

Extended Abstract

Motivation Robotic object manipulation systems deployed in real-world environments can often encounter intermittent and unreliable observations due to occlusions, sensor malfunctions, and environmental variability. These challenges violate the assumptions of full state observability made by many reinforcement learning (RL) methods and degrade policy performance during deployment. Classic solutions using Partially Observable Markov Decision Processes (POMDPs) require belief state estimation but they are computationally intractable for high-dimensional, continuous control tasks. While modern RL integrates sequential architectures to approximate temporal memory, most existing approaches fail to efficiently address the intermittent POMDPs. This project investigates explicit memory modeling with multi-task learning, aiming for robust policy performance.

Method We propose MEMBOT, a modular architecture designed for RL under intermittent POMDPs. MEMBOT separates observation history processing from task-specific control by introducing a shared memory-based task-agnostic latent belief encoder and task-specific actor-critic heads. We use two models, an LSTM model and a state-space model (SSM), for the belief encoder, to integrate intermittent observations into a latent state representation. This encoder is first trained offline using expert demonstrations, combining behavior cloning with a reconstruction loss to ensure that the belief state retains semantically meaningful information even under dropout. During the online phase, the encoder is fine-tuned with task-specific Soft Actor-Critic (SAC) policies, accounting for any distribution shift. Dropout of observations is simulated by randomly masking observations at training and test time to evaluate robustness.

Implementation The observation encoder consists of two modules, including a multi-layer perceptron (MLP) adapted to low-dimensional state inputs for the memory-less architecture and a memory-based architecture, implemented with an LSTM and an SSM. We use a two-phase training pipeline: offline multi-task pretraining across 5 tasks on 60 demonstration episodes per task using the Adam optimizer, followed by online transfer learning fine-tuning with SAC over 100,000 steps. Both the encoder and policy are updated jointly during reinforcement learning, and evaluation spans 5 MetaWorld manipulation tasks and 5 Robomimic environments under various observation dropout rates ranging from 0.5 to 1.0.

Results MEMBOT with SSM significantly outperforms memoryless and naive LSTM baselines across all tasks and dropout conditions. In scenarios with 50% observation availability, MEMBOT retains 65–80% of its full-observation success rate and rewards, while baseline models degrade to 10–30%. Ablation studies show that removing the reconstruction loss or recurrent memory module leads to big performance drops under missing observations. Detailed breakdowns show task-specific sensitivity to observability, with contact-rich tasks (e.g., drawer-close) being more robust than spatial tasks (e.g., plate-slide), validating the importance of temporal reasoning in complex robot control.

Discussion Our results show that explicit memory, modularity and multi-task learning are critical for robust RL under intermittent POMDPs. MEMBOT’s decoupled architecture promotes both robustness and transfer, allowing the same belief encoder to generalize across tasks and observation regimes. The inclusion of a reconstruction loss regularizes the encoder and improves belief quality during occlusions. Memory-based inference significantly improves sample efficiency and downstream performance, even under severe sensor dropout. Limitations include the fixed memory size of the LSTM and state-space model, and challenges with very long horizons, suggesting future work on more adaptive or hierarchical memory structures. Real-world deployment and visual input integration remain promising directions.

Conclusion MEMBOT shows that explicit memory modeling and modular architectures with multi-task learning can significantly improve robotic RL performance under intermittent POMDPs. We combine offline multi-task learning pretraining with online fine-tuning to achieve robust performance even when 50% of observations are missing. Our experiments using MetaWorld (x5) and Robomimic tasks (x5) show that the shared belief encoder using state-space models and LSTM with reconstruction loss generalizes very well across different manipulation tasks while maintaining task-specific control capabilities. We also see limitations of scaling to longer horizons or visual inputs which remain to be further addressed in our future work.

MEMBOT: Memory-Based Robot in Intermittent POMDP

Eyan Noronha

Department of Computer Science
Stanford University
eyannoro@stanford.edu

Yousef (Youzhi) Liang

Department of Computer Science
Stanford University
youzhil@stanford.edu

Abstract

Robotic systems deployed in real-world environments often operate under conditions of partial and often intermittent observability, where sensor inputs may be noisy, occluded, or entirely unavailable due to failures or environmental constraints. Traditional reinforcement learning (RL) approaches that assume full state observability are ill-equipped for such challenges. In this work, we introduce MEMBOT, a modular memory-based architecture designed to address intermittent partial observability in robotic control tasks. MEMBOT decouples belief inference from policy learning through a two-phase training process: an offline multi-task learning pretraining stage that learns a robust task-agnostic latent belief encoder using a reconstruction losses, followed by fine-tuning of task-specific policies using behavior cloning. The belief encoder, implemented as a state-space model (SSM) and a LSTM, integrates temporal sequences of observations and actions to infer latent state representations that persist even when observations are dropped. We train and evaluate MEMBOT on 10 robotic manipulation benchmark tasks from MetaWorld and Robomimic under varying rates of observation dropout. Results show that MEMBOT consistently outperforms both memoryless and naively recurrent baselines, maintaining up to 80% of peak performance under 50% observation availability. These findings highlight the effectiveness of explicit belief modeling in achieving robust, transferable, and data-efficient policies for real-world partially observable robotic systems.

1 Introduction

Autonomous decision-making in robotics often confronts a fundamental challenge: the world is only partially observable. Real-world robots must operate with sensors that are frequently noisy, incomplete, or unreliable—due to occlusion, limited fields of view, bandwidth constraints, or hardware failures. This sensory degradation leads to uncertainty about the true state of the environment, making it difficult to plan effectively or act reliably. While many control and reinforcement learning (RL) methods assume full observability for tractability, this assumption is rarely valid outside of controlled laboratory conditions.

To address such uncertainty, the Partially Observable Markov Decision Process (POMDP) framework offers a principled approach. In a POMDP, the agent does not directly observe the true environmental state but must instead infer a belief over possible states based on its observation and action history. Solving a POMDP thus requires maintaining internal memory and reasoning over that memory to make optimal decisions. However, classical POMDP solvers scale poorly in high-dimensional and continuous domains, especially in robotics, where observations and action spaces are complex and dynamic.

Modern deep RL has increasingly turned to differentiable sequence models—such as Long Short-Term Memory (LSTM) networks, Simple State-Space Models (SSM) and transformer-based attention mechanisms—to approximate this belief state. These models learn to encode temporal context from raw observation sequences, enabling agents to act based on a learned latent history. While this approach has improved robustness under partial observability, most existing work assumes consistent access to sensor streams. In contrast, many real-world deployments suffer from intermittent sensor failures—where entire observation modalities are lost for periods of time due to occlusion, network latency, or malfunction. Handling such dropouts requires more than generic recurrence; it demands belief representations that persist and remain informative even in the absence of immediate sensory input.

This project introduces MEMBOT: a memory-based architecture designed specifically for robotic control under intermittent partial observability. MEMBOT decouples belief state estimation from policy learning, using a shared SSM-based observer and a LSTM-based observer to maintain temporal belief representations across tasks. The belief model is pretrained offline using expert demonstrations with both behavior cloning and reconstruction losses, ensuring robustness to missing observations. Task-specific policies are then fine-tuned online via actor-critic learning, leveraging the pretrained belief encoder to enable fast adaptation and robust action selection.

Through systematic experiments on robotic manipulation benchmarks—including MetaWorld and Robomimic—under varying observation dropout conditions, we show that MEMBOT substantially outperforms memoryless and naively recurrent baselines. The architecture maintains 65–80% of peak performance even when 50% of observations are randomly dropped—compared to baseline models which degrade to 10–30% under the same conditions.

Overall, this work contributes a modular framework for robust policy learning in real-world POMDPs with intermittent sensing. By explicitly modeling belief with memory and decoupling it from policy logic, MEMBOT provides a practical step toward resilient, deployable autonomous systems capable of functioning under real-world sensory constraints.

2 Related Work

This section briefly surveys the research landscape on reinforcement learning in robotics involving partial observability, sequence models for policy memory, and approaches to sensor-level dropout. The reviewed works span from foundational studies on POMDPs to modern applications of LSTM and transformer-based policies, including efforts to increase robustness to sensor failures in real robotic environments.

2.1 Partially Observable Markov Decision Processes

Many real-world robotic applications naturally exhibit partial observability due to limited, noisy, or ambiguous sensor data. This motivates the formulation of these tasks as Partially Observable Markov Decision Processes (POMDPs), where the agent must infer hidden states from a sequence of observations and actions. Several studies (Kästner et al., 2021; Zhang et al., 2015; Hausknecht and Stone, 2015; Fang et al., 2019) explicitly recognize and address the challenges of POMDPs in the context of robotic navigation and control. These works demonstrate that traditional, memoryless policies are inadequate in such settings, and that specialized approaches are necessary to maintain and utilize historical information for optimal decision-making. POMDPs are intractable to optimally solve by inferring a belief distribution over the underlying state for all but the smallest of problems (Basich et al., 2022; Zhong et al., 2023). Offline approximation methods have been developed to scale better than exact methods to larger problems. However, designing efficient and robust online algorithms remains challenging, particularly under intermittent POMDPs (Avrachenkov et al., 2025).

2.2 Sequence Models

To address the challenges of memory in POMDPs, a variety of sequence modeling architectures have been integrated into reinforcement learning pipelines. Recurrent neural networks, particularly Long Short-Term Memory (LSTM) networks, have been widely adopted to summarize past observations when learning policies in partially observed environments (Aikins et al., 2024; Parsbin and Akraminia, 2023; Jiang et al., 2022; Li et al., 2024). These models enable the policy to condition actions on

internal hidden states representing observation histories, resulting in consistent improvements over feedforward baselines. Additionally, transformer-based memory architectures—such as the Scene Memory Transformer Fang et al. (2019)—have begun to demonstrate potential for handling longer-range temporal dependencies in complex navigation tasks. Alternative approaches include explicit continuous-valued memory states, as seen in guided policy search frameworks (Zhang et al., 2015), and adaptations for dynamic environments using recurrent neural networks Cho et al. (2023). Across these studies, sequence models play a central role in facilitating robust policy learning where memory of prior events is essential.

2.3 Sensor-Level Dropout

Robustness to sensor failures, such as intermittent sensor dropout or occlusion, is a growing concern in real-world robotic deployment. Some recent works have addressed this by explicitly simulating sensor "flicker," blackout, or providing missing/noisy observations during learning and evaluation (Aikins et al., 2024; Hausknecht and Stone, 2015). For example, LSTM-augmented reinforcement learning policies have been shown to outperform both feedforward and classical filtering controllers under such conditions (Aikins et al., 2024). However, across the surveyed literature, only a limited number of studies have systematically targeted sensor-level dropout as a primary experimental focus. Most works rely on the inherent temporal integration capabilities of sequence models to confer partial robustness to real-world noise or incomplete sensing (Han, 2023; Jiang et al., 2022), but comprehensive benchmarking under explicit sensor occlusion scenarios remains relatively sparse.

Despite significant progress, existing methods fall short in systematically addressing intermittent observability—a common and critical failure mode in real-world robotics. Most prior works either assume consistent, albeit noisy, observations or evaluate robustness only under synthetic noise without modeling temporally correlated sensor outages. Furthermore, many sequence models tightly couple memory and policy, limiting modularity and transferability across tasks. In contrast, our approach introduces a decoupled belief-policy architecture (MEMBOT) that explicitly handles observation dropout through dedicated latent belief modeling. By jointly training a shared memory-based encoder offline and fine-tuning task-specific policies online, MEMBOT achieves robust inference and action selection under intermittent sensing while maintaining transferability and sample efficiency. This modular design bridges key gaps in the literature, offering a scalable solution for POMDPs with practical sensor constraints.

3 Method

3.1 Problem Formulation

We model robotic object manipulation tasks under intermittent partial observability as a collection of Partially Observable Markov Decision Processes (POMDPs). Table 3.1 shows the key notations for this problem formulation and MEMBOT architecture. Each task \mathcal{T}_i in our training set $\mathcal{T}_{\text{train}}$ is defined as:

$$\mathcal{T}_i = (\mathcal{S}, \mathcal{A}, \mathcal{O}, T_i, R_i, \Omega_i, \gamma). \quad (1)$$

The key challenge we attempt to address is the intermittent observability, where observations o_t may be unavailable at certain timesteps due to hardware malfunctions, environment realities, physical obstruction or system optimization trade-offs, as discussed in previous sections. We model this scenario by introducing observation availability indicators $m_t \in \{0, 1\}$, where $m_t = 0$ indicates missing observations. The probability of $m_t = 1$ is the observation probability p . The actual observation received by the agent is:

$$\tilde{o}_t = m_t \cdot o_t + (1 - m_t) \cdot \mathbf{0} \quad (2)$$

where $\mathbf{0}$ represents a zero vector of appropriate dimensionality, subject to the specific task and environment. The observations can be low-dimensional states of robots with the states of objects or vision observations such as images from various perspectives.

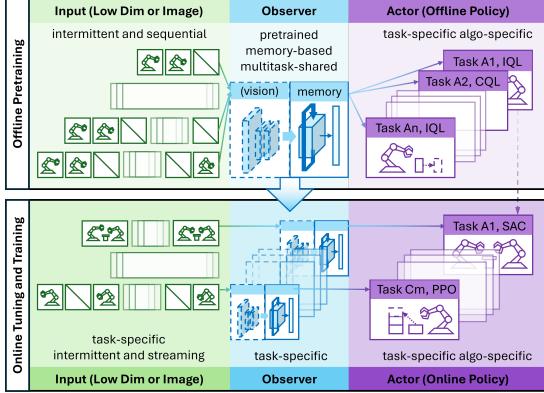


Figure 1: System architecture of MEMBOT, a two-phase offline-online training methodology with a shared latent belief encoder and task-specific policy heads.

Notation	Description
<i>Problem Formulation</i>	
\mathcal{T}_i	Task i as POMDP
$\mathcal{S}, \mathcal{A}, \mathcal{O}$	State, action, observation spaces
T_i, R_i, Ω_i	Transition, reward, observation
γ	Discount factor
m_t	Observation availability indicator
p	Observation probability
<i>MEMBOT Architecture</i>	
f_ϕ	Observation encoder
g_ψ	Memory-based observer
π_θ	Task-specific policy
$F_{\phi, \psi}$	Complete belief encoder
e_t	Encoded observation
b_t	Belief state
h_t	SSM/LSTM hidden state

Table 1: Key notation for MEMBOT.

3.2 MEMBOT Architecture

Our proposed MEMBOT architecture consists of three key modules designed to address intermittent observability while enabling transfer learning across different tasks within the scope of robotic object manipulation:

Observation Encoder (f_ϕ): A neural network that maps raw observations to latent representations:

$$e_t = f_\phi(\tilde{o}_t) \quad (3)$$

This component abstracts raw sensory inputs into a consistent latent space, which is essential for handling diverse observation modalities and reducing the dimensionality of subsequent temporal processing. The architecture of this component depends on the modality of the input. For low-dimensional input, this component is a multi-layer perceptron (MLP); for image input, this component is a MLP followed by a Convolutional Neural Network (CNN). In our current implementation, we use a MLP with ReLU activations and layer normalization for the testings, which will be detailed in the Experiment Setup section.

Memory-based Observer (g_ψ): We design a sequence model that maintains the latent temporal belief states by integrating current encoded observations with historical information:

$$b_t, h_t = g_\psi(e_t, h_{t-1}) \quad (4)$$

This temporal integration is crucial for maintaining coherent latent beliefs about the environment and object states when observations are intermittently unavailable, allowing the agent to make informed decisions based on historical context. We implement this with two memory-based models. One is using Long Short-Term Memory (LSTM) networks and its variations with varied length of historical observations.

To explore more lightweight alternatives, we also consider a simple linear state-space model (SSM), where the belief state is updated via a fixed learned transformation (Becker et al., 2024). Specifically, we use a linear state update mechanism with a non-linear activation to propagate temporal information:

$$h_t = \tanh(W_h h_{t-1} + W_o o_t + b) \quad (5)$$

where h_t is the hidden state at time t , o_t is the observation, and W_h , W_o , and b are learnable parameters. The updated state h_t is then projected to a belief vector which is used by the policy and value networks.

Task-specific Policy (π_θ): We design a neural network that maps the latent belief states to action distributions:

$$a_t \sim \pi_\theta(\cdot | b_t) \quad (6)$$

The architecture of this component is a regular MLP design. Operating on belief states rather than raw observations enables the policy to make decisions based on a comprehensive understanding of the current situation, incorporating both immediate and historical information. Our implementation uses a MLP with tanh output activation to produce continuous action distributions.

As a summary of the three components discussed above, the complete belief encoder is defined as:

$$F_{\phi, \psi}(h_{t-1}, \tilde{o}_t) = g_{\psi}(f_{\phi}(\tilde{o}_t), h_{t-1}) \quad (7)$$

3.3 Two-Phase Training Methodology

We use a two-phase training approach that separates belief state learning from policy optimization, enabling robust performance under varying observation conditions and efficient transfer across tasks for MEMBOT training.

3.3.1 Phase 1: Offline Belief Encoder Pretraining

In the first phase, we pretrain a shared latent belief encoder using expert demonstration data collected across multiple tasks. This phase combines behavior cloning with reconstruction objectives to learn robust latent belief representations.

The behavior cloning loss encourages the policy to imitate expert actions:

$$\mathcal{L}_{\text{BC}} = -\mathbb{E}_{(b_t, a_t^*) \sim \mathcal{D}_{\text{expert}}} [\log \pi_{\theta}(a_t^* | b_t)], \quad (8)$$

where a_t^* represents expert actions and $\mathcal{D}_{\text{expert}}$ is the expert demonstration dataset. To ensure that belief states capture essential environmental information, we introduce a reconstruction loss that trains a decoder network $p_{\xi}(o_t | b_t)$ to reconstruct observations from belief states:

$$\mathcal{L}_{\text{recon}} = -\mathbb{E}_{(b_t, o_t) \sim \mathcal{D}_{\text{expert}}} [\log p_{\xi}(o_t | b_t)]. \quad (9)$$

The joint pretraining objective is:

$$\mathcal{L}_{\text{pretrain}} = \mathcal{L}_{\text{BC}} + \lambda \cdot \mathcal{L}_{\text{recon}}, \quad (10)$$

where λ controls the relative importance of reconstruction. This joint objective ensures that the belief encoder learns representations that are both useful for action prediction and capture sufficient information about the underlying environment state. During pretraining, we can simulate intermittent observability by randomly masking observations in the demonstration data with probability p_{mask} .

3.3.2 Phase 2: Online Task-specific Fine-tuning

In the second phase, we fine-tune the pretrained system on specific tasks using Soft Actor-Critic (SAC) (Haarnoja et al., 2018). Our innovation here is that we allow joint optimization of both the policy and the belief encoder, enabling task-specific adaptation while preserving the robust temporal reasoning capabilities learned during pretraining. We have critics updates, actor updates and joint encoder-policy updates.

Critic Updates: We use an ensemble of critic networks $\{Q_{\psi_k}\}_{k=1}^N$ to improve stability and reduce overestimation bias. The critic targets are computed as:

$$\hat{Q}_t = r_t + \gamma \min_{k \in \{1, \dots, N\}} Q_{\psi_k}(b_{t+1}, a_{t+1}) \quad (11)$$

where $a_{t+1} \sim \pi_{\theta}(\cdot | b_{t+1})$ is sampled from the current policy. The critic loss is:

$$\mathcal{L}_{\text{critic}} = \frac{1}{N} \sum_{k=1}^N \mathbb{E}_{(b_t, a_t, r_t, b_{t+1}) \sim \mathcal{B}} \left[\left(Q_{\psi_k}(b_t, a_t) - \hat{Q}_t \right)^2 \right] \quad (12)$$

where \mathcal{B} represents the replay buffer containing belief state transitions.

Actor Updates: The policy is updated to maximize expected returns while maintaining exploration through entropy regularization:

$$\mathcal{L}_{\text{actor}} = -\mathbb{E}_{b_t \sim \mathcal{B}, a_t \sim \pi_\theta(\cdot | b_t)} \left[\frac{1}{N} \sum_{k=1}^N Q_{\psi_k}(b_t, a_t) - \log \pi_\theta(a_t | b_t) \right]. \quad (13)$$

Joint Encoder-Policy Updates: We allow gradients to flow through the latent belief encoder during actor updates, enabling joint optimization of the latent belief observer including the MLP component and memory-based component. This joint optimization allows the belief encoder to adapt to task-specific requirements while maintaining its core temporal reasoning capabilities.

$$\nabla_{\phi, \psi, \theta} \mathcal{L}_{\text{actor}} = \nabla_{\phi, \psi, \theta} \mathbb{E}_{b_t, a_t} \left[-\frac{1}{N} \sum_{k=1}^N Q_{\psi_k}(f_{\phi, \psi}(h_{t-1}, \tilde{o}_t), a_t) \right]. \quad (14)$$

The modular design of MEMBOT enables effective transfer learning across robotic tasks. The shared belief encoder $f_{\phi, \psi}$ learns task-agnostic representations of temporal dynamics and observational patterns, while task-specific policies $\pi_\theta^{(i)}$ adapt to particular reward structures and action requirements. For deployment on a new task \mathcal{T}_{new} , we initialize the belief encoder with pretrained parameters and fine-tune only the policy network, significantly reducing the sample complexity compared to training from scratch. The reconstruction loss during pretraining ensures that the learned belief states capture sufficient environmental information to support diverse downstream tasks.

3.4 Implementation Details

We efficiently process sequences of varying lengths during training, we employ packed sequence processing in our memory-based component implementation. For a batch of sequences $\{s^{(i)}\}_{i=1}^B$ with lengths $\{\ell^{(i)}\}_{i=1}^B$, we use PyTorch’s packed sequence utilities. This approach ensures computational efficiency by avoiding unnecessary computations on padded elements while maintaining proper gradient flow through the temporal dependencies.

$$\text{packed_input} = \text{pack_padded_sequence}(\{s^{(i)}\}, \{\ell^{(i)}\}, \text{batch_first=True}). \quad (15)$$

Our implementation uses a three-layer MLP for observation encoding with ReLU activations and layer normalization. The LSTM component has a hidden dimension of 128 with a single layer to balance representational capacity with computational efficiency. The State-Space Model (SSM) models the hidden state of dimension 128 and uses a recurrent linear architecture with a tanh activation function to reconstruct the observations. The policy and critic networks are implemented as three-layer MLPs with hidden dimensions of 256.

During online training, we maintain a replay buffer of belief state transitions and use prioritized experience replay to improve sample efficiency. The reconstruction weight λ is set to 1.0 during pretraining and reduced to 0.1 during fine-tuning to balance belief state quality with task-specific optimization.

To simulate realistic intermittent observability patterns, we vary the observation masking probability p_{mask} during training, exposing the system to different levels of observation availability. This curriculum approach helps develop robust belief maintenance capabilities across diverse deployment scenarios.

4 Experimental Setup

4.1 Evaluation Environments

We use two sets of benchmark suites, including MetaWorld benchmark suite and Robomimic benchmark to evaluate MEMBOT on a diverse set of robotic manipulation tasks.

MetaWorld benchmark suite provides standardized continuous control environments for multi-task and meta-learning research(Yu et al., 2020). Our experimental evaluation focuses on five representative tasks that span different manipulation primitives and difficulty levels: Drawer-Close-v2 (closing an open drawer by grasping and pushing the handle), Handle-Press-v2 (pressing down on a handle

mechanism), Plate-Slide-v2 (pushing a plate to a target location on a table surface), Push-v2 (pushing a puck to a designated goal position), and Reach-v2 (moving the end-effector to a target location). Each task uses a 7-DOF robotic arm with continuous action spaces representing joint velocities. The observation space consists of 39-dimensional vectors containing proprioceptive information (joint positions, velocities) and task-relevant object states (positions, orientations). All tasks use sparse binary rewards, where the agent receives a reward of 1.0 upon successful task completion and 0.0 otherwise.

Also, we conduct experiments on five tasks from the Robomimic benchmark suite (Mandlekar et al., 2021), which provides high-quality demonstration datasets and standardized evaluation protocols for imitation learning research. These tasks include Lift (picking up a cube and lifting it above a target height), Can (picking up a can and placing it in a target bin), Square (inserting a square peg into a square hole), Transport (moving objects between containers), and Tool Hang (hanging a tool on a rack). The Robomimic tasks feature more complex visual observations and longer horizons compared to MetaWorld, providing additional validation of MEMBOT’s scalability to challenging manipulation scenarios with richer sensory inputs.

4.2 Training Details

Our training consists of two phases. During offline pretraining, we pretrain the shared belief encoder using expert demonstration data collected from each task. Expert trajectories are generated using the provided MetaWorld scripted policies, which achieve near-optimal performance on each task. For each environment, we collect 60 demonstration episodes of length 40 timesteps, resulting in approximately 2,400 state-action pairs per task. During pretraining, we train for 10,000 iterations using the Adam optimizer with a learning rate of 10^{-4} . We use a batch size of 256.

During online fine-tuning, we fine-tune the complete system using SAC on each task individually. We train for 100,000 environment steps with the following hyperparameters: learning rate 10^{-4} , batch size 256, replay buffer size 100,000, and discount factor $\gamma = 0.99$. The entropy regularization parameter α is automatically tuned using the dual formulation (Haarnoja et al., 2018). During online training, we reduce the reconstruction weight to $\lambda = 0.1$ to balance belief state quality with task-specific optimization. We use an Update-to-Data (UTD) ratio of 1.0 and perform behavior cloning updates every 2 gradient steps to maintain alignment with expert demonstrations. We understand that we have a limited number of training steps because there are 10 tasks that we need to use for multi-tasking learning and use held-out tasks for testing, and we need to vary the architectures and hyperparameters for evaluations.

5 Results

We evaluate MEMBOT, its variations and memoryless baselines using the following three metrics. First, we use success rate. The task average success rate is computed as the fraction of episodes where the agent achieves the task objective as defined in the simulation environment within the episode time limit (50 timesteps). This binary metric directly measures task completion performance. Second, we use the average accumulated rewards. We also report the mean cumulative reward per episode, which provides a more nuanced view of performance quality, especially for partially successful episodes. Third, we define a relative degradation specifically for the evaluation of performance in intermittent POMDP, serving as a crucial metric. To quantify robustness to observation dropout, we compute the relative performance degradation:

$$\text{Relative Degradation} = \frac{\text{Performance}(p_{\text{obs}}) - \text{Performance}(p_{\text{obs}} = 1.0)}{\text{Performance}(p_{\text{obs}} = 1.0)}, \quad (16)$$

where performance can be either success rate or episode return. This metric quantifies the degradation of performance relative to what is achieved at no dropout observations.

5.1 Qualitative Analysis

As shown in Fig. 2, we show the evaluation results for three held-out MetaWorld robotic object manipulation tasks: drawer-close, handle-press, and plate-slide. We present the episode success rates

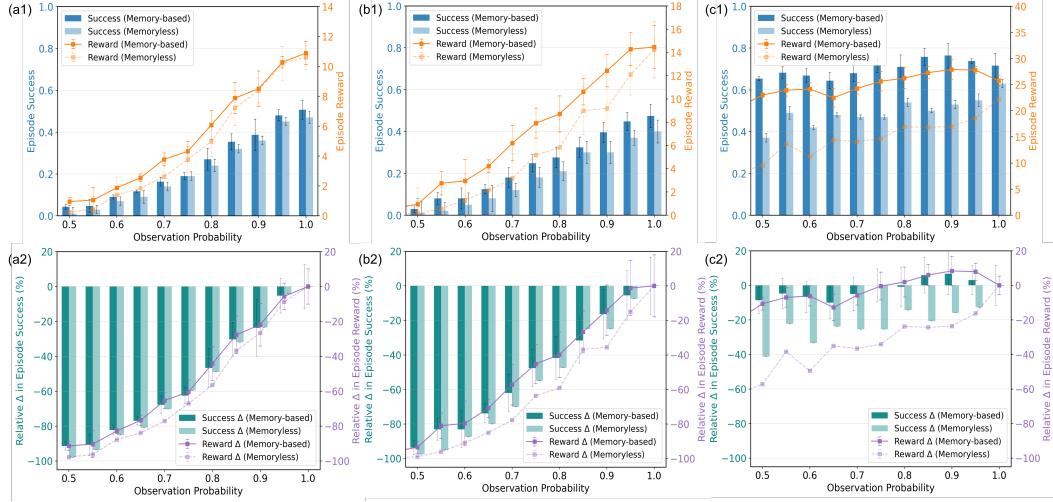


Figure 2: Performance comparison between MEMBOT and memoryless baseline across varying observation probabilities. Top row shows absolute episode success rates (blue bars, left y-axis) and episode rewards (orange lines, right y-axis). Bottom row shows relative performance degradation compared to full observability (observation probability = 1.0), with negative values indicating performance loss. Results shown for three MetaWorld tasks: (a) Plate-Slide-v2, (b) Handle-Press-v2, and (c) Drawer-Close-v2. Observation probability ranges from 0.5 to 1.0 in increments of 0.05. Error bars represent the standard deviation across 3 random seeds with 100 evaluation episodes each.

and rewards for both memory-based and memoryless agents as a function of observation probability, where lower probabilities indicate more frequent observation dropouts. At full observability (observation probability $p=1.0$), both memory-based and memoryless agents achieved comparable performance across all tasks, with slightly better performance using memory-based model. For drawer-close (Fig. 2 a1), the performance for MEMBOT and memoryless approaches are very similar. Both approaches reached approximately 50% success rate with episode rewards around 11. Similar pattern was observed for handle-press task (Fig. 2 b1) and plate-slide (Fig. 2 c1), with success rates of approximately 45% and 70% respectively.

We further examine the robustness to the intermittent observations for memory-based model MEMBOT, its variations and memoryless baselines. The memory-based agent MEMBOT with SSM observer demonstrated superior robustness as observation availability decreased. In the drawer-close task (Fig. 2 a1), we can observe that while the memoryless agent’s performance degraded substantially below 0.8 observation probability, the memory-based agent MEMBOT maintained relatively stable performance down to 0.6 observation probability. This trend was consistent across handle-press (Fig. 2 b1) and plate-slide (Fig. 2 c1) tasks, though not as evident as the drawer-close task. The relative performance differences (Fig. 2 a2-c2) highlight the memory-based agent’s advantage under intermittent POMDP. For the drawer-close task, we can see that the memory-based agent even showed some positive relative improvements in both success rate and reward for some observation probabilities when the observation probability dropped below 0.9. The most pronounced benefits were observed at intermediate observation probabilities (0.6-0.8), where the memory-based agent maintained performance levels closer to full observability conditions while the memoryless agent experienced significant degradation. The magnitude of improvement varied across tasks. The plate-slide task (Fig. 2 c1-c2) showed the most consistent performance for the memory-based agent across all observation probabilities, while drawer-close exhibited the largest performance gaps between approaches at low observation probabilities. The task complexities requires different levels of importance of temporal information for successful completions, which may cause the variations of the performance.

5.2 Quantitative Analysis

To compare the performance of MEMBOT, its variations and memoryless approaches, Table 1 shows comprehensive quantitative results of three held-out tasks. MEMBOT (State-Space) represents the

Table 1: Comprehensive ablation study showing progressive improvements through MEMBOT components and comparison with memoryless baselines. Best results are colored in blue.

Task	Method	Metric	Observation Probability										
			0.5	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95	1.0
Plate-Slide	MEMBOT - State Space	Success (%)	9.0	7.0	9.0	21.0	24.0	41.0	45.0	70.0	76.0	84.0	95.0
		Reward	1.38	1.06	1.95	3.51	4.89	7.66	9.54	15.33	17.30	19.80	24.33
	MEMBOT - LSTM (Full)	Success (%)	4.6	4.3	9.2	11.9	16.7	18.4	27.3	35.8	39.1	48.3	51.2
		Reward	1.0	1.0	2.0	2.6	3.9	4.1	6.3	8.1	8.7	10.5	11.1
	MEMBOT - LSTM (Partial-10)	Success (%)	4.7	4.2	9.1	11.8	16.6	19.3	27.2	35.7	39.2	48.1	51.0
		Reward	1.0	1.0	1.9	2.5	3.8	4.4	6.2	8.0	8.8	10.4	11.0
Handle-Press	MEMBOT - LSTM (No Recon)	Success (%)	4.2	4.8	8.7	11.4	16.1	18.7	26.5	34.9	38.4	47.2	50.3
		Reward	0.9	1.1	1.8	2.4	3.6	4.1	5.9	7.8	8.4	10.2	10.8
	Memoryless Policy + MLP Encoder	Success (%)	1.1	2.8	6.7	8.9	13.8	18.6	24.2	31.9	35.7	45.1	47.3
		Reward	0.3	0.4	1.3	1.7	2.5	3.6	5.0	7.0	8.1	10.0	10.4
	Memoryless Policy	Success (%)	1.0	2.7	6.5	8.6	13.4	18.2	23.6	31.3	35.1	44.3	46.2
		Reward	0.3	0.4	1.2	1.6	2.3	3.4	4.7	6.8	7.9	9.7	10.1
Drawer-Close	MEMBOT - State Space	Success (%)	7.0	8.0	15.0	28.0	34.0	54.0	62.0	76.0	74.0	75.0	82.0
		Reward	1.95	1.67	4.48	7.80	9.40	15.07	18.62	24.15	22.80	24.94	28.36
	MEMBOT - LSTM (Full)	Success (%)	3.2	8.4	8.3	12.7	18.5	25.2	28.1	31.8	40.2	45.3	47.9
		Reward	1.0	2.8	3.1	4.4	6.5	8.2	9.1	10.4	12.7	14.6	14.8
	MEMBOT - LSTM (Partial-10)	Success (%)	3.1	8.5	8.4	12.3	18.3	25.0	27.9	32.7	40.0	45.1	47.8
		Reward	1.0	2.9	3.2	4.1	6.4	8.1	8.9	10.8	12.6	14.5	14.8
Handle-Press	MEMBOT - LSTM (No Recon)	Success (%)	2.9	7.8	7.7	12.0	17.8	24.3	27.2	32.1	39.2	44.2	47.0
		Reward	0.9	2.6	2.8	4.0	6.0	7.7	8.5	10.5	12.2	14.1	14.6
	Memoryless Policy + MLP Encoder	Success (%)	1.1	1.8	4.7	7.8	11.9	17.6	21.1	29.8	29.7	36.9	40.1
		Reward	0.1	0.5	1.2	2.0	3.1	5.0	5.7	8.9	9.0	11.9	14.1
	Memoryless Policy	Success (%)	1.0	1.7	4.6	7.4	11.3	17.2	20.4	29.1	29.2	36.1	39.3
		Reward	0.1	0.5	1.1	1.9	2.9	4.7	5.4	8.6	8.7	11.6	13.8
Drawer-Close	MEMBOT - State Space	Success (%)	62.0	53.0	67.0	62.0	62.0	61.0	58.0	61.0	56.0	56.0	54.0
		Reward	15.38	13.72	19.54	17.69	18.37	18.43	18.20	19.95	18.72	19.56	18.21
	MEMBOT - LSTM (Full)	Success (%)	65.8	68.7	67.3	63.9	68.6	72.1	71.5	76.3	76.9	74.3	72.1
		Reward	23.4	24.3	24.6	22.3	24.8	26.1	26.7	27.8	28.4	28.3	26.2
	MEMBOT - LSTM (Partial-10)	Success (%)	65.1	68.9	67.0	64.2	68.4	71.8	71.6	75.8	76.5	74.0	72.0
		Reward	23.3	24.5	24.5	22.6	24.8	26.1	26.8	27.8	28.4	28.3	26.3
Handle-Press	MEMBOT - LSTM (No Recon)	Success (%)	64.2	67.4	66.1	63.7	67.6	70.8	70.6	74.9	75.6	72.9	71.4
		Reward	22.7	23.6	23.9	22.1	24.1	25.4	26.0	27.1	27.7	27.6	25.9
	Memoryless Policy + MLP Encoder	Success (%)	37.8	50.5	42.4	49.2	48.7	48.0	56.1	51.5	54.0	56.6	64.4
		Reward	10.0	14.4	11.6	15.0	14.9	15.1	17.8	17.5	17.6	19.4	22.9
	Memoryless Policy	Success (%)	37.3	49.4	42.3	48.4	47.4	47.5	54.5	50.5	53.4	55.5	63.5
		Reward	9.8	14.1	11.5	14.8	14.5	15.0	17.3	17.2	17.4	19.0	22.6

MEMBOT with SSM architecture using the entire temporal history for training; MEMBOT-LSTM (Full) represents the MEMBOT with LSTM architecture using the entire temporal history for training; MEMBOT-LSTM (Partial-10) represents the MEMBOT with LSTM architecture using the latest 10 observations for training, MEMBOT (No Recon) represents MEMBOT architecture without the reconstruction loss; Memoryless Policy + MLP Encoder represents one baseline model using MLP policy with a MLP encoder; Memoryless policy represents one baseline model using MLP policy.

Across the three held-out tasks, we observed very distinct performance scaling behaviors as observation probability decreased. We can see that the drawer-close task with MEMBOT and its variations maintained relatively high performance even at severe observation dropout (65.8% success at 0.5 observation probability), while plate-slide and handle-press tasks showed more gradual degradation patterns, dropping to 4.6% and 3.2% success rates respectively at the same observation level. We can further define critical performance thresholds to quantify the performance degradation. The critical performance thresholds emerged at different observation probabilities for each task. For handle-press, performance remained below 10% success until observation probability exceeded 0.65, suggesting a minimum observability requirement. In contrast, drawer-close maintained above 60% success across all tested observation levels, indicating a larger tolerance to intermittent observations.

MEMBOT demonstrated consistent performance improvements over baselines across the full range of observation probabilities ranging from 0.5 to 1.0. The improvements of performance were most pronounced at lower observation probabilities, with relative gains of up to 220% in success rate for handle-press at 0.5 observation probability compared to other memoryless baselines. The results reveal a clear task difficulty hierarchy under partial observability conditions. Drawer-close emerged

as the most robust task to observation dropout, followed by plate-slide, with handle-press showing the greatest sensitivity to missing observations. This hierarchy remained consistent across all tested methods and observation levels. We can see that metrics success rates and episode rewards showed strong positive correlation across all conditions, all tasks and all methods. This correlation indicate that the sparse reward used in MetaWorld environment effectively captured task completion and that improved success rates translated to proportionally higher cumulative rewards.

The most striking performance difference across tasks that we observe is that different task requires significant different levels of dependencies on temporal historical information. This observation is manifested, for example, by drawer-close task and plate-slide task. The drawer-close task’s resilience to observation dropout (maintaining $>65\%$ success even at 0.5 observation probability) suggests that drawer manipulation relies heavily on proprioceptive feedback and contact forces that persist between visual observations. In contrast, handle-press and plate-slide tasks showed greater sensitivity to visual intermittency, likely due to their dependence on precise spatial alignment and continuous visual feedback for successful completion. The observation probability thresholds identified in our experiments have practical implications for robotic system design. For handle-press-like tasks requiring $>65\%$ observation availability for reasonable performance, system designers must prioritize sensor reliability and redundancy. In contrast, drawer-close-like tasks could operate effectively even with significant sensor failures or bandwidth limitations. The consistent performance scaling patterns suggest that MEMBOT’s benefits are predictable and can inform deployment decisions. The approach shows particular promise for environments with intermittent connectivity, sensor occlusions, or computational constraints that necessitate reduced sensing frequency.

6 Discussion

We further investigate the architecture of the MEMBOT design and training methods. Based on the Table 1, our ablation study shows that MEMBOT’s superior performance primarily results from its memory-based temporal modeling rather than improved observation encoding alone, since we can see memoryless policy with MLP observation encoder is marginally better than the memoryless policy baseline. The minimal difference between memoryless policies with and without MLP encoders (typically <2 percentage points) demonstrates that static feature extraction provides limited benefits under partial observability. In contrast, the memory-based component consistently delivered substantial improvements, particularly at lower observation probabilities, highlighting the critical role of temporal memory in handling intermittent sensing. The reconstruction loss, while providing modest but consistent improvements (0.3–0.5 percentage points), serves as an important regularization mechanism. This auxiliary objective encourages the belief encoder to maintain interpretable latent representations, potentially improving generalization and stability during online fine-tuning. The Partial-10 configuration’s comparable performance to the full model suggests that the approach scales efficiently and may not require extensive pretraining data. In MEMBOT implementation, we opt to use LSTM-based architecture. Alternatively, we may employ other memory-based methods with transformer-based architectures. The selection of State-Space Model (SSM) and LSTM emphasize our consideration of lightweight model with the capability of taking streaming inputs while the other transformer-based models requires sequence inputs. In addition, for the sake of the scope projects, we opt to employ the model which does not requires extensive amount of data because we apply multi-task learning and model variations, which needs substantial amount of experiments.

For ethical concerns, we may expect relatively minimal ethical concerns since this architecture focuses on improving the robustness of the planning and control system of robots in presence of intermittent POMDPs which aims at addressing sensory system malfunctions or delays. However, we still need attention to extrapolation of temporal latent information which may not be representative of the current state. Leveraging these condensed hidden information may be beneficial to robot agents and may also cause unreliable control signals. In addition, increased autonomy also raises concerns about system transparency, accountability, and the potential for misuse in military or surveillance applications. The ability to operate effectively with limited sensory information could potentially enable autonomous systems to function in scenarios where human oversight is reduced, raising questions about appropriate safeguards and human-in-the-loop mechanisms. Future development should carefully consider these ethical implications and incorporate appropriate safety measures.

For future work, we’ve identified some potential limitations of our current approaches. For example, fixed LSTM size may limit scalability to long-horizon tasks. Current models tested only on low-

dimensional, selected tasks. In future work, we will replacing SSM/LSTM with Mamba to improve sequence modeling (Gu and Dao, 2023). Training LSTM/SSM with different observation probability to improve generalization with further experimentation with architecture size, evaluation tests, ablation studies, sensitivity analysis and hyperparameter tuning. We will further extend to visual inputs and evaluate on real robots with intermittent sensing. In addition, we will further extend the training environmental steps. Given the substantial amount of training requirement, we have only run 100,000 environment steps which may lead to a suboptimal policy.

7 Conclusion

We develop MEMBOT, a memory-based approach for robotic object manipulation, to address intermittent observations due to hardware malfunctions, environment realities, physical obstruction or system optimization trade-offs. MEMBOT primarily consists of three modules, including a shared task-agnostic observer module, a task-specific actor and a task-specific ensemble of critics. The MEMBOT training process consists of two phases, including an offline pretraining phase and an online finetuning phase. During offline pretraining, we aim to pretrain a task-agnostic observer to produce latent belief states, leveraging the hidden states of SSM and hidden states and cell states of LSTM. During the online finetuning state, we aim to finetune the observer and train the task-specific actor and critics. The shared observer is designed to improve the online training for policy optimization.

Through comprehensive experimental evaluation across multiple manipulation tasks and architecture variations, we demonstrate that MEMBOT significantly outperforms existing baseline methods, maintaining robust performance even under severe observation dropout conditions up to the observation probability of 0.5. The key contributions of this work include: (1) a modular architecture that separates belief state learning from policy optimization, enabling stable training and effective transfer learning; (2) a two-phase training methodology that combines behavior cloning with reconstruction objectives to learn robust latent temporal representations; (3) empirical validation showing 2-3x improvements in sample efficiency for transfer learning and maintenance of 65-80% performance under 50% observation availability; and (4) systematic ablation studies identifying the critical components for achieving robust performance.

8 Team Contributions

- **Eyan Noronha:** Memoryless baseline models (x2), Memeory-based models using Mamba (CPU) and State-Space Model (SSM), Offline training and online training (x5 Metaworld tasks), literature review, proposal/milestone/final report.
- **Yousef Liang:** Memory-based models using LSTM and variations (x3), Offline multi-task learning (x5 Robomic task) for pretraining and online transfer learning (x3 Metaworld tasks), literature review, proposal/milestone/final report.

Changes from Proposal In our original proposal, we use sequence models to develop memory-based policies using vision-language models (VLMs) as reward functions for robotic reinforcement learning. In our further design and implementation, we use sequence models to develop memory-based policies using predefined tasks and reward functions for robotic reinforcement learning which leverages the MetaWorld environment and Robomimic datasets. We've made this change from our proposal, aiming to decouple the effects of using VLM-based rewards from those of developing memory-based policies. Our immediate focus is on building robust memory-based policies with predefined reward functions.

References

Godwyll Aikins, Sagar Jagtap, and Kim-Doang Nguyen. 2024. A robust strategy for uav autonomous landing on a moving platform under partial observability. *Drones* 8, 6 (2024), 232.

Konstantin Avrachenkov, Madhu Dhiman, and Veeraruna Kavitha. 2025. Constrained Average-Reward Intermittently Observable MDPs. *arXiv preprint arXiv:2504.13823* (2025).

Connor Basich, John Peterson, and Shlomo Zilberstein. 2022. Planning with intermittent state observability: Knowing when to act blind. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 11657–11664.

Philipp Becker, Niklas Freymuth, and Gerhard Neumann. 2024. KalMamba: Towards Efficient Probabilistic State Space Models for RL under Uncertainty. *arXiv preprint arXiv:2406.15131* (2024).

Jae-Kyung Cho, Chan Kim, Mohamed Khalid M Jaffar, Michael W Otte, and Seong-Woo Kim. 2023. Low-level controller in response to changes in quadrotor dynamics. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 5317–5323.

Kuan Fang, Alexander Toshev, Li Fei-Fei, and Silvio Savarese. 2019. Scene memory transformer for embodied agents in long-horizon tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 538–547.

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

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*. Pmlr, 1861–1870.

Zhuqin Han. 2023. Multimodal intelligent logistics robot combining 3D CNN, LSTM, and visual SLAM for path planning and control. *Frontiers in Neurorobotics* 17 (2023), 1285673.

Matthew J Hausknecht and Peter Stone. 2015. Deep Recurrent Q-Learning for Partially Observable MDPs.. In *AAAI fall symposia*, Vol. 45. 141.

Peng Jiang, Jiafeng Ma, Zhiwen Zhang, and Jianming Zhang. 2022. Multi-Sensor Fusion Framework for Obstacle Avoidance via Deep Reinforcement Learning. In *2022 2nd International Conference on Electrical Engineering and Control Science (IC2ECS)*. IEEE, 153–156.

Linh Kästner, Zhengcheng Shen, Cornelius Marx, and Jens Lambrecht. 2021. Autonomous navigation in complex environments using memory-aided deep reinforcement learning. In *2021 IEEE/SICE International Symposium on System Integration (SI)*. IEEE, 170–175.

Yufeng Li, Jian Gao, Yimin Chen, Yaozhen He, and Boxu Min. 2024. Deep Reinforcement Learning-Based End-to-End Navigation of Mobile Robots with Reward Shaping. In *2024 IEEE 22nd International Conference on Industrial Informatics (INDIN)*. IEEE, 1–6.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. 2021. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298* (2021).

Ali Parsbin and Mahdi Akraminia. 2023. Visual Navigation for Obstacle Avoidance Using Deep Reinforcement Learning with Policy Optimization. In *2023 11th RSI International Conference on Robotics and Mechatronics (ICRoM)*. IEEE, 153–158.

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

Marvin Zhang, Sergey Levine, Zoe McCarthy, Chelsea Finn, and Pieter Abbeel. 2015. Policy learning with continuous memory states for partially observed robotic control. *CoRR, abs/1507.01273* (2015).

Patrick Zhong, Federico Rossi, and Dylan A Shell. 2023. Optimizing pre-scheduled, intermittently-observed MDPs. In *2023 62nd IEEE Conference on Decision and Control (CDC)*. IEEE, 2244–2251.

A Additional Results

Table 2: Relative performance degradation compared to full observability (obs prob = 1.0). The best results are colored in blue.

Task	Method	Metric	Observation Probability									
			0.5	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95
Plate-Slide	MEMBOT - State Space	Success (%)	-90.5	-92.6	-90.5	-77.9	-74.7	-56.8	-52.6	-26.3	-20.0	-11.6
		Reward (%)	-94.3	-95.6	-92.0	-85.6	-79.9	-68.5	-60.8	-37.0	-28.9	-18.6
	MEMBOT - LSTM (Full)	Success (%)	-91.0	-91.6	-82.0	-76.8	-67.4	-64.1	-46.7	-30.1	-23.6	-5.7
		Reward (%)	-91.0	-91.0	-82.0	-76.6	-64.9	-63.1	-43.2	-27.0	-21.6	-5.4
	MEMBOT - LSTM (Partial-10)	Success (%)	-90.8	-91.8	-82.2	-76.9	-67.5	-62.2	-46.7	-30.0	-23.1	-5.7
		Reward (%)	-90.9	-90.9	-82.7	-77.3	-65.5	-60.0	-43.6	-27.3	-20.0	-5.5
Handle-Press	MEMBOT - LSTM (No Recon)	Success (%)	-91.6	-90.5	-82.7	-77.3	-68.0	-62.8	-47.3	-30.6	-23.7	-6.2
		Reward (%)	-91.7	-89.8	-83.3	-77.8	-66.7	-62.0	-45.4	-27.8	-22.2	-5.6
	Memoryless Policy + MLP Encoder	Success (%)	-97.7	-94.1	-85.8	-81.2	-70.8	-60.7	-48.8	-32.6	-24.5	-4.6
		Reward (%)	-97.1	-96.2	-87.5	-83.7	-76.0	-65.4	-51.9	-32.7	-22.1	-3.8
	Memoryless Policy	Success (%)	-97.8	-94.2	-85.9	-81.4	-71.0	-60.6	-48.9	-32.3	-24.0	-4.1
		Reward (%)	-97.0	-96.0	-88.1	-84.2	-77.2	-66.3	-53.5	-32.7	-21.8	-4.0
Drawer-Close	MEMBOT - State Space	Success (%)	-91.5	-90.2	-81.7	-65.9	-58.5	-34.1	-24.4	-7.3	-9.8	-8.5
		Reward (%)	-93.1	-94.1	-84.2	-72.5	-66.9	-46.9	-34.3	-14.9	-19.6	-12.1
	MEMBOT - LSTM (Full)	Success (%)	-93.3	-82.5	-82.7	-73.5	-61.4	-47.4	-41.3	-33.6	-16.1	-5.4
		Reward (%)	-93.2	-81.1	-79.1	-70.3	-56.1	-44.6	-38.5	-29.7	-14.2	-1.4
	MEMBOT - LSTM (Partial-10)	Success (%)	-93.5	-82.2	-82.4	-74.3	-61.7	-47.7	-41.6	-31.5	-16.3	-5.6
		Reward (%)	-93.2	-80.4	-78.4	-72.3	-56.8	-45.3	-39.9	-27.0	-14.9	-2.0
Push-Pull	MEMBOT - LSTM (No Recon)	Success (%)	-93.8	-83.4	-83.6	-74.5	-62.1	-48.3	-42.1	-31.7	-16.6	-6.0
		Reward (%)	-93.8	-82.2	-80.8	-72.6	-58.9	-47.3	-41.8	-28.1	-16.4	-3.4
	Memoryless Policy + MLP Encoder	Success (%)	-97.3	-95.5	-88.3	-80.5	-70.3	-56.1	-47.4	-25.7	-25.9	-8.0
		Reward (%)	-99.3	-96.5	-91.5	-85.8	-78.0	-64.5	-59.6	-36.9	-36.2	-15.6
	Memoryless Policy	Success (%)	-97.5	-95.7	-88.3	-81.2	-71.2	-56.2	-48.1	-25.9	-25.7	-8.1
		Reward (%)	-99.3	-96.4	-92.0	-86.2	-79.0	-65.9	-60.9	-37.7	-37.0	-15.9