ACTION-INFORMED BELIEF VIA LATENT FLOW MATCHING FOR MEMORY-INTENSIVE ROBOTIC TASKS

Dawei Wang[1] **Xingrui Yu**[2,†] **Zhenglin Wan**[3] **Ivor Tsang**[2,4]
[1]Leiden University, The Netherlands
[2]CFAR and IHPC, Agency for Science, Technology and Research (A*STAR), Singapore
[3]National University of Singapore [4]CCDS, Nanyang Technological University, Singapore
d.wang.17@umail.leidenuniv.nl, yu_xingrui@a-star.edu.sg

ABSTRACT

Memory is essential for achieving fully autonomous robots in real-world settings, yet reliably remembering fine-grained information in complex and dynamic environments for robots remains an unsolved challenge. While RNN-based approaches combined with reinforcement learning can handle simple memory problems, they often struggle with memory-intensive robotics tasks where success hinges on subtle, fine-grained distinctions. Belief-based methods that leverage privileged information offer a solution for partially observable Markov decision processes (POMDPs). However, the current method focuses solely on reconstructing the belief representation, while ignoring the strong coupling between state variations and actions in the representation space. As a result, the structure of its latent representation is not well-suited to handling the high-dimensional observations and complex dynamics encountered in robotic environments. Motivated by this gap, we propose an action-informed belief reconstruction method. Specifically, we employ Flow Matching to model inverse dynamics as a structural constraint, explicitly steering the latent belief distribution to align with action requirements. By formulating this auxiliary objective within the latent space, our approach enforces strong causal coupling that better preserves fine-grained information and achieves strong performance on the memory-intensive robotic manipulation benchmarks.

1 INTRODUCTION

In recent years, memory has attracted increasing attention and is widely considered a key component for building general-purpose intelligent agents (Hu et al., 2025; Chhikara et al., 2025). Despite this progress, memory-intensive tasks for robotics in the physical world remain relatively underexplored. For embodied agents, memory is not merely an auxiliary capability: it is often a prerequisite for autonomy. For example, a fully automated cooking robot must retain and recall a large amount of procedural knowledge (e.g., recipes and intermediate steps) in order to reliably complete real-world cooking tasks.

However, acquiring the memory capability for robotics described above is highly challenging. Taking robotic manipulation as an example, a robot must operate in complex, dynamic environments, where it needs to perceive, understand, and remember task-relevant objects—either static objects or objects moving through the physical world at varying speeds—while executing appropriate actions. This is undoubtedly a highly challenging problem.

Recently, there has indeed been growing interest in memory-dependent tasks for robotics, but most existing efforts build memory management on Transformer-based (Cherepanov et al., 2024) or vision–language–action (VLA) architectures (Kim et al., 2024; Sridhar et al., 2025). In contrast, reinforcement learning (RL), as a classical paradigm in robot learning, features a lightweight and concise architecture and still achieves leading performance on many robotic tasks, remaining one of the mainstream approaches in robotics. However, for memory-related manipulation, the dominant

†Corresponding author.

RL solutions largely remain limited to augmenting policies with recurrent neural networks (RNNs), which has been shown to be ineffective on recent, more complex memory-intensive robotics manipulation benchmarks (Cherepanov et al., 2025a).

An alternative direction for reinforcement learning in memory tasks is to learn belief representations with privileged information, a paradigm where additional state information is available during training but not at execution time. Prior work (Pinto et al., 2018; Baisero & Amato, 2021) focus on partial observability that arises naturally from robotic cameras, and thus do not consider extra "unobservable information" such as memory. Wang et al. (2023) explores environments containing more "unobservable information" and achieves strong performance on POMDP problems with unobservable information. However, this method is evaluated primarily in grid-world settings that require a relatively simple dynamic model, and it overlooks the strong causal coupling between belief representations and actions in the latent space, which makes the dynamic model more complex in the memory-intensive robotic manipulation tasks.

Building on this observation, we consider introducing auxiliary tasks and representation learning to improve the structure of belief representations. Inverse dynamics is widely regarded as an effective representation learning objective for action-relevant multi-task learning (Brandfonbrener et al., 2023), and incorporating it into belief learning has the potential to strengthen the alignment between reconstructed belief representations and the action structure required for control. However, existing VAE-based belief methods (Wang et al., 2023; Kingma & Welling, 2014) typically adopt a static, history-independent prior, which prevents the model from retaining—during inference—the structured information injected into the latent space by auxiliary objectives (e.g., inverse dynamics). As a result, these frameworks cannot readily support auxiliary multi-task learning. Moreover, belief modeling relies on data collected under stochastic actions to enrich representation diversity; however, we find that conventional supervised inverse dynamics is difficult to train effectively when predicting such random actions from the latent space.

To address these limitations, we propose a new privileged-information method, namely the action-informed belief (AIB) modeling approach. We design a belief modeling framework with more consistent training and inference, so that the structured information learned during training can be better preserved and utilized at test time. Building on this framework, we further introduce an inverse dynamics objective with flow matching (Liu et al., 2022) in the latent space, which can alleviate the representational limitations of conventional inverse dynamics when operating in latent space. The key idea is to jointly shape the latent space with two coupled objectives: while learning to reconstruct belief representations from latent variables, we simultaneously solve a latent-space inverse dynamics task with flow matching. The action-prediction gradients are backpropagated into the latent variables, reshaping the latent distribution to better reflect the structure required for action control and thereby enabling more faithful belief reconstruction. On memory-intensive robotic manipulation benchmarks, our method outperforms strong privileged-belief baselines.

(1) We are the first to incorporate privileged information belief modeling into reinforcement learning for complex memory-intensive robotic tasks, substantially improving performance on memory-intensive robotic manipulation benchmarks.

(2) We propose an action-informed belief learning method that leverages Flow Matching to impose dynamic constraints on the latent space. Our method further improves performance over strong baselines on memory-intensive robotic manipulation benchmarks.

2 Problem Setting: Robot Memory Tasks as POMDPs

Robot memory tasks are control problems in which successful decision-making depends on task-relevant information that was observed earlier in the episode but is not fully available in the current observation. In our setting, each task contains an episode-specific variable that must be remembered for success, such as the identity or attributes of a previously presented target. Therefore, acting solely on the current observation can be insufficient.

The robot memory task is a class of POMDPs in which the partial observability arises from task-relevant information that must be retained over time. Following the standard formulation of partially observable Markov decision processes (POMDPs), we model a robot memory task as

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, T, R, \Omega, \gamma).$$

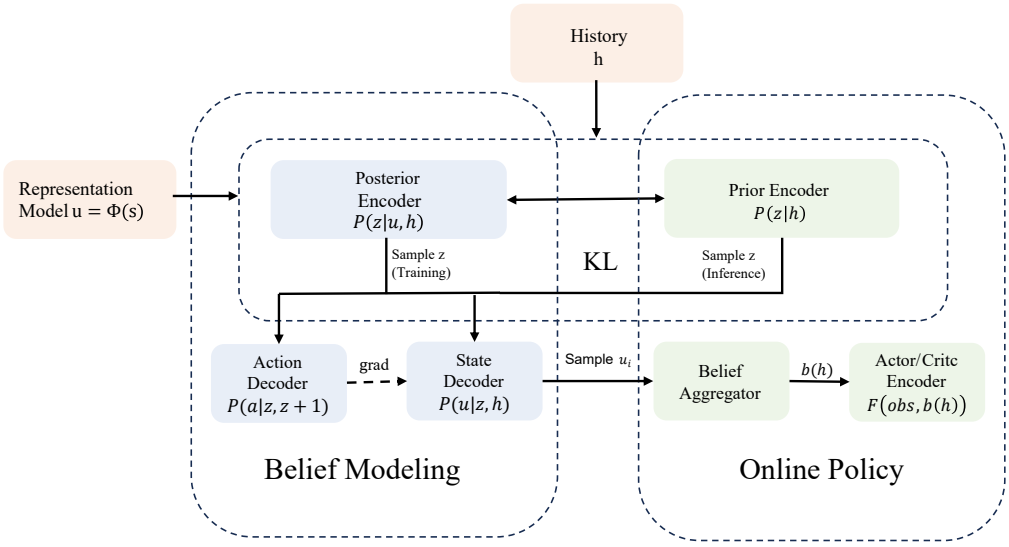


Figure 1: Overview of AIB. A compact privileged representation $u = \phi(s)$ is first learned, which is presented in Algorithm 1. During belief modeling, the posterior $q(z \mid u, h)$ is trained with a flow-matching inverse-dynamics objective and distilled into the history-conditioned prior $p(z \mid h)$ through KL matching. At deployment, prior samples are decoded into particles $\hat{u}_i$, aggregated into the belief $b(h)$, and combined with the current observation for PPO.

where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $\mathcal{O}$ the observation space, T the transition kernel, R the reward function, Ω the observation model, and γ the discount factor. Additional definitions and background on POMDPs and reinforcement learning are provided in Appendix E. At time step t, the environment is in state $s_t \in \mathcal{S}$, the robot executes action $a_t \in \mathcal{A}$, and receives observation $o_t \in \mathcal{O}$. We decompose the state as

$$s_t = (\tilde{s}_t, m_t),$$

where $\tilde{s}_t$ denotes the ordinary environment state and m_t denotes the task-relevant memory variable. The environment evolves according to

$$s_{t+1} \sim T(\cdot \mid s_t, a_t),$$

and the agent receives observations through

$$o_t \sim \Omega(\cdot \mid s_t).$$

Belief is also a common component in POMDPs. The belief state is the posterior distribution

$$b_t(s) = \Pr(s_t = s \mid h_t),$$

where h_t denotes the interaction history. In our work, we consider a compact latent encoding $u_t = \phi(s_t)$ and approximate the belief as

$$b(h_t) \approx p(u_t \mid h_t).$$

3 METHOD

Inspired by prior work (Wang et al., 2023), we propose Action-Informed Belief (AIB), a privileged-information-based belief modeling method that explicitly couples belief learning with action-informed dynamics. Specifically, AIB employs Flow Matching (Liu et al., 2022) to model inverse dynamics as a soft constraint in the latent space. By backpropagating these gradients, we explicitly steer the latent representation to align with the action dynamics. This enforces a strong structural coupling, making the belief state more robust for complex, high-dimensional robotic environments.

Our approach consists of three stages: representation learning, belief modeling and online reinforcement learning. Our final objective is to learn a policy of the form

$$\pi(a_t \mid o_t, b(h_t)),$$

Algorithm 1 Learning compact state representations.

Input: Dataset $\mathcal{D} = \{(s_t, o_t, a_t, r_t, s_{t+1}, o_{t+1}) : 1 \le t \le N\}$.
repeat
  Sample transition $(s_t, o_t, a_t, r_t, s_{t+1}, o_{t+1}) \sim \mathcal{D}$.
  Sample latents $u_t^s \sim \phi(\cdot \mid s_t),\ u_t^o \sim \psi(\cdot \mid o_t)$.
  Predict $\hat{r}_t, \hat{u}_{t+1}^s, \hat{u}_{t+1}^o = g(u_t^s, u_t^o, a_t)$.
  Compute targets $u_{t+1}^s \sim \phi(\cdot \mid s_{t+1}),\ u_{t+1}^o \sim \psi(\cdot \mid o_{t+1})$.
  $\mathcal{L}_r = (r_t - \hat{r}_t)^2$.
  $\mathcal{L}_s = \left\|\mathrm{stopgrad}(u_{t+1}^s) - \hat{u}_{t+1}^s\right\|^2$.
  $\mathcal{L}_o = \left\|\mathrm{stopgrad}(u_{t+1}^o) - \hat{u}_{t+1}^o\right\|^2$.
  $\mathcal{L}_{\mathrm{KL}} = \mathrm{KL}[\phi(\cdot \mid s_t) \,\|\, \mathcal{N}(0, I)]$.
  Update ϕ, ψ, g on $\lambda_r \mathcal{L}_r + \lambda_s \mathcal{L}_s + \lambda_o \mathcal{L}_o + \lambda_{\mathrm{KL}} \mathcal{L}_{\mathrm{KL}}$.
until convergence
Output: State encoder ϕ.

where the belief representation $b(h_t)$ is obtained by encoding the interaction history h_t with a Gated Recurrent Unit (GRU) (Chung et al., 2014). The resulting belief is then concatenated with the current observation o_t and fed into the policy network to produce actions and execute action in the environment.

As illustrated in Figure 1, AIB consists of three stages. First, it train a representation model as the input in the next stage. Next, it infers a belief representation. Finally, this belief representation is used to optimize the policy online with PPO for robot memory tasks.

3.1 Representation Learning

In the first stage, we collect an offline dataset with a random policy to train the representation model. We acknowledge that this stage is substantially based on (Wang et al., 2023). The goal of this stage is to learn a compact state encoder ϕ that preserves task-relevant information. Moreover, to better scale to real-world robotic settings with high-dimensional and complex sensory inputs, we aim to reduce redundancy between privileged states and visual observations. Concretely, we encourage information that is directly observable to be captured by the observation stream, while the state representation stores only unobservable yet task-relevant factors.

To achieve the first goal, we adopt a bisimulation (Zhang et al., 2020; Gelada et al., 2019) method: by predicting rewards and latent dynamics conditioned on actions, the learned state representation is encouraged to retain information that is important for decision making. Specifically, given a transition $(s_t, o_t, a_t, r_t, s_{t+1}, o_{t+1})$, we encode the privileged state and observation as $u_t^s = \phi(s_t)$ and $u_t^o = \psi(o_t)$, and train a predictor g to estimate the reward and next-step latents:

$$(\hat{r}_t, \hat{u}_{t+1}^s, \hat{u}_{t+1}^o) = g(u_t^s, u_t^o, a_t),$$

where these terms $(r_t, \hat{u}_{t+1}^s, \hat{u}_{t+1}^o)$ is the predicted value. To achieve the second goal (reducing overlap between state and observation information), we jointly train the state encoder ϕ and observation encoder ψ with an variational information bottleneck (Alemi et al., 2016) on the privileged state representation. The specific way is treating ϕ, ψ as stochastic encoders, encouraging to retain only task-relevant information beyond what is already available from observations. In practice, We regularize the state encoder via a KL penalty,

$$\mathcal{L}_{\mathrm{KL}} = \mathrm{KL}[\phi(\cdot \mid s_t) \,\|\, \mathcal{N}(0, I)]\,,$$

and optimize the overall representation learning objective

$$\mathcal{L}_{\text{repr}} = \lambda_r \mathcal{L}_r + \lambda_s \mathcal{L}_s + \lambda_o \mathcal{L}_o + \lambda_{\mathrm{KL}} \mathcal{L}_{\mathrm{KL}}.$$

This joint training encourages ϕ and ψ to compete for explanatory capacity, thereby reducing redundant information and yielding a more compact privileged representation for downstream belief learning. The specific training algorithm is in Algorithm 1.

Although we introduce multiple auxiliary tasks at this stage, ultimately we retain and use only the state encoder ϕ.

3.2 ACTION-INFORMED BELIEF MODELING WITH LEARNED PRIORS

Given the compact state encoder learned in stage 1, we denote the compact representation as $u_t = \phi(s_t)$ for simplicity. In stage 2, our goal is to learn a belief model that estimates the conditional distribution $p(u_t \mid h_t)$ from the GRU-compressed history h_t. As shown in Figure 1, our method and workflow can be clearly illustrated.

VAE for belief modeling We consider using a VAE-based architecture (Kingma & Welling, 2014), which has been shown to model complex distributions and to handle belief modeling in POMDPs (Wang et al., 2023). From a probabilistic perspective, we aim to maximize the conditional log-likelihood over the dataset $\mathcal{D}$:

$$\max \ \mathbb{E}_{(u_t,h_t)\sim\mathcal{D}}\big[\log p(u_t \mid h_t)\big].$$

To effectively capture the inherent uncertainty and multimodal distribution of the state u_t in this objective, we follow the formulation in Believer (Wang et al., 2023), which introduces a stochastic latent variable z_t. In these prior approaches, z_t is typically sampled from a fixed, history-independent standard Gaussian prior $p(z_t) = \mathcal{N}(0, I)$, leading to the joint factorization:

$$p(u_t, h_t, z_t) = p(h_t)\, p(z_t)\, p_\theta(u_t \mid h_t, z_t).$$

However, relying on a fixed standard Gaussian prior implies that during inference, the latent variable z_t acts as a generic random variable, decoupled from the dynamic context of the history h_t. This static assumption creates a fundamental limitation: such an unstructured latent space is incompatible with our proposed flow-matching auxiliary task, which requires z_t to encode specific action-dependent structures to predict inverse dynamics. Consequently, any causal coupling learned during training is effectively lost at deployment time.

Generative model with a learned prior To bridge the gap between training-time constraints and inference-time sampling, we propose to replace the static Gaussian distribution $p(z_t)$ with a learnable, history-dependent prior $p_\eta(z_t \mid h_t)$. Unlike the fixed prior which models z_t as a generic random variable, our learned prior predicts a structured latent distribution conditioned directly on the interaction history:

$$p(u_t, z_t \mid h_t) = p_\theta(u_t \mid h_t, z_t)\, p_\eta(z_t \mid h_t).$$

Here, p_η acts as a dynamic generator optimized to match the geometry of the latent space. However, since exact inference is intractable, we resort to variational inference and introduce an inference network $q_\varphi(z_t \mid u_t, h_t)$ to approximate the true posterior. Crucially, this q_φ serves as a teacher during training: it has access to privileged information u_t and is directly steered by our auxiliary structural constraints. By minimizing the divergence between q_φ and p_η, we effectively distill this action-aligned structure into the prior, ensuring that the causal dependencies learned during training are preserved during deployment. The resulting ELBO objective is

$$\mathcal{L}_{\text{VAE}} = \mathbb{E}_{q_\varphi(z_t|u_t,h_t)}\big[\log p_\theta(u_t \mid h_t, z_t)\big] - D_{\text{KL}}\Big(q_\varphi(z_t \mid u_t, h_t) \,\Big\|\, p_\eta(z_t \mid h_t)\Big).$$

Importantly, as we talked before, the KL term above serves not merely as a regularizer but as a knowledge distillation channel: it forces the learned prior $p_\eta(z_t \mid h_t)$ to match the posterior $q_\varphi(z_t \mid u_t, h_t)$, which contains privileged state information and (as described next) auxiliary action-informed semantics. Consequently, at deployment time when u_t is unavailable and the agent only conditions on h_t, samples $z_t \sim p_\eta(z_t \mid h_t)$ can still retain structured information learned during training.

Action-informed constraints Building on the learned-prior VAE, we obtain a history-conditioned latent space where the posterior $q_\varphi(z_t \mid u_t, h_t)$ is available during training and the learned prior $p_\eta(z_t \mid h_t)$ is used at deployment. We leverage this latent space to inject action-informed semantics via an inverse-dynamics objective, i.e., modeling $p(a_t \mid z_t, z_{t+1})$ so that z_t captures information predictive of actions.

A practical challenge is that our belief model is trained from random-policy offline data, where inverse-dynamics supervision can be noisy and prone to regression-to-the-mean, resulting in weak or unstable gradients for shaping the latent geometry. To obtain a robust learning signal under such

stochasticity, we cast inverse dynamics as a conditional generative modeling problem and introduce a flow-matching auxiliary loss to model $p(a_t \mid z_t, z_{t+1})$.

Specifically, we adopt a rectified-flow (Liu et al., 2022) parameterization over actions conditioned on (z_t, z_{t+1}). Let $a_0 \sim \mathcal{N}(0, I)$ be Gaussian noise, $a_1 = a_t$ be the data action, and $\tau \sim \mathcal{U}(0, 1)$. We define the straight-line interpolation

$$a_\tau = (1 - \tau)a_0 + \tau a_1,$$

whose target velocity field is the constant vector $(a_1 - a_0)$. We then train a conditional vector-field network $v_\xi(a_\tau, \tau, z_t, z_{t+1})$ to match this rectified-flow velocity, yielding

$$\mathcal{L}_{\text{FM}} = \mathbb{E}\Big[\big\|v_\xi(a_\tau, \tau, z_t, z_{t+1}) - (a_1 - a_0)\big\|_2^2\Big],$$

where the expectation is taken over transitions from the offline dataset, $\tau \sim \mathcal{U}(0, 1)$, $a_0 \sim \mathcal{N}(0, I)$, and latent variables sampled from the posterior (e.g., $z_t \sim q_\varphi(z_t \mid u_t, h_t)$ and $z_{t+1} \sim q_\varphi(z_{t+1} \mid u_{t+1}, h_{t+1})$).

The gradients of $\mathcal{L}_{\text{FM}}$ backpropagate through (z_t, z_{t+1}) into the inference network q_φ, effectively steering the latent space to align with the action manifold. Crucially, this dynamics-aligned structure learned in the posterior is transferred to the deployment prior $p_\eta(z_t \mid h_t)$ through the KL term in $\mathcal{L}_{\text{VAE}}$, yielding action-informed belief modeling with learned priors. Our final objective is

$$\mathcal{L}_{\text{Total}} = -\mathcal{L}_{\text{VAE}} + \lambda\, \mathcal{L}_{\text{FM}}.$$

3.3 Online Reinforcement Learning

During policy deployment, we need to infer the belief distribution over the current latent state conditioned on the observation history h. We approximate this distribution via Monte Carlo sampling by drawing a set of representative samples from the generative model. Concretely, we first sample latent variables $z_i \sim p_\eta(z \mid h)$ using the learned history-conditioned prior, and then feed each z_i into the decoder to produce the corresponding state encoding $\hat{u}_i$. The resulting history-driven sample set $(\hat{u}_1, \ldots, \hat{u}_n)$ constitutes the belief state $b(h)$ used by the agent for decision making. We use the following network architecture to encode the samples and perform the aggregation:

$$W\big(b(h_t)\big) = W_{\text{agg}}\left(\frac{1}{n}\sum_{i=1}^{n} W_{\text{enc}}(\hat{u}_{t,i})\right),$$

where W_{enc} and W_{agg} are neural networks.

Finally, we concatenate the inferred belief $b(h)$ with the current observation o and use them as inputs to the PPO actor and critic networks. Our work in this section is similar to prior work (Zaheer et al., 2017; Wang et al., 2023). During PPO online training, we freeze the belief-related parameters, while leaving the belief aggregator parameters trainable.

4 Experiment

4.1 Experiment Environment

We conduct our experiments on the MIKASA-Robo benchmark (Cherepanov et al., 2025a), which categorizes tasks into three difficulty levels: Easy, Medium, and Hard. In the MIKASA-Robo evaluations, conventional online methods such as PPO and PPO with LSTM, as well as offline approaches including Diffusion Policy and Decision Transformer, perform poorly even on Easy or Medium tasks. As a result, MIKASA-Robo has recently been regarded as a highly challenging memory-intensive robotic manipulation benchmark.

Our experiments primarily focus on Medium and Hard tasks to demonstrate that our privileged-information-based method can perform well under extremely challenging conditions. Specifically, we evaluate four tasks: RememberColor9, RememberShapeAndColor3x2, InterceptMedium, and InterceptFast. To further illustrate the difficulty, consider RememberColor9: among the offline methods reported on the MIKASA-Robo benchmark, the best one achieves only 0.17 success rate

(Cherepanov et al., 2025a), and a more recent state-of-the-art method targeting memory tasks improves this only to 0.23 (Cherepanov et al., 2025b). In our experiments, online methods like vanilla PPO perform even worse, highlighting the substantial difficulty of this task suite.

The MIKASA-Robo environments consist primarily of a robotic arm and several tabletop objects. Each task involves an episode-specific variable that must be remembered. Further details of the environment are provided in Appendix A.

4.2 Experiment Setup

Our experimental results focus on comparing three algorithms: vanilla PPO (Schulman et al., 2017), a strong baseline Believer + PPO (Wang et al., 2023), and our Action-Informed Believer + PPO. Since the MIKASA-Robo benchmark has already shown the weak performance of methods such as LSTM+PPO on this suite, we do not repeat those evaluations. Instead, in our subsequent experiments, we focus on comparing against Believer, which has demonstrated substantial improvements over vanilla PPO. We choose these two baselines for the following reasons. PPO is a classic and strong baseline for robotic control, and we aim to examine whether it can maintain its strong performance on the memory-intensive MIKASA-Robo benchmark. In addition, Believer has demonstrated excellent performance on POMDP problems, making it a suitable and strong baseline for memory-intensive robotic manipulation tasks, which are also naturally formulated as POMDPs.

For a fair comparison, we keep all shared architectural and training hyperparameters identical across the three algorithms on the same task. For example, when comparing AIB with the Believer baseline, we use the same PPO hyperparameters and the same VAE architecture/training settings for each task. The only additional component in AIB is the flow-matching inverse dynamics objective, which introduces one extra hyperparameter specific to this loss. After training the VAE with the same hyperparameter settings, we evaluate our module in reinforcement learning environments using the same three random seeds. We provide more training details in Appendix B.

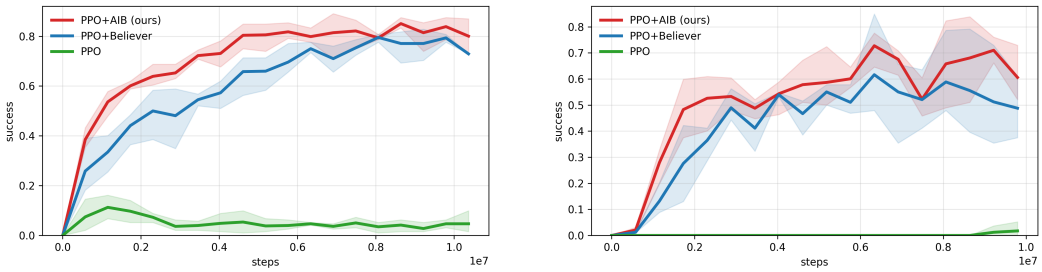
4.3 Main Result

We represent the curve with success_once metrics under evaluation mode, which is in the Figure 2. We also average the results from the last three evaluations and report them in Table 1 to quantitatively analyze the advantage of our method. We also report the success_at_end metrics and additional metric comparisons like the maximum achieved performance in Appendix C. Overall, our method improves over all baselines on three of four tasks, with varying margins. One notable exception is InterceptMedium, where PPO eventually surpasses our method late in training. However, on the other key metric, success end presented in the Appendix C, PPO's performance on the InterceptMedium task remains close to zero, which indicates that the surpass of PPO in this task is unusual.

Table 1: Eval success (success_once). Task names are abbreviated (see Appendix B). Results are mean±std over 3 seeds. Best results are **bolded**.

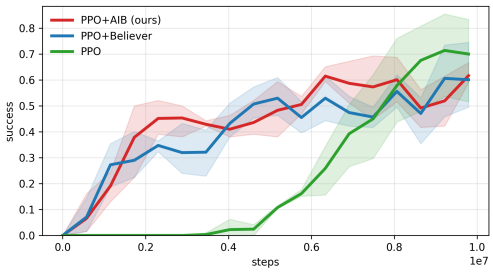
Task	PPO+AIB	PPO+Believer	PPO
RSC3x2	**0.82**±0.04	0.76±0.02	0.04±0.02
RC9	**0.36**±0.07	0.35±0.03	0.17±0.03
IMed	0.54±0.06	0.56±0.10	**0.70**±0.14
IFast	**0.67**±0.07	0.52±0.14	0.01±0.01
Avg	**0.60**±0.18	0.55±0.17	0.23±0.29

In addition, although all three methods exhibit some instability on the hardest tasks (e.g., InterceptFast), our approach is consistently faster in both improvement and convergence than Believer+PPO. For example, on RememberShapeAndColor3x2, our method reaches the final success rate achieved by Believer at 10M steps already at around 4M steps. We believe this is because the belief learned with the inverse dynamics auxiliary objective yields a more task-relevant representation space, which helps the RL agent adapt to the environment more quickly in the early stages of training.

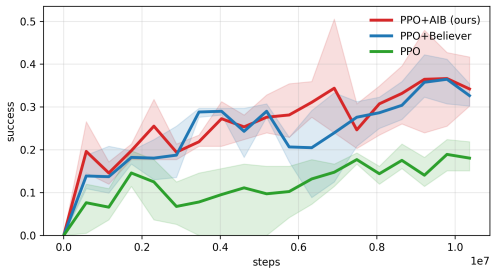


(a) RememberShapeAndColor3x2

(b) InterceptFast



(c) InterceptMedium



(d) RememberColor9

Figure 2: Success_once results across tasks on 3 seeds. **Red**: PPO+AIB (Ours), **Blue**: PPO+Believer, **Green**: PPO.

4.4 Ablation Study

We further discuss whether the flow-matching-based inverse dynamics objective is truly necessary. In addition, our method involves two key innovations, namely the learnable prior and the action head. In this ablation study, we also completely remove the action head in order to explicitly isolate and examine the contribution of the learnable prior alone to the performance improvement. We conduct an ablation study on InterceptFast, which is categorized as a hard task in MIKASA-Robo. In this task, the robotic arm must catch a small ball rolling at high speed with varying velocities. We believe this task critically depends on the robot's ability to capture fast-changing environment dynamics, and therefore best reflects the necessity of inverse dynamics. Keeping all other components fixed, we compare two variants of our method to isolate the effect of inverse dynamics: (1) removing the action head entirely, and (2) using a standard inverse dynamics objective that replaces flow matching with a conventional regression loss (e.g., MSE) to predict actions. Our results in the Figure 3 show that the flow-matching-based inverse dynamics achieves superior performance on this dynamics-sensitive task and the necessity of our flow-matching-based action head.

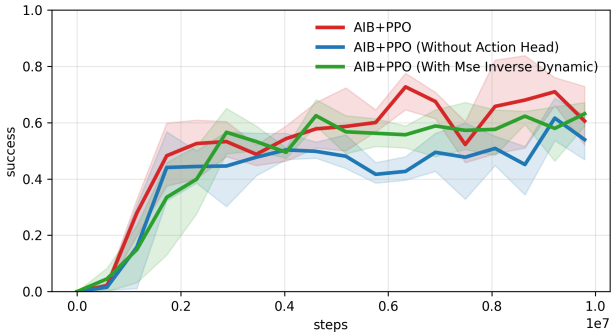


Figure 3: Ablation Results on InterceptFast-v0 on 3 seeds (success_once). **Red**: AIB+PPO, **Blue**: AIB+PPO (Without Action Head), **Green**: AIB+PPO (with MSE Inverse Dynamic).

4.5 Effect of the Flow-Matching Loss Weight on Training Stability

In our experiments, we found that although we had already adopted relatively small parameter values to train the model more conservatively, the model still exhibited instability, such as high variance. To further investigate this phenomenon, we analyzed the effect of our action-informed belief method on model stability. Specifically, we compared the impact of the flow matching loss weight on both success rate and stability in the 4.

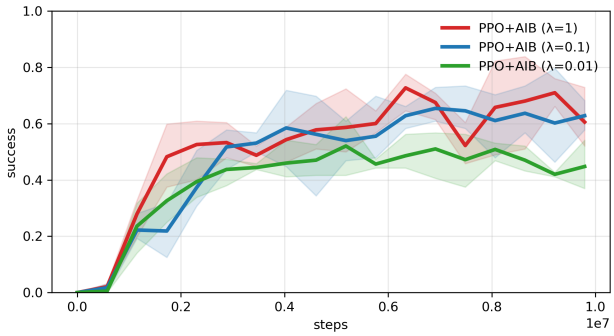


Figure 4: Sensitivity of training stability and final performance to the flow-matching loss weight on InterceptFast-v0 (3 seeds, success_once). **Red**: AIB+PPO (λ=1), **Blue**: AIB+PPO (λ=0.1), **Green**: AIB+PPO (λ=0.01). Where λ represents the weight for the flow matching inverse dynamic loss. over the last three evaluations and then across seeds gives 0.666 ± 0.072 for $\lambda = 1$, 0.623 ± 0.081 for $\lambda = 0.1$, and 0.446 ± 0.011 for $\lambda = 0.01$

The experimental results show that when $\lambda = 0.01$, the action head has only a limited effect on the model, resulting in a relatively low success rate, but it can effectively improve training stability. When $\lambda = 1$, although the variance of the model increases to some extent, the success rate is significantly improved.

5 Discussion

We propose an action-informed belief modeling method that applies a flow-matching-based inverse dynamics objective in the latent space to address the decoupling between belief modeling and control in prior work. In existing approaches, the VAE-based belief model exhibits a mismatch between training and inference during subsequent reinforcement learning, and its architecture is not compatible with our goal of introducing an inverse dynamics auxiliary task in the latent space. To enable this idea, we augment the original VAE framework with a learnable prior distribution. Our method outperforms current strong baselines on challenging memory-intensive robotic manipulation tasks.

Meanwhile, we acknowledge that our method still has limitations. Although it improves the average success on hard tasks, it still appears to suffer from instability. In our implementation, we found that this may be partly due to the need for further PPO hyperparameter tuning, and partly due to the nature of memory tasks themselves: when the policy is not yet sufficiently accurate, success or failure in remembering can induce a large reward gap. Our work primarily tackles the memory for robotics from the perspective of improving belief modeling; however, we believe it is a promising direction to explore how to better integrate the learned belief information into the reinforcement learning architecture (e.g., PPO) to further enhance training stability and performance.

Since we train an additional action head, we also experimented with replacing the random data used for belief training with expert data, aiming to pretrain an action head that outputs expert actions. However, based on our preliminary attempts, expert data seems ill-suited for belief training: in several trials, both the Believer baseline and our method performed poorly when trained on expert data. We leave a deeper investigation of this phenomenon as a promising direction for future work.

REFERENCES

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*, 2016.

Andrea Baisero and Christopher Amato. Unbiased asymmetric actor-critic for partially observable reinforcement learning. *CoRR*, abs/2105.11674, 2021.

David Brandfonbrener, Ofir Nachum, and Joan Bruna. Inverse dynamics pretraining learns good representations for multitask imitation. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Anthony R. Cassandra, Michael L. Littman, and Nevin Lianwen Zhang. Incremental pruning: A simple, fast, exact method for partially observable markov decision processes. In Dan Geiger and Prakash P. Shenoy (eds.), *UAI '97: Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence, Brown University, Providence, Rhode Island, USA, August 1-3, 1997*, pp. 54–61. Morgan Kaufmann, 1997.

Xiaoyu Chen, Yao Mark Mu, Ping Luo, Shengbo Li, and Jianyu Chen. Flow-based recurrent belief state learning for pomdps. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 3444–3468. PMLR, 2022.

Egor Cherepanov, Alexey Staroverov, Dmitry Yudin, Alexey K. Kovalev, and Aleksandr I. Panov. Recurrent action transformer with memory, 2024.

Egor Cherepanov, Nikita Kachaev, Alexey K. Kovalev, and Aleksandr I. Panov. Memory, benchmark & robots: A benchmark for solving complex tasks with reinforcement learning, 2025a.

Egor Cherepanov, Alexey K. Kovalev, and Aleksandr I. Panov. ELMUR: external layer memory with update/rewrite for long-horizon RL. *CoRR*, abs/2510.07151, 2025b.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory, 2025.

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

Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G Bellemare. Deepmdp: Learning continuous latent space models for representation learning. In *International conference on machine learning*, pp. 2170–2179. PMLR, 2019.

Danijar Hafner, Timothy P. Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps, 2017.

Nicolas Heess, Jonathan J Hunt, Timothy P Lillicrap, and David Silver. Memory-based control with recurrent neural networks. *arXiv preprint arXiv:1512.04455*, 2015.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, Senjie Jin, Jiejun Tan, Yanbin Yin, Jiongnan Liu, Zeyu Zhang, Zhongxiang Sun, Yutao Zhu, Hao Sun, Boci Peng, Zhenrong Cheng, Xuanbo Fan, Jiaxin Guo, Xinlei Yu, Zhenhong Zhou, Zewen Hu, Jiahao Huo, Junhao Wang, Yuwei Niu, Yu Wang, Zhenfei Yin, Xiaobin Hu, Yue Liao, Qiankun Li, Kun Wang, Wangchunshu Zhou, Yixin Liu, Dawei Cheng, Qi Zhang, Tao Gui, Shirui Pan, Yan Zhang, Philip Torr, Zhicheng Dou, Ji-Rong

Wen, Xuanjing Huang, Yu-Gang Jiang, and Shuicheng Yan. Memory in the age of ai agents. *arXiv preprint arXiv:2512.13564*, 2025.

Maximilian Igl, Luisa M. Zintgraf, Tuan Anh Le, Frank Wood, and Shimon Whiteson. Deep variational reinforcement learning for pomdps. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2122–2131. PMLR, 2018.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Paul Foster, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Conference on Robot Learning, 6-9 November 2024, Munich, Germany*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 2024.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun (eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Vijay R. Konda and John N. Tsitsiklis. Actor-critic algorithms. In Sara A. Solla, Todd K. Leen, and Klaus-Robert Müller (eds.), *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*, pp. 1008–1014. The MIT Press, 1999.

Hung Le, Dung Nguyen, Kien Do, Sunil Gupta, and Svetha Venkatesh. Stable hadamard memory: Revitalizing memory-augmented agents for reinforcement learning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

Steven D. Morad, Ryan Kortvelesy, Stephan Liwicki, and Amanda Prorok. Reinforcement learning with fast and forgetful memory. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018.

Stéphane Ross, Joelle Pineau, Brahim Chaib-draa, and Pierre Kreitmann. A bayesian approach for learning and planning in partially observable markov decision processes. *Journal of Machine Learning Research*, 12(48):1729–1770, 2011.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

Ajay Sridhar, Jennifer Pan, Satvik Sharma, and Chelsea Finn. Memer: Scaling up memory for robot control via experience retrieval, 2025.

Andrew Wang, Andrew C. Li, Toryn Q. Klassen, Rodrigo Toro Icarte, and Sheila A. McIlraith. Learning belief representations for partially observable deep RL. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 35970–35988. PMLR, 2023.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*, 2020.

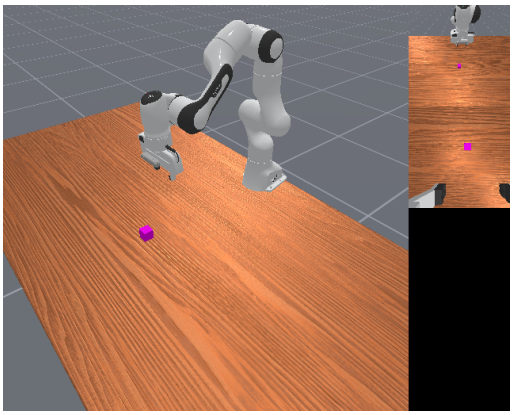
Marvin Zhang, Zoe McCarthy, Chelsea Finn, Sergey Levine, and Pieter Abbeel. Learning deep neural network policies with continuous memory states. In *2016 IEEE international conference on robotics and automation (ICRA)*, pp. 520–527. IEEE, 2016.

A Environment Details

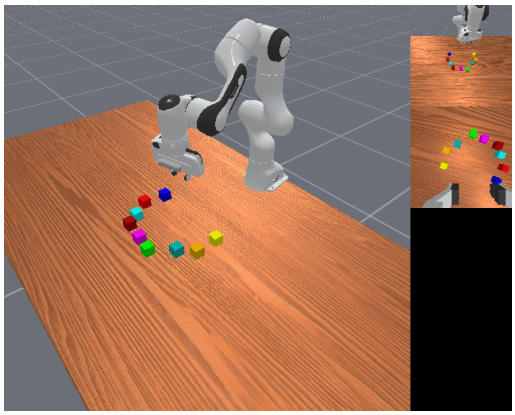
We provide a detailed description of our experimental environments. MIKASA-Robo reports two success metrics: success_once and success_at_end. A trajectory is counted as success_once if the agent achieves success at any point within the episode, and as success_at_end if the agent remains successful at the end of the episode.

A.1 RememberColor

The RememberColor environment consists of a robotic arm and several tabletop objects. At the beginning of each episode, the environment presents the target object to be remembered; the object then disappears, and multiple objects with identical shapes but different colors appear on the table. The robot succeeds if it correctly identifies the target object and touches it with the robotic arm. The environment has three variants: RememberColor3, RememberColor5, and RememberColor9. They differ in the total number of objects presented, while the number of target objects to remember remains one, resulting in an increasing difficulty from easy to medium to hard. We represent the environment of RememberColor9 as a example in the Figure 5



(a) The initial presentation stage of the RememberColor9 task, during which the colors of the objects to be remembered are shown.



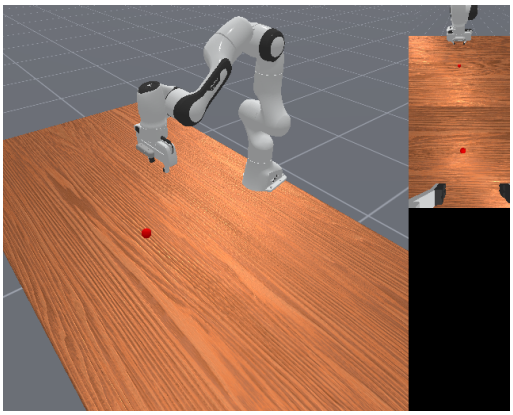
(b) The next stage of the RememberColor9 task, in which the robotic arm is required to select, from nine objects of different colors, the one whose color was just presented.

Figure 5: The task environment for RememberColor9. Panel 5a shows the initial presentation stage, and panel 5b shows the selection stage.

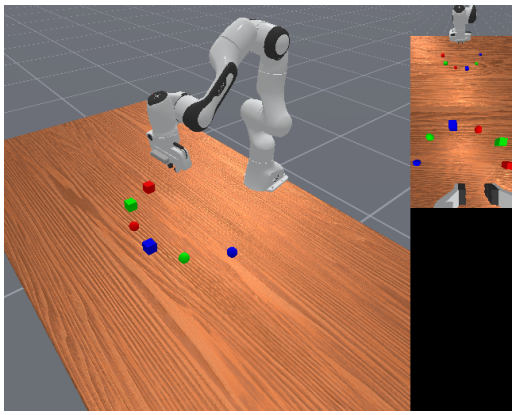
A.2 RememberShapeAndColor

The RememberShapeAndColor environment consists of a robotic arm and multiple tabletop objects. At the beginning of each episode, the environment briefly presents a target object that the robot must remember; the object then disappears, and a set of objects appears on the table. These objects vary in both shape and color, requiring the robot to recall the target's attributes and identify the correct instance among visually similar distractors. The robot succeeds if it touches the correct object with the robotic arm. The environment includes three variants: RememberShapeAndColor3x2, RememberShapeAndColor3x3, and RememberShapeAndColor5x3, where the notation indicates the

number of shapes and colors, respectively. As the number of possible shape–color combinations increases, the task becomes progressively more difficult from easy to medium to hard. We represent the environment of RememberShapeAndColor3x2 as a example in the Figure 6



(a) The initial presentation stage of the RememberShapeAndColor3x2 task, during which the colors of the objects to be remembered are shown.

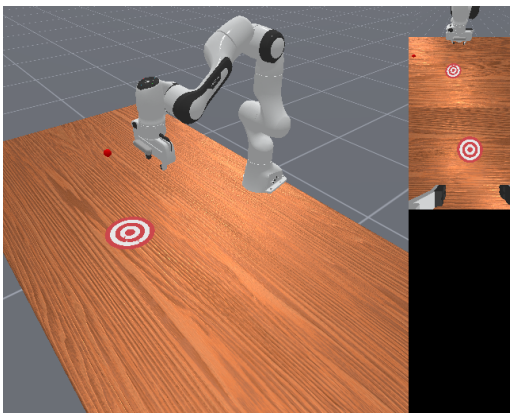


(b) The initial presentation stage of the RememberShapeAndColor3x2 task, during which the colors of the objects to be remembered are shown.

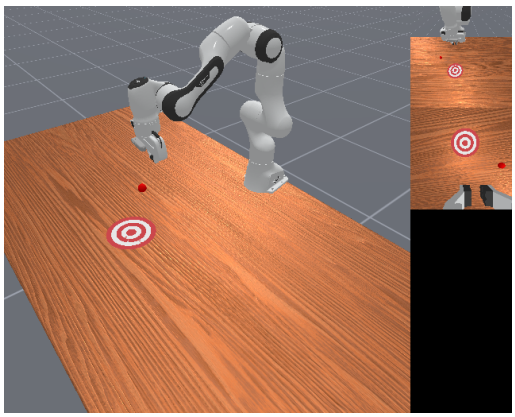
Figure 6: The task environment for RememberShapeAndColor3x2. Panel 6a shows the initial presentation stage, and panel 6b shows the selection stage.

A.3 INTERCEPT

In the Intercept suite, the environment consists of a robotic arm, a small ball on a tabletop, and a target region. In each episode, the ball rolls from one side of the table to the other at a randomly sampled speed, and the robot must push the ball into the target region during its motion. The suite includes three variants—InterceptSlow, InterceptMedium, and InterceptFast—which correspond to different speed ranges of the ball and therefore represent increasing difficulty from easy to medium to hard. We represent the environment of InterceptFast as a example in the Figure 7



(a) In the InterceptFast task, the ball moves rapidly forward at the start. This figure shows the ball leaving its starting point.



(b) This figure shows the ball after it has rolled some distance following the start of the InterceptFast task.

Figure 7: This figure shows the environment of the InterceptFast task. Panels 7a and 7b depict the initial state of the ball's motion and its state after moving for a period of time, respectively.

B HYPERPARAMETERS

Here, we report the hyperparameters and some implementation details used in our training. All three PPO-based algorithms are trained using the same dataset/configuration for PPO training. Among the training-related hyperparameters, we only slightly tune a small number of parameters (e.g., update_epochs; the guiding principle is that harder tasks require slower updates to stabilize PPO convergence. You can find the specific hyperparameters in Tables 2, 3, 4, and 5.

Across different tasks, we keep the hyperparameters of the first two belief-training stages—i.e., the representation model and the belief model (VAE)—fixed and consistent. However, due to differences in task difficulty and the well-known sensitivity of PPO to hyperparameter choices, we make minor adjustments (typically 1–2 hyperparameters) between tasks of different difficulty levels, while keeping the vast majority of settings unchanged. Our adjustment principle is that harder tasks require slower PPO updates to ensure stable convergence. In our empirical tests, using the same PPO configuration for hard tasks as for easy ones often led to training collapse in later stages.

For the random data we collect for the representation learning and belief modeling. We note that the original MIKASA-Robo environment does not natively support obtaining both obs and state in the required format. Therefore, we make minor modifications to certain conditional checks in the MIKASA-Robo codebase so that the environment can return the state alongside the obs for dataset construction. Importantly, this modified environment is used only for data collection; when training PPO, we still use the original, unmodified MIKASA-Robo environment. Although the number of VAE epochs may appear somewhat high compared to common practice, this setting was obtained by strictly following and tuning based on the Believer baseline implementation. From the PPO learning curves, training converges well, and we do not observe signs of overfitting.

Table 2: VAE training hyperparameters.

PARAMETER	VALUE
LR	3E-4
EPOCHS	10000
BATCH-SIZE	256
LATENT-DIM	32
BETA	1.0

Table 3: AIB training hyperparameters.

PARAMETER	VALUE
LR	3E-4
EPOCHS	10000
BATCH-SIZE	256
LATENT-DIM	32
BETA	1.0
ACTION-LAMBDA	1.0

C ADDITIONAL RESULTS' METRICS

Here, we present additional metrics for the four tasks used in the main experiments, including the learning curves for success_at_end in Figure 8 and quantitative results from the last three evaluations in Table 6, as well as quantitative results in evaluation mode for the maximum success rates of success_at_end and success_once in the Table 7 and Table 8, respectively.

Table 4: Representation-model training hyperparameters.

PARAMETER	VALUE
LR	3E-4
EPOCHS	200
BATCH-SIZE	500
LATENT-DIM	16
BETA	0.03
DYNAMICS-LOSS-S-COEF	0.3
DYNAMICS-LOSS-O-COEF	0.03
REWARD-LOSS-COEF	10

Table 5: PPO training hyperparameters across four tasks (RSC3x2: RememberShapeAndColor3x2-v0; RC9: RememberColor9-v0; IMed: InterceptMedium-v0; IFast: InterceptFast-v0).

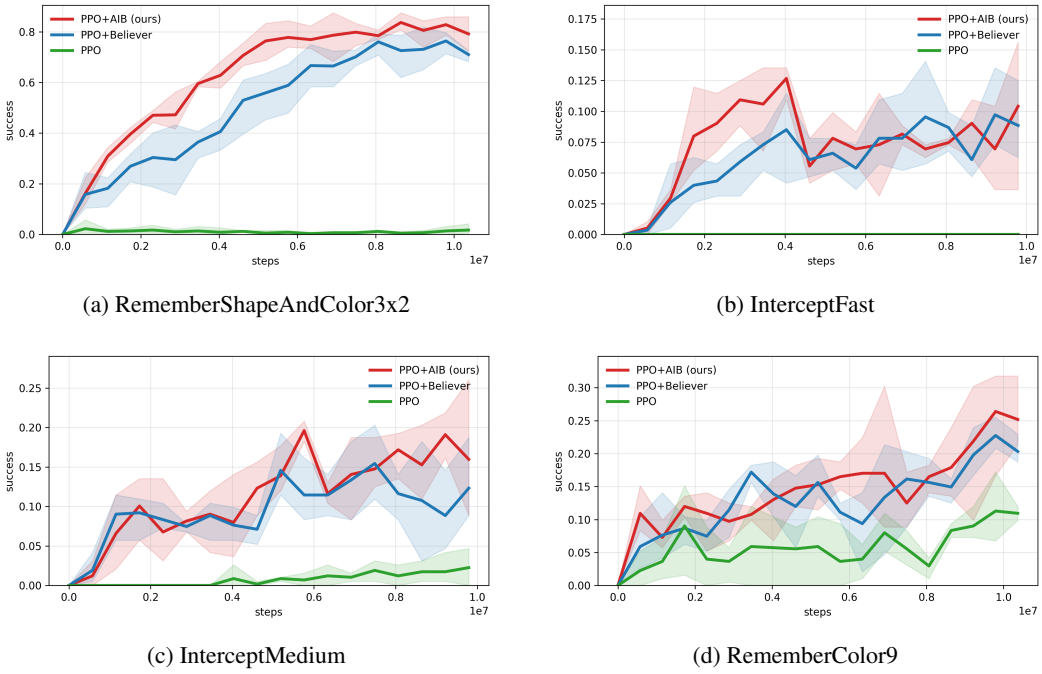
PARAMETER	RSC3X2	RC9	IMED	IFAST
TOTAL TIMESTEPS	10,380,000	10,380,000	10,000,000	10,000,000
NUM ENVS	200	200	256	256
ROLLOUT STEPS (NUM-STEPS)	60	60	90	90
LEARNING RATE	1E-4	1E-4	1E-4	1E-4
UPDATE EPOCHS	4	2	2	2
DISCOUNT FACTOR γ	0.95	0.95	0.99	0.99
GAE LAMBDA λ	0.9	0.9	0.9	0.9
ENTROPY COEF	0.001	0.001	0.001	0.001
TARGET KL	0.05	0.05	0.05	0.05
FINITE-HORIZON GAE	FALSE	FALSE	FALSE	FALSE
OBSERVATION	RGB + JOINTS	RGB + JOINTS	RGB + JOINTS	RGB + JOINTS

Table 6: Eval success (success_at_end). Task names are abbreviated (see Appendix B). Results are mean±std over 3 seeds.

TASK	PPO+AIB	PPO+BELIEVER	PPO
RSC3X2	**0.81**±0.03	0.74±0.04	0.01±0.01
RC9	**0.25**±0.06	0.21±0.01	0.10±0.01
IMED	**0.17**±0.04	0.11±0.03	0.02±0.02
IFAST	**0.09**±0.03	0.08±0.02	0.00±0.00
AVG	**0.33**±0.29	0.28±0.27	0.03±0.04

Table 7: Eval success (success_once). Task names are abbreviated (see Appendix B). Results are mean±std over seeds.

TASK	PPO+AIB	PPO+BELIEVER	PPO
RSC3X2	**0.87±0.01**	0.82±0.01	0.14±0.03
RC9	**0.41±0.08**	0.38±0.03	0.20±0.02
IMED	0.68±0.01	0.65±0.07	**0.71±0.13**
IFAST	**0.77±0.06**	0.69±0.12	0.02±0.02
AVG	**0.69±0.18**	0.63±0.17	0.27±0.28



(a) RememberShapeAndColor3x2

(b) InterceptFast

(c) InterceptMedium

(d) RememberColor9

Figure 8: Success_at_end results across tasks on 3 seeds. **Red**: PPO+AIB (Ours), **Blue**: PPO+Believer, **Green**: PPO.

Table 8: Eval success (success_at_end). Task names are abbreviated (see Appendix B). Results are mean±std over seeds.

TASK	PPO+AIB	PPO+BELIEVER	PPO
RSC3X2	**0.86±0.01**	0.80±0.02	0.04±0.01
RC9	**0.28±0.05**	0.23±0.02	0.15±0.02
IMED	**0.23±0.02**	0.18±0.03	0.03±0.01
IFAST	**0.14±0.01**	0.12±0.02	0.00±0.00
AVG	**0.38±0.28**	0.33±0.27	0.05±0.06

D Related Work

D.1 Recurrent Deep Reinforcement Learning

Reinforcement learning (RL) has long been studied as a framework for sequential decision-making, but early approaches often struggled to scale to environments with high-dimensional observations. The combination of deep learning and RL enabled policies and value functions to be represented by neural networks, leading to the success of deep reinforcement learning (DRL) methods such as DQN (Mnih et al., 2013), which achieved strong performance on high-dimensional visual domains including Atari games.

Beyond value-based methods, actor-critic algorithms provide a principled way to optimize parameterized policies, and have become a widely adopted backbone for modern DRL (Konda & Tsitsiklis, 1999). Proximal Policy Optimization (PPO) further improves the stability and practicality of actor–critic training and is commonly used as a strong baseline across a wide range of continuous control benchmarks (Schulman et al., 2017).

To address partial observability, recurrent architectures have been integrated into DRL to enable agents to aggregate information over time. DRQN Hausknecht & Stone (2017) incorporates LSTM Hochreiter & Schmidhuber (1997) into the DQN framework, improving performance in POMDP settings by conditioning on observation histories rather than only the current observation. More broadly, recent work has explored various memory mechanisms for RL, including designs inspired by cognitive science (Morad et al., 2023), as well as methods that analyze and mitigate training instabilities such as vanishing gradients in memory-dependent RL tasks (Le et al., 2025).

D.2 Exploiting Privilege Information

Early approaches to solving POMDPs primarily relied on explicitly modeling the environment dynamics. For example, Cassandra et al. (1997) proposed an efficient dynamic programming method for solving POMDPs by exploiting structured planning.

Ross et al. (2011) model belief states via Bayesian updates. Hafner et al. (2021) introduced deep belief representations by combining gaussian distribution with deep neural networks, achieving strong performance on challenging partially observable domains such as Atari. Chen et al. (2022) further improved the expressiveness of belief modeling by incorporating normalizing flows to capture more complex distributions.

Igl et al. (2018) proposed a POMDP method for reinforcement learning that uses variational inference to model environment beliefs, outperforming traditional approaches that rely solely on neural network memory.

Pinto et al. (2018) introduced a privileged-information training paradigm for robotic RL. Since robotic observations are often partially observable, their method stabilizes learning by training the critic with access to the full state instead of raw observations, improving robustness under partial observability. Baisero & Amato (2021) analyzed the theoretical foundations of asymmetric training and showed that it can introduce bias. They further proposed an unbiased asymmetric actor–critic variant to address this issue. Wang et al. (2023) proposes a belief modeling approach that combines representation learning with a VAE. During training, it leverages and optimizes state representations, and then uses a VAE to learn the mapping from interaction history to the belief representation. They model uncertainty in POMDPs by sampling the latent variable from a Gaussian distribution.

D.3 Memory for Robotics

Although the notion of memory has attracted attention in robot learning for quite some time (Zhang et al., 2016; Heess et al., 2015). However, in recent years, as the concept of memory has received increasing attention, more works specifically targeting memory-intensive robotic tasks have gradually emerged. Cherepanov et al. (2024) incorporates recurrent neural networks to provide a memory mechanism for the Decision Transformer architecture. Cherepanov et al. (2025b) proposes a least-recently-used (LRU)-based memory strategy, further strengthening the memory mechanism of Decision Transformer–based robotics. Sridhar et al. (2025) improves the memory of robotics under

VLA architectures by introducing a novel hierarchical memory design. At present, there is still no mature and effective solution for memory-intensive robotic tasks.

E PRELIMINARY

E.1 REINFORCEMENT LEARNING

Reinforcement learning (RL) is a learning paradigm in which an agent improves its behavior through trial-and-error interactions with an environment. At each time step t, the agent observes the current state $s_t \in \mathcal{S}$, takes an action $a_t \in \mathcal{A}$, and receives a scalar reward r_t. The reward signal is used to improve the agent's policy, with the goal of maximizing long-term cumulative returns.

We model the environment as a Markov decision process (MDP), defined by

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, R, \gamma),$$

where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, $T(s' \mid s, a)$ is the state transition distribution, $R(s, a)$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor

At time step t, the environment transitions according to

$$s_{t+1} \sim T(\cdot \mid s_t, a_t), \qquad r_t = R(s_t, a_t).$$

E.2 PARTIALLY OBSERVABLE MARKOV DECISION PROCESSES

Partially observable Markov decision processes (POMDPs) extend MDPs by modeling limited observability in the real world or in the memory questions. A POMDP is defined by the tuple

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, T, R, \Omega, \gamma),$$

where $\mathcal{O}$ is the observation space and $\Omega(o \mid s)$ denotes the observation distribution. Unlike in fully observable MDPs, the agent cannot directly access the underlying state s_t, but instead receives an observation

$$o_t \sim \Omega(\cdot \mid s_t),$$

and selects actions based on these observations.

Belief states. In a POMDP, a belief state is the posterior distribution over latent states given the history h_t:

$$b_t(s) = \Pr(s_t = s \mid h_t), \quad h_t = (o_0, a_0, \ldots, o_t).$$

In deep RL settings, we often approximate belief states using a learned inference model. Following prior work, we consider a learned belief over a compact latent encoding $u_t = \phi(s_t)$, i.e.,

$$b(h_t) \approx p(u_t \mid h_t),$$

and learn a policy of the form

$$\pi(a_t \mid o_t, b(h_t)).$$