enforce partial observability, the agent's vision is restricted to a 3×3 frame. Consequently, this environment features a two-dimensional space of image observations, a discrete action space, and a sparse reward function.

G.7 BALLET

In the Ballet environment (Lampinen et al., 2021) tasks take place in an 11×11 tiled room, consisting of a 9×9 central area surrounded by a one-tile-wide wall. Each tile is upsampled to 9 pixels, resulting in a 99 × 99 pixel input image. The agent is initially placed at the center of the room, while dancers are randomly positioned in one of 8 possible locations around it. Each dancer has a distinct shape and color, selected from 15 possible shapes and 19 colors, ensuring uniqueness. These visual features serve only for identification and do not influence behavior. The agent itself is always represented as a white square. The agent receives egocentric visual observations, meaning its view is centered on its own position, which has been shown to enhance generalization.

G.8 PASSIVE VISUAL MATCH

The Passive Visual Match environment (Ni et al., 2023) is a color recognition task where the agent must memorize a target color and select it among distractors. In each episode, four colors are randomly chosen from a set of 16, with one as the target and the other three as distractors. These colors form four squares, each occupying one wall unit.

The episode consists of three phases. First, the agent is placed in a 1×3 corridor, facing a wall with the target color at the opposite end. There are no rewards, and this phase lasts for 5 seconds. Next, the agent encounters a distractor phase. Finally, the environment expands to 4×7 with the four color squares aligned on one side in random order. The agent spawns in the center of the opposite side, facing them. In front of each square is a ground pad. Stepping on the pad in front of the target color grants 10 points, while any other pad gives 1 point. If no choice is made within 5 seconds, no reward is given.

G.9 Passive-T-Maze

The Passive-T-Maze environment (Ni et al., 2023) is a T-shaped maze with a four-directional action space $\{L, R, U, D\}$. It consists of a corridor of length L, starting at state O (oracle) and ending at state J (junction), which branches into two possible goal states G_1 and G_2. The agent observes only at states $\{J, O, G_1, G_2\}$, with O revealing the goal $G \in \{G_1, G_2\}$ at the start of an episode. The agent's movement is deterministic, the agent remains static upon hitting a wall.

In the passive T-Maze, the oracle state O is equivalent to the start state S, allowing the agent to immediately observe the goal position G. The length of the corridor is set to $L = T - 1$. The reward function is defined as:

$$R_t(h_{1:t}, a_t) = \frac{\mathbb{1}(x_{t+1} \geq t) - 1}{T - 1} \quad \text{for } t \leq T - 1, \quad \text{and} \quad R_T(h_{1:T}, a_T) = \mathbb{1}(o_{T+1} = G). \tag{1}$$

The optimal policy moves right for $T - 1$ steps and then heads to G, achieving an expected return of 1.0. A Markovian policy, which can only guess the goal, yields an expected return of 0.5, while the worst policy results in -1.0.

G.10 ViZDoom-Two-Colors

The ViZDoom-Two-Colors (Sorokin et al., 2022) is a reinforcement learning environment where an agent is placed in a room with constantly depleting health. The room contains red and green objects, one of which restores health (+1 reward), while the other reduces it (-1 reward). The beneficial color is randomly assigned at the beginning of each episode and indicated by a column. The environment is structured so that the agent must memorize the column's color to collect the correct items. Initially, the column remains visible, but in a harder variant, it disappears after 45 steps, increasing the memory requirement. To succeed, the agent must maximize survival by collecting beneficial objects while avoiding harmful ones.

G.11 POPGym environments

The following environments are included from the POPGym benchmark (Morad et al., 2023), which is designed to evaluate RL agents in partially observable settings. POPGym provides a diverse collection of lightweight vectorized environments with varying difficulty levels.

G.11.1 POPGym Autoencode

The environment consists of a deck of cards that is shuffled and sequentially shown to the agent during the watch phase. While observing the cards, a watch indicator is active, but it disappears when the last card is revealed. Afterward, the agent must reproduce the sequence of cards in the correct order. The environment is structured to evaluate the agent's ability to encode a sequence of observations into an internal representation and later reconstruct the sequence one observation at a time.

G.11.2 POPGym Concentration

The environment represents a classic memory game where a shuffled deck of cards is placed face-down. The agent sequentially flips two cards and earns a reward if the revealed cards form a matching pair. The environment is designed in such a way that the agent must remember previously revealed cards to maximize its success rate.

G.11.3 POPGym Repeat First

The environment presents the agent with an initial value from a set of four possible values, along with an indicator signaling that this is the first value. In subsequent steps, the agent continues to receive random values from the same set but without the initial indicator. The structure requires the agent to retain the first received value in memory and recall it accurately to receive a reward.

G.11.4 POPGym Repeat Previous

The environment consists of a sequence of observations, where each observation can take one of four possible values at each timestep. The agent is tasked with recalling and outputting the value that appeared a specified number of steps in the past.

G.11.5 POPGym Stateless Cartpole

This is a modified version of the traditional Cartpole environment (Barto et al., 1983) where angular and linear position information is removed from observations. Instead, the agent only receives velocity-based data and must infer positional states by integrating this information over time to successfully balance the pole.

G.11.6 POPGym Stateless Pendulum

In this variation of the swing-up pendulum environment (Doya, 1995), angular position data is omitted from the agent's observations. The agent must infer the pendulum's position by processing velocity information and use this estimate to determine appropriate control actions.

G.11.7 POPGym Noisy Stateless Cartpole

This environment builds upon Stateless Cartpole by introducing Gaussian noise into the observations. The agent must still infer positional states from velocity information while filtering out the added noise to maintain control of the pole.

G.11.8 POPGym Noisy Stateless Pendulum

This variation extends the Stateless Pendulum environment by incorporating Gaussian noise into the observations. The agent must manage this uncertainty while using velocity data to estimate the pendulum's position and swing it up effectively.

G.11.9 POPGym Multiarmed Bandit

The Multiarmed Bandit environment is an episodic formulation of the multiarmed bandit problem (Slivkins, 2024), where a set of bandits is randomly initialized at the start of each episode. Unlike conventional multiarmed bandit tasks, where reward probabilities remain fixed across episodes, this structure resets them every time. The agent must dynamically adjust its exploration and exploitation strategies to maximize long-term rewards.

G.11.10 POPGym Higher Lower

Inspired by the higher-lower card game, this environment presents the agent with a sequence of cards. At each step, the agent must predict whether the next card will have a higher or lower rank than the current one. Upon making a guess, the next card is revealed and becomes the new reference. The agent can enhance its performance by employing card counting strategies to estimate the probability of future values.

G.11.11 POPGym Count Recall

At each timestep, the agent is presented with two values: a next value and a query value. The agent must determine and output how many times the query value has appeared so far. To succeed, the agent must maintain an accurate count of past occurrences and retrieve the correct number upon request. This environment evaluates the agent's capacity to form and manage a structured memory representation.

G.11.12 POPGym Battleship

A partially observable variation of the game Battleship, where the agent does not have access to the full board. Instead, it receives feedback on its previous shot, indicating whether it was a HIT or MISS, along with the shot's location. The agent earns rewards for hitting ships, receives no reward for missing, and incurs a penalty for targeting the same location more than once. The environment challenges the agent to construct an internal representation of the board and update its strategy based on past observations.

G.11.13 POPGym Mine Sweeper

A partially observable version of the computer game Mine Sweeper, where the agent lacks direct visibility of the board. Observations include the coordinates of the most recently clicked tile and the number of adjacent mines. Clicking on a mined tile results in a negative reward and ends the game. To succeed, the agent must track previous selections and deduce mine locations based on the numerical clues, ensuring it avoids mines while uncovering safe tiles.

G.11.14 POPGym Labyrinth Explore

The environment consists of a procedurally generated 2D maze in which the agent earns rewards for reaching new, unexplored tiles. Observations are limited to adjacent tiles, requiring the agent to infer the larger maze layout through exploration. A small penalty per timestep incentivizes efficient navigation and discovery strategies.

G.11.15 POPGym Labyrinth Escape

This variation of Labyrinth Explore challenges the agent to find an exit rather than merely exploring the maze. The agent retains the same restricted observation space, seeing only nearby tiles. Rewards are only given upon successfully reaching the exit, making it a sparse reward environment where the agent must navigate strategically to achieve its goal.