Learning Belief Representations for Partially Observable Deep RL

Andrew Wang [*1 2] **Andrew C. Li** [*1 2] **Toryn Q. Klassen** [1 2 3] **Rodrigo Toro Icarte** [4 5] **Sheila A. McIlraith** [1 2 3]

Abstract

Many important real-world Reinforcement Learning (RL) problems involve partial observability and require policies with memory. Unfortunately, standard deep RL algorithms for partially observable settings typically condition on the full history of interactions and are notoriously difficult to train. We propose a novel deep, partially observable RL algorithm based on modelling belief states — a technique typically used when solving tabular POMDPs, but that has traditionally been difficult to apply to more complex environments. Our approach simplifies policy learning by leveraging state information at training time, that may not be available at deployment time. We do so in two ways: first, we decouple belief state modelling (via unsupervised learning) from policy optimization (via RL); and second, we propose a representation learning approach to capture a compact set of reward-relevant features of the state. Experiments demonstrate the efficacy of our approach on partially observable domains requiring information seeking and long-term memory.

1. Introduction

The world is inherently partially observable. Agents (humans or otherwise) operating in real-world environments rarely have full information about environment state. In the context of Reinforcement Learning (RL), partial observability is identified as one of nine important challenges to the deployment of RL in real-world settings (Dulac-Arnold et al., 2021). Nevertheless, in many real-world settings we are able to train our RL agents under *full* or near-full observability using a simulator or other generative model of the environment. If the RL agent will subsequently be operating in a partially observable environment, we would like to ensure that the agent's learned policy is robust to any lack of observability it will encounter. Sometimes this partial observability is dynamic and unpredictable – perhaps the result of visual occlusion. In a subset of such cases, partial observability can be mitigated or rectified by the agent taking some action such as changing its viewpoint or opening a box to see what's inside. In other cases, aspects of the state that are unobservable at deployment time are known or can be inferred because of known sensor placement, such as the fixed sensors in a car engine or other electro-mechanical device. Our claim is that having full observability at learning time, combined with knowing what won't be observable at deployment time, enables an RL agent to learn a policy that is more robust to its partial observability. How we build RL systems to exploit this knowledge is the topic of this paper.

Motivated by this setting, we propose a novel approach for partially observable reinforcement learning (PO-RL) that exploits a fully observable state at training time. Our approach is inspired by traditional belief state methods for solving POMDPs (Monahan, 1982; Kaelbling et al., 1998; Silver & Veness, 2010), which naturally incorporate inference and memory. While these methods are limited to POMDPs with small, tractable state spaces and known models, our approach targets domains with complex, high-dimensional states and observations and unknown models. Furthermore, while many applied problems and their solutions involve estimating pre-determined state variables (e.g. localization (Lauri & Ritala, 2016), decentralized robotic coordination (Spaan et al., 2010); see Lauri et al. (2022) for more), our approach is general, assuming no such knowledge.

Our end-to-end approach achieves this via three key steps:

1. Representation Learning: we extract critical features of the state that are task-relevant and unobservable to the agent at deployment time to produce a compact state encoding;
2. Belief Modelling: we model the complex belief distribution over these state encodings given histories using variational autoencoders (Kingma & Welling, 2014);
3. Policy Learning: we incorporate the learned belief to serve as a memory for an RL policy.

[*]Equal contribution [1]Department of Computer Science, University of Toronto [2]Vector Institute [3]Schwartz Reisman Institute for Technology and Society [4]Pontificia Universidad Católica de Chile [5]Centro Nacional de Inteligencia Artificial. Correspondence to: Andrew Wang <andrewwang@cs.toronto.edu>, Andrew Li <andrewli@cs.toronto.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

This approach leads to a number of conceptual advantages over existing PO-RL approaches, which we highlight with examples throughout the paper. In particular, we claim that leveraging state information throughout training can provide a stronger learning signal than rewards or prediction of observations. We support this with experiments and ablations that demonstrate the effectiveness of our approach in partially observable image-based environments that require active information seeking, long-term memory, and probabilistic state inference, which pose a significant challenge to deep PO-RL methods (Liu et al., 2021; Pleines et al., 2023).

We summarize our contributions below.

- We propose a new end-to-end approach for RL under partial observability inspired by traditional belief state methods for POMDPs. Our method is scalable to settings with high-dimensional inputs, does not rely on a model of the environment, and makes no assumptions on the representation of states or observations.
- We outline the conceptual benefits of our approach on tasks involving long-term memory, active information seeking, and probabilistic inference, which pose a significant challenge to existing PO-RL methods.
- We demonstrate the advantages of our method through experiments and ablations on image-based MiniGrid environments (Chevalier-Boisvert et al., 2018) as well as a continuous-control environment with high-dimensional image observations.

2. Preliminaries

2.1. POMDPs

We consider an environment modelled as a *partially observable Markov Decision Process* (POMDP). Formally, a POMDP is a tuple $\langle \mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \omega, \gamma, \mu \rangle$ where $\mathcal{S}, \mathcal{O}, \mathcal{A}$ are the *state*, *observation* and *action* spaces, respectively, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta\mathcal{S}$ is the *state-action transition function*, $\omega : \mathcal{S} \to \Delta\mathcal{O}$ is the *observation probability distribution*, $\mathcal{R} : \mathcal{S} \times \mathcal{A}$ is the *reward function*, γ is the *discount factor* and μ is the *initial state distribution* (Åström, 1965; Kaelbling et al., 1998). Each *episode*, the environment is set to an initial state $s_0 \sim \mu(\cdot)$. At every step t, the agent observes $o_t \sim \omega(s)$, executes an action $a_t \in \mathcal{A}$, and receives reward $r_t = \mathcal{R}(s_t, a_t)$. The environment then transitions to the next state $s_{t+1} \sim \mathcal{P}(s_t, a_t)$.

Belief MDPs. One technique often used for solving tabular POMDPs with known transition and observation probabilities is to learn a policy over *belief states*. The belief state $b_t \in \Delta\mathcal{S}$ at time t is the probability distribution over states s_t given history h_t. Any POMDP can be equivalently formulated as an MDP over belief states (Kaelbling et al., 1998); thus the belief state b_t can be seen as a sufficient statistic of the history h_t towards deciding optimal actions.

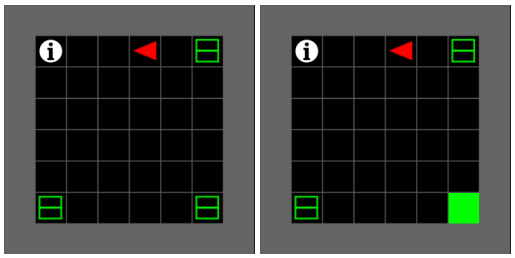

Figure 1. Visualization of the *Sphinx* running example. The agent must choose a single box to open from three boxes, one of which randomly contains a +1 reward. However, the agent can talk to the sphinx (represented by the information symbol), incurring a cost of -0.2 but revealing the correct box for a single timestep. **Left**: The agent's *observation*. The agent cannot see which box contains the +1 reward. **Right**: The corresponding *state* (unobservable to the agent at deployment time). The box at the bottom right corner contains the +1 reward.

2.2. Problem Statement

We begin with the reinforcement learning (Sutton & Barto, 2018) framework, where transition probabilities, observation probabilities, and the reward function are not known. The agent interacts with the environment by receiving observations o_t and executing actions a_t, with the goal of maximizing the expected discounted return $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_{t+1}]$. In POMDPs, the optimal policy π^* may depend on the entire history of observations and actions $h_t = (o_0, a_0, \ldots, o_{t-1}, a_{t-1}, o_t)$.

In this paper, we assume that both states s_t and observations o_t can be observed by the agent during training, while *only* observations o_t are observable during deployment. Our goal remains unchanged – we aim to learn a policy $\pi(a_t|h_t)$ that depends only on the observation-action history to maximize the expected discounted return when deployed. This setting has also been referred to as *asymmetric RL* or *offline learning/online execution* (Pinto et al., 2018; Baisero & Amato, 2022).

2.3. Running Example (*Sphinx*)

As a running example, we consider a partially observable environment called *Sphinx* (inspired by a meta-RL task by Liu et al. (2021)) that emphasizes information-seeking and long-term memory. Illustrated in Figure 1, the gridworld environment contains three boxes and a sphinx. One box contains a +1 reward (while the others contain nothing) and opening any box immediately ends the episode. The agent cannot observe which box contains the reward, but the agent can reveal it for a single timestep by speaking to the sphinx, for a cost of -0.2. The optimal strategy is to speak to the

sphinx (once), remember the correct box, and then visit that box, always yielding 0.8 return. If the agent does not speak to the sphinx, the agent must guess between the three boxes, yielding only 0.33 return on average.

As remarked by Liu et al. (2021), deep RL often struggles to find optimal solutions to such problems that require the agent to seek information due to an *exploration-exploitation dilemma*. Behaving optimally requires the agent to speak to the sphinx (*exploration*) – however, as the agent does not immediately know how to *exploit* the sphinx's advice, this will result in a net negative expected return for a large duration of training. Often, the agent will learn to stop speaking to the sphinx altogether, precluding it from learning an optimal policy.

We also observe that for this task, the *only* state information missing from the agent's observation is the identity of the box containing the reward. Thus, a belief over states can be compactly represented as the agent's current observation together with a categorical belief over the identity of the correct box. This categorical belief is a function of the history h_t and should assign $\frac{1}{3}$ probability for each box if the agent has never spoken to the sphinx, and otherwise, it should assign full probability mass to the box revealed by the sphinx.

3. Related Work

Model-free Deep RL. Most model-free deep RL algorithms can be adapted for partially observable environments by adding a recurrent network to the policy. Deep Recurrent Q-Networks (DRQN) (Hausknecht & Stone, 2015; Zhu et al., 2017) adds an LSTM (Hochreiter & Schmidhuber, 1997) to the DQN architecture (Mnih et al., 2013) to condition on past observations and/or actions. Heess et al. (2015) and Meng et al. (2021) introduced recurrent variants of the Deterministic Policy Gradient (Silver et al., 2014) algorithm. Zhang et al. (2016) trained memory-based policies using a combination of trajectory optimization and supervised learning. Toro Icarte et al. (2020) trained an agent to utilize an external memory for partially observable tasks.

Common partially observable RL tasks include flickering Atari, in which the screen is obscured with some probability, memorizing object locations (Heess et al., 2015), and continuous control from noisy sensors. Such tasks do not involve *information seeking*, since the agent does not have significant control over the information it can acquire.

Predicting the Future. Many approaches learn latent state representations in POMDPs that are predictive of future rewards and observations. Lee et al. (2020), Hafner et al. (2019), Wang & Tan (2021), and Chen et al. (2022) trained latent variable models to predict future observations or rewards, but were evaluated in environments with little need for long-term memory. Guo et al. (2018) learn belief representations in partially observable environments, but do not use the representations downstream for decision making. Variational Recurrent Neural Networks (VRNNs) (Chung et al., 2015) extend Variational Autoencoders (VAEs) (Kingma & Welling, 2014) to model sequential dependencies of latent variables. Han et al. (2020) adapted VRNNs into a control algorithm for POMDPs called Variational Recurrent Model (VRM) that takes actions into account. However, learning latent states that capture long-term dependencies is known to be challenging (Hafner et al., 2019).

Exploiting State Information. Traditional POMDP approaches often model belief states using a model of the environment. Monahan (1982), Littman (1994), and Cassandra et al. (1997) proposed exact methods for discrete POMDPs by representing value functions over beliefs as a set of hyperplanes. Monte-Carlo approaches (particle filtering) can be used to approximate belief state updates (Katt et al., 2017; Thrun, 1999; Silver & Veness, 2010). Unfortunately, these techniques are typically limited to POMDPs with small state spaces and known dynamics.

In the absence of an environment model, the *asymmetric RL* framework assumes observable states *only* during training. Pinto et al. (2018) proposed a heuristic actor-critic method where the critic conditions on ground-truth states (as the critic is not necessary for deploying the final policy) instead of histories. Baisero & Amato (2022) showed that such methods can exhibit bias for an agent under partial observability and proposed an unbiased actor-critic method that incorporates state information.

A number of deep RL problems that can be cast as POMDPs are often solved using privileged state information during training. Humplik et al. (2019) viewed the problem of meta-RL as inferring the task from a distribution. Liu et al. (2021) similarly leveraged encodings of the task as state information to improve meta-RL, but explicitly treated information gathering and reward exploitation as separate objectives. Reward functions specified in a formal language are often non-Markovian (Vaezipoor et al., 2021), and typically exploit privileged information about the task state during training and/or deployment (Tuli et al., 2022; Li et al.; Toro Icarte et al., 2019). The *centralized training, decentralized execution* paradigm for multi-agent RL (Sharma et al., 2021) assumes that the local observations of agents are common knowledge during training. For autonomous driving, Chen et al. (2020) trained a limited-observability agent (with image inputs) to imitate an agent with a full layout of the environment. Kumar et al. (2021) leveraged the state of a simulator to improve sim-to-real transfer of a robot.

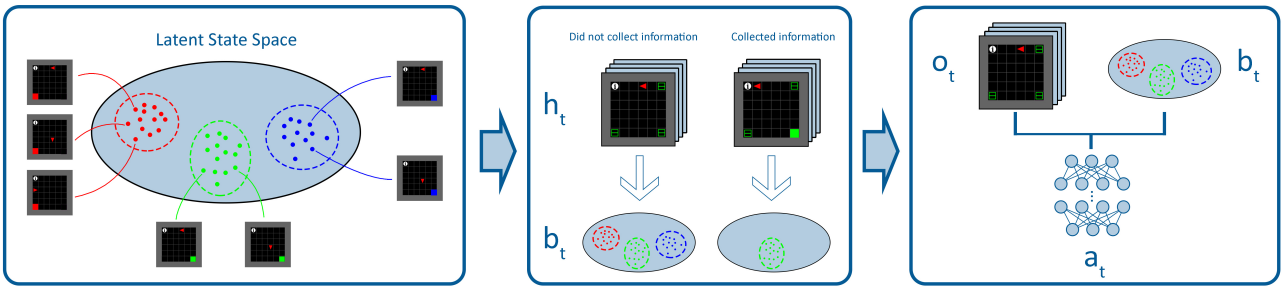

Figure 2. Illustrative diagram of the three training phases of Believer. **Left**: *Representation learning*. We train a stochastic mapping from states to latent representations that captures task-relevant, unobservable aspects of the state. **Center**: *Belief modelling*. Given histories, we predict a belief over latent states using VAEs. **Right**: *Policy learning*. We train a policy conditioned on the current observation and belief. The belief aids in long-term memory and probabilistic inference.

4. Method

Inspired by traditional POMDP algorithms that model a belief over states, we present a novel end-to-end approach, which we call **Believer**, for learning policies in partially observable, deep RL environments. To scale our approach to complex settings with high-dimensional state representations (e.g. images), our approach learns to model a belief over only task-relevant features of state. More precisely, our ultimate objective is to learn a policy of the form $\pi(a_t|o_t, b(h_t))$, where o_t is the current observation and $b(h_t)$ is a compact representation of a belief over relevant state variables, given the history h_t.

Our approach consists of three training phases (shown in Figure 2): (1) *representation learning*, (2) *belief modelling*, and (3) *policy learning*. Phases (1) and (2) assume access to an offline dataset of environment interactions (e.g. data generated by a random policy) labelled with ground-truth states. Unlike prior works (Lauri & Ritala, 2016; Spaan et al., 2010; Toro Icarte et al., 2019; Li et al.; Tuli et al., 2022), our method does not make assumptions about the structure or representation of states. In phase (3), we train a policy conditioned on the current observation and a learned belief over latent states. At deployment time, this policy can be deployed in a partially observable environment *without* access to ground-truth states.

4.1. Learning Compact Representations of States

In practice, belief state approaches for solving POMDPs are difficult to apply to deep RL domains, even when training with state information; states may take on high-dimensional representations, and the environment dynamics are usually unknown and challenging to model. To scale our approach to these settings, we propose to learn compact neural representations $\phi(s_t)$ that encode relevant aspects of the state that an agent should infer based on the observation-action history. We highlight two desirable properties of these learned representations. **(P1):** They should compactly capture task-relevant state features, while discarding all information irrelevant to future rewards. For example, with respect to the *Sphinx* example, $\phi(s_t)$ should capture the agent's position, direction, and the correct box in a lower-dimensional encoding than the original image, while ignoring any irrelevant noise (say, the colour of the boxes). **(P2):** Features of state (that may be task-relevant) should not be captured by $\phi(s_t)$ if they are also observable via o_t. In *Sphinx*, the agent only needs to infer which box contains the goal, based on the history; other aspects of the state (e.g. the agent's position) are always observable to the agent and do not need to be inferred.

To achieve (P1), we draw inspiration from prior representation learning works based on *bisimulation* in MDPs (Zhang et al., 2021; Kemertas & Aumentado-Armstrong, 2021; Gelada et al., 2019). Such methods typically train encoders $\phi(s_t)$ that, given actions a_t, are predictive of immediate rewards r_t and the distribution of next state encodings $\phi(s_{t+1})$. Unfortunately, these representations usually will not achieve (P2), i.e., they may contain redundant information that is always observable to the agent.

To also achieve this second property, we adapt these representation learning approaches to consider observations o_t in a partially observable setting. We train a joint encoding of state $(\phi(s_t), \psi(o_t))$ that considers both states and observations while minimizing the overlapping information between $\phi(s_t)$ and $\psi(o_t)$. Notice that observations o_t provide no additional information towards predicting future states, observations, and rewards than the current state s_t, by the Markov property. Nonetheless, s_t and o_t may duplicate information. Intuitively, our goal is to encode this redundant information into $\psi(o_t)$ but not $\phi(s_t)$, in order to produce a parsimonious representation $\phi(s_t)$.

We train the joint encoding $(\phi(s_t), \psi(o_t))$ to be predictive of immediate rewards r_t and the next joint encoding $(\phi(s_{t+1}), \psi(o_{t+1}))$, given actions a_t. This is accomplished by training a reward/latent dynamics model g alongside ϕ, ψ

Algorithm 1 Learning compact state representations.

Input: Dataset $\mathcal{D} = \{(s_i, o_i, a_i, r_i, s'_i, o'_i) : 1 \leq i \leq N\}$
repeat
  Sample data $(s, o, a, r, s', o') \sim \mathcal{D}$.
  Sample stochastic encodings $u_s \sim \phi(s), u_o \sim \psi(o)$.
  Predict rewards/dynamics $\hat{r}, \hat{u}_{s'}, \hat{u}_{o'} = g(u_s, u_o, a)$.
  Compute stochastic targets $u_{s'} \sim \phi(s'), u_{o'} \sim \psi(o')$.
  $\mathcal{L}_r = (r - \hat{r})^2$
  $\mathcal{L}_s = ||\text{stop\_grad}(u_{s'}) - \hat{u}_{s'}||^2$
  $\mathcal{L}_o = ||\text{stop\_grad}(u_{o'}) - \hat{u}_{o'}||^2$
  $\mathcal{L}_{\text{KL}} = \text{KL}\big[\phi(s)||\mathcal{N}(0, 1)\big]$
  Update ϕ, ψ, g on loss $\lambda_r\mathcal{L}_r + \lambda_s\mathcal{L}_s + \lambda_o\mathcal{L}_o + \lambda_{\text{KL}}\mathcal{L}_{\text{KL}}$.
until convergence
Output: State encoder ϕ.

using an offline dataset of state-labelled experiences. To ensure that duplicated information is encoded in $\psi(o_t)$ but not $\phi(s_t)$, we apply a variational information bottleneck (Tishby et al., 2000; Alemi et al., 2017) to ϕ while treating ϕ, ψ as stochastic encoders. In this work, ϕ, ψ are Gaussian with learned mean and standard deviation, and gradients through ϕ, ψ are computed using the reparameterization trick. At the end of Algorithm 1, the result is: (1) state features that are *unobservable* but *task-relevant* are encoded by $\phi(s_t)$; (2) state features that are *observable* and *task-relevant* are encoded by $\psi(o_t)$. Finally, we discard all models except for $\phi(s_t)$, which encodes only *task-relevant* and *unobservable* information — the minimum information necessary for an optimal policy, given observations.

4.2. Belief Modelling with VAEs

With the state encoder ϕ in hand, our next goal is to model the belief distribution over relevant state features $p(\phi(s_t)|h_t)$ for any observation-action history h_t. We turn to variational autoencoders (Kingma & Welling, 2014), which have shown to be capable of modelling complex distributions. Given an offline dataset $\mathcal{D}$ of state-labelled trajectories from the POMDP, our objective is to maximize the joint log-likelihood $p(\phi(s), h)$ averaged over $(s, h) \sim \mathcal{D}$.

Generative Model. We assume the data is generated as follows. A history h (i.e. a partial trajectory) is sampled from an arbitrary data-generating distribution. Sampling a particular state representation $\phi(s)$ from the belief distribution given h is controlled by the latent variable z, independently sampled from a prior distribution (e.g. multivariate standard Gaussian). Since a history h may correspond to multiple possible states s, the state representation $\phi(s)$ is a stochastic function of h, z, hence

$$p(\phi(s), h, z) = p(h)p(z)p(\phi(s)|h, z)$$

Variational Objective. We define an encoder network $q_\xi(z|\phi(s), h)$ parameterized by ξ and a decoder network $p_\theta(\phi(s)|h, z)$ parameterized by θ.

Using the evidence lower bound (Kingma & Welling, 2014),

$$\begin{aligned}
&\log p_\theta(\phi(s), h) \\
&\geq \mathbb{E}_{q_\xi(z|\phi(s),h)}\big[\log p_\theta(\phi(s), h, z) - \log q_\xi(z|\phi(s), h)\big] \\
&= \mathbb{E}_{q_\xi(z|\phi(s),h)}\big[\log p(h) + \log p(z) + \\
&\qquad \log p_\theta(\phi(s)|h, z) - \log q_\xi(z|\phi(s), h)\big]
\end{aligned}$$

We can safely drop the constant $\log p(h)$ term from the objective, which does not contribute to the gradients with respect to our model parameters θ, ξ. Thus, our optimization objective is

$$\begin{aligned}
&\mathcal{L}^{\text{VAE}}_{\theta,\xi}(\phi(s), h) \\
&= \mathbb{E}_{q_\xi(z|\phi(s),h)}\big[\log p(z) + \log p_\theta(\phi(s)|h, z) - \\
&\qquad \log q_\xi(z|\phi(s), h)\big]
\end{aligned}$$

Training Details. Following Kingma & Welling 2014, we draw samples $(s, h) \sim \mathcal{D}$ and $z \sim q_\xi(z|\phi(s), h)$ (using the reparameterization trick) to estimate the objective $\mathcal{L}^{\text{VAE}}_{\theta,\xi}$. We use a shared recurrent network to encode the history h between the encoder and decoder. We represent q_ξ, p_θ as Gaussian distributions with learned means and standard deviations.

Belief State Inference. At deployment time, we are interested in using a trained VAE to approximate the belief distribution over state encodings $\phi(s)$ given the history h. Unfortunately, this belief distribution $p_\theta(\phi(s)|h)$ can be rather complex and we opt to represent it approximately via a collection of n i.i.d. samples $\hat{u}_1, \ldots, \hat{u}_n \sim p_\theta(\phi(s)|h)$. These samples are obtained by ancestral sampling from the generative model of the VAE: we first sample $z_i \sim p(z)$ and then $\hat{u}_i \sim p_\theta(\phi(s)|h, z_i)$ using the VAE decoder network. We then represent the belief over state encodings given the history as $b(h) = (\hat{u}_1, \ldots, \hat{u}_n)$.

4.3. Policy Learning with Deep Belief Representations

Here, we describe the policy architecture for our approach. We train a policy of the form $\pi(a_t|o_t, b(h_t))$, which does not rely on ground-truth states for choosing actions. For the purposes of training π, we treat $b(h_t)$ as a fixed part of the environment state, rather than a learnable function. Particularly, note that any dependence of the policy on the history is captured through $b(h_t)$. Updates to the policy π can then be made using any standard, *Markovian* deep RL algorithm. We can optionally continue training the belief state VAE with on-policy data as the distribution of states and histories shifts during training[1]. While this may

[1] In our experiments, we fine-tune the belief state VAE using on-policy data.

introduce non-stationarity when training the policy, we did not observe this to be an issue in our experiments.

We use the following architecture when encoding $b(h_t)$ within the policy. Exploiting the fact that the belief $b(h_t) = (\hat{u}_{t,1}, \ldots, \hat{u}_{t,n})$ is represented by an order-invariant collection of samples, we encode $b(h_t)$ using a function

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

where $W_{\text{enc}}, W_{\text{agg}}$ are neural networks. This is similar to prior works on encoding sets (Zaheer et al., 2017).

4.4. Comparison with Traditional Recurrent Policies

While they may appear similar in architecture, an important difference between our approach and traditional methods like DRQN (Hausknecht & Stone, 2015) is in the manner in which the recurrent model is trained. Our approach decomposes the RL problem by training a recurrent network to predict a belief over states, using ground-truth states s_t as a learning signal. This objective is *separate* from the policy objective. On the other hand, DRQN trains a recurrent policy end-to-end with a single RL objective, using reward as the only learning signal. Prior work has also proposed predicting future observations and rewards as a learning signal for the recurrent network (Lee et al., 2020; Chen et al., 2022; Gregor et al., 2019a; Han et al., 2020; Gregor et al., 2019b; Hafner et al., 2019).

Unfortunately, these other objectives might provide a weak learning signal on tasks requiring long-term memory. In the *Sphinx* example, the sphinx's advice improves the predictability of rewards from opening a box (occurring several steps into the future), but not of immediate rewards or observations. In contrast, the sphinx's advice provides an immediate boost in the predictability of the current state (and hence, the box containing the reward), thus, we posit that state prediction may provide a stronger learning signal.

5. Experiments

We conduct a series of experiments to demonstrate the efficacy of Believer in partially observable domains. In particular, we focus on tasks involving long-term memory, information seeking, and/or state inference. Our hypothesis is that Believer has an advantage over existing approaches when faced with one or more of the following problems: **(1)** The *exploration-exploitation dilemma* arising when acquiring information is costly. **(2)** The need to remember information over long periods of time. **(3)** High-dimensional states and observations containing redundant and/or task-irrelevant information. **(4)** The need for probabilistic inference of an uncertain state due to noisy information.

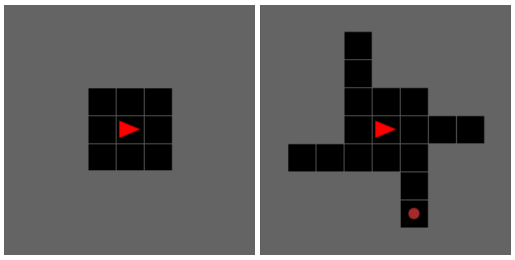

Figure 3. Visualization of the Cookie task. **Left**: The agent's *observation*, represented as a $9 \times 9 \times 3$ image. The agent's view is limited to its current room, and the cookie is currently unobservable to the agent. **Right**: The corresponding *state* (unobservable to the agent at deployment time), represented as a $9 \times 9 \times 3$ image. The cookie is in the south corridor.

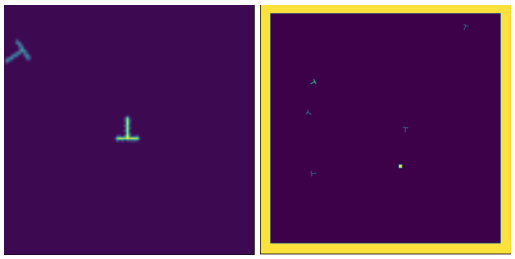

Figure 4. Visualization of the Escape Room task. **Left:** The agent's observation, represented as a 100×100 greyscale image. The statue at the top left indicates that the portal is towards the agent's bottom left. **Right:** Visualization of the full map, showing the agent (green), statues (blue), and the portal (the yellow square). States are represented as vectors containing the positions and orientations of all objects.

5.1. Experimental Setup

For each evaluation environment, we collect a small amount of offline data from a random-action policy. We use this dataset in Believer to learn state representations (Section 4.1) and to pretrain the belief state VAE (Section 4.2). Several other baselines also make use of the offline dataset and we adjust all learning curves to account for this additional data.

To ensure a fair comparison, we train all policies using PPO and choose network architectures to be as similar as possible between methods. We generally observed that larger minibatch sizes led to improved policy stability and final performance, thus, we equalize the minibatch size across methods to a large value (representing a limit on GPU memory). For additional implementation details and a full description of network architectures and hyperparameters, please see Appendix B and C.

Code available at https://github.com/awwang10/sphinx.

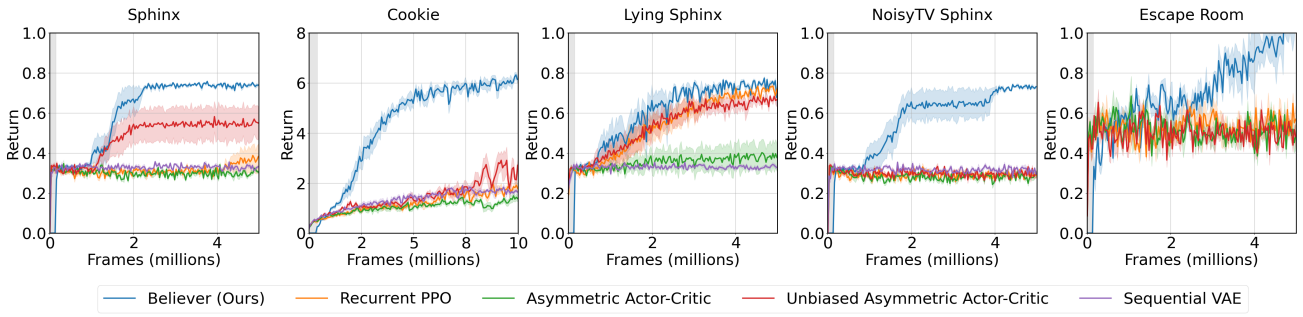

Figure 5. We evaluate Believer (ours) against common approaches for partially observable RL in five environments. The figure shows return per episode plotted over the course of training (measured in total training frames), averaged over five seeds, with shaded bars showing standard error and the shaded grey area representing the pretraining dataset size for applicable methods. Believer outperforms other methods on all domains.

5.2. Environments

Sphinx. We consider the *Sphinx* task described in Section 2.3 in a MiniGrid (Chevalier-Boisvert et al., 2018) environment with observations and states represented as $8 \times 8 \times 3$ images. The state always reveals the box containing the reward to the agent.

Cookie. We adapt the *Cookie* task from Toro Icarte et al. (2019) into a MiniGrid environment. The agent begins in a 3×3 atrium with a hidden corridor leading out of each of the four walls. The north corridor leads to a button that, when pressed, spawns a cookie (if one does not already exist) at the end of one of the other three corridors at random. Eating the cookie gives a reward of +1 and resets the button. States are $9 \times 9 \times 3$ images revealing the entire map while observations only show the agent's current room (see Figure 3). The agent's goal is to eat as many cookies as possible. This requires *exploration* to find cookies and the hidden entrances to the corridors, and *long-term memory* of the state of the button, the corridor entrance locations, and which corridors were recently checked for the cookie.

Lying Sphinx. In this variant of *Sphinx*, speaking to the sphinx yields a random answer 50% of the time and costs -0.05 reward. This allows the agent to improve its inference of the correct box by speaking to the sphinx multiple times.

NoisyTV Sphinx. We modify *Sphinx* so that the outer border of the map flashes random colours at each timestep, both in the state and the observation. While these colours are unrelated to environment rewards, learning an environment model over raw states or observations becomes significantly harder.

Escape Room *Escape Room* is a continuous-state, continuous-action environment where the agent's goal is to locate and enter a portal in a large, hazy room (Figure 4). Unfortunately, the agent only observes a limited view of its nearby surroundings, represented as an 100×100 grayscale image, but several statues placed around the map point towards the portal to help guide the agent. The agent should explore to find the statues and portal and remember the directions of statues, once found. States are represented as vectors containing the positions and orientations of all objects.

For additional environment details, please see Appendix A.

5.3. Baselines

We compare Believer against the following baselines. Asymmetric Actor-Critic (Pinto et al., 2018) and Unbiased Asymmetric Actor-Critic (Baisero & Amato, 2022) are model-free approaches that also exploit state information during training (but not deployment). We also compare against a recurrent version of PPO, representing model-free approaches that do not exploit state information (Hausknecht & Stone, 2015; Zhu et al., 2017; Heess et al., 2015). Lastly, we consider a sequential VAE that learns a latent state representation by predicting the sequence of future observations and rewards given actions, with similar motivations to, e.g., Lee et al. (2020), Hafner et al. (2019), Chen et al. (2022), Chung et al. (2015), and Han et al. (2020).

5.4. Main Results

We report the performance of each method, averaged over 5 seeds, in Figure 5. Believer learns a strong solution in each domain, often outperforming the other baselines by a significant margin. In Sphinx and Cookie, the only other method to make progress beyond a trivial solution (e.g. opening a box at random in Sphinx, or checking corridors randomly in Cookie) is Unbiased Asymmetric Actor-Critic, which also incorporates state information. In Escape Room, a more difficult environment with high-dimensional image observations, none of the other methods consistently reach the portal. Believer also effectively infers states under noisy

We drop this baseline for Escape Room due to the computational challenge of reconstructing high-dimensional images.

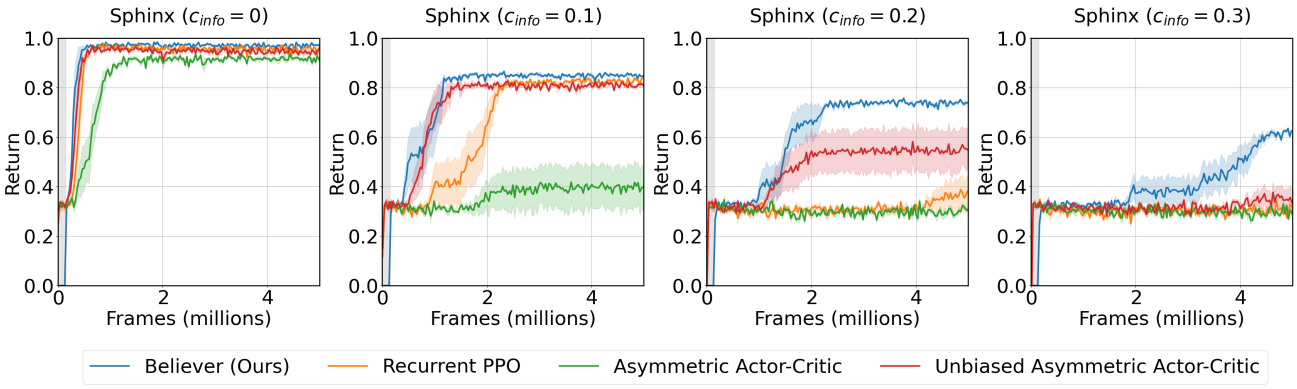

Figure 6. We investigate the impact of varying the cost of speaking to the sphinx $c_{\text{info}} \in \{0, 0.1, 0.2, 0.3\}$ in the Sphinx task on various methods. The figure shows return per episode plotted over the course of training (measured in total training frames), averaged over five seeds, with shaded bars showing standard error and the shaded grey area representing the pretraining dataset size for applicable methods. Most approaches learn to seek information when c_{info} is low, but for higher c_{info}, only Believer is able to succeed.

information in Lying Sphinx and is robust to the irrelevant distractors in states and observations in NoisyTV Sphinx.

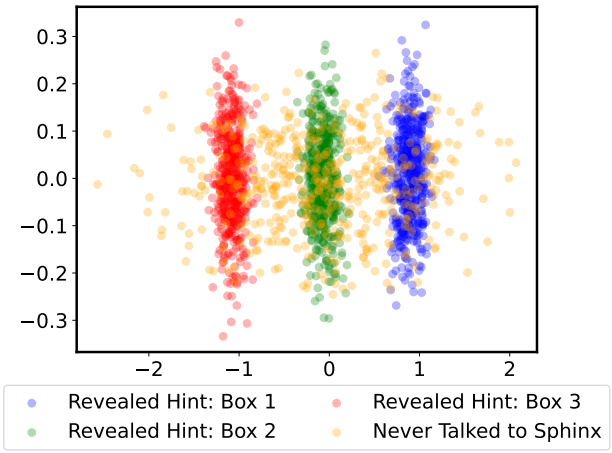

Figure 7. A visualization of various belief states learned in the Sphinx task. Each colour represents a belief state arising from a trajectory that reveals a particular type of information about the correct box. **Blue, green, and red**: agent talked to the sphinx, revealing the correct box to be 1, 2, and 3, respectively. **Orange:** agent never talked to the sphinx. Observe that the VAE predicts a unique cluster when the correct box is known, and assigns probability to all clusters when lacking information.

To visualize the learned beliefs, we retrain Believer on Sphinx with two-dimensional state encodings $\phi(s)$. We then sample trajectories with the same final agent position but that reveal different information about the correct box (Figure 7). We find that distinct clusters emerge for each possible box containing the reward and that the learned beliefs can capture both knowledge and uncertainty of this information.

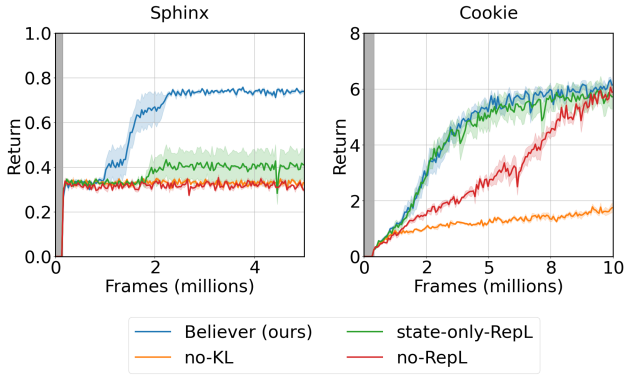

Figure 8. We conduct ablations to verify the design choices in Believer. The figure shows return per episode plotted over the course of training (measured in total training frames), averaged over five seeds, with shaded bars showing standard error and the shaded grey area representing the pretraining dataset size. Representation learning and applying an information bottleneck are critical for performance and/or sample efficiency. Discarding observable features from the state encoding is critical in Sphinx.

5.5. Why is Seeking Information Hard?

Next, we aim to better understand why tasks requiring information gathering can be particularly challenging for RL agents. Liu et al. (2021) attribute this to an exploration-exploitation dilemma: an agent that spends time or resources to acquire information (e.g. by speaking to the sphinx) but that does not yet know how to *use* that information to garner reward will be disincentivized from continuing to acquire such information. We evaluate two hypotheses. First, we predict that the cost of acquiring information significantly impacts exploration, increasing the likelihood of suboptimal behaviours that do not actively seek out information. Second, we predict that methods that leverage state information during training are less susceptible to this effect than methods that only maximize a reward signal. Our key insight

supporting the second hypothesis is that the auxiliary task of predicting states provides a useful training signal; acquiring task-relevant information (e.g. speaking to the sphinx) leads to an immediate improvement in state prediction. On the other hand, it can be challenging to discern the utility of information from rewards that are sparse, delayed, or require specific action (e.g. opening the correct box).

To test this, we evaluate several approaches on the Sphinx environment while varying the cost c_{info} of speaking to the sphinx. We report the results in Figure 6. For $c_{\text{info}} = 0$, all methods learn a near-optimal solution. However, as the cost of speaking to the sphinx increases, learning the optimal behaviour becomes more challenging. At $c_{\text{info}} = 0.2$, Believer learns a near-optimal solution while Unbiased Asymmetric Actor-Critic (which trains using state information) is able to make some progress. At $c_{\text{info}} = 0.3$, only Believer learns to speak to the sphinx to select the correct box. This suggests that for tasks requiring active information acquisition, existing partially observable RL methods are prone to falling into local optima, particularly when obtaining information is costly. We believe that training with state information is a key reason for the success of our approach.

5.6. Ablation Studies

To verify our design choices, we conduct ablation experiments for Believer on the Sphinx and Cookie domains. We investigate the following: **(Q1)** Is representation learning important? **(Q2)** Is it important for state encodings $\phi(s_t)$ to *discard* duplicate information from the observation o_t? **(Q3)** Is the information bottleneck on $\phi(s_t)$ important?

To answer (Q1), we evaluate a baseline that forgoes representation learning and instead models a belief over the current state as an $8 \times 8 \times 3$ image **(no-RepL)**. To answer (Q2), we train state encodings $\phi(s_t)$ that predict future rewards and state encodings, but without considering observations **(state-only-RepL)**. To answer (Q3), we train our approach with the KL-loss coefficient λ_{KL} set to 0 **(no-KL)**. The results are reported in Figure 8.

(Q1) Representation learning is critical to the success of Believer in *Sphinx* and significantly improves sample efficiency in *Cookie* vs modelling states as raw images. We note that in both environments, most pixels of the state can be perfectly reconstructed without considering the most critical state information (e.g. the correct box, or the location of a cookie).

(Q2) Training state encodings $\phi(s_t)$ that discard duplicate information from the observation o_t is critical in *Sphinx* but not in *Cookie*. An important remark is that while *Sphinx* states contain many pieces of information (e.g. agent position, direction, the correct box), a sufficient encoding $\phi(s_t)$ *only* needs to encode the correct box. Thus, it is possible to learn a very compact representation $\phi(s_t)$. This is not the case in *Cookie*, where $\phi(s_t)$ should encode the state of the button, the locations of cookies, the locations of corridor entrances, and which corridors have been checked for the cookie.

(Q3) The information bottleneck on $\phi(s_t)$ is critical in both environments. Intuitively, the information bottleneck drives the state encoding $\phi(s_t)$ to be as compact as possible, relegating duplicate information between states and observations to the observation encoding $\psi(o_t)$.

6. Conclusion

This paper introduces a novel deep RL approach for partially observable problems, motivated by the plethora of applications where full states are observable during training but the RL agent's observation may be limited when deployed. Our approach, Believer, is based on traditional belief state approaches in POMDPs and is scalable to systems with high-dimensional states and observations. We highlight a number of advantages of training with state information, and demonstrate the effectiveness of our approach on challenging, partially observable domains involving long-term memory, active information seeking, and probabilistic inference. Social implications of this work are reflected upon in Appendix D.

Acknowledgements

We gratefully acknowledge funding from the Natural Sciences and Engineering Research Council of Canada (NSERC), the Canada CIFAR AI Chairs Program, and Microsoft Research. Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute for Artificial Intelligence. We thank the Schwartz Reisman Institute for Technology and Society for providing a rich multi-disciplinary research environment. The fourth author would like to acknowledge funding from (1) the National Center for Artificial Intelligence CENIA FB210017 (Basal ANID) and (2) a Fondecyt grant 11230762.

References

Alemi, A. A., Fischer, I., Dillon, J. V., and Murphy, K. Deep variational information bottleneck. In *5th International Conference on Learning Representations*, 2017.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P. F., Schulman, J., and Mané, D. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016. doi: 10.48550/arXiv.1606.06565.

Åström, K. J. Optimal control of Markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, 10(1):174–205, 1965.

Baisero, A. and Amato, C. Unbiased asymmetric reinforcement learning under partial observability. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pp. 44–52, 2022.

Cassandra, A., Littman, M. L., and Zhang, N. L. Incremental pruning: a simple, fast, exact method for partially observable markov decision processes. In *Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence*, pp. 54–61, 1997.

Chen, D., Zhou, B., Koltun, V., and Krähenbühl, P. Learning by cheating. In *Conference on Robot Learning*, pp. 66–75. PMLR, 2020.

Chen, X., Mu, Y. M., Luo, P., Li, S., and Chen, J. Flow-based recurrent belief state learning for POMDPs. In *International Conference on Machine Learning*, pp. 3444–3468. PMLR, 2022.

Chevalier-Boisvert, M., Willems, L., and Pal, S. Minimalistic gridworld environment for gymnasium, 2018. URL https://github.com/Farama-Foundation/Minigrid.

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. C., and Bengio, Y. A recurrent latent variable model for sequential data. In *Advances in Neural Information Processing Systems 28*, 2015.

Dulac-Arnold, G., Levine, N., Mankowitz, D. J., Li, J., Paduraru, C., Gowal, S., and Hester, T. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110(9):2419–2468, 2021. doi: 10.1007/s10994-021-05961-4.

Gelada, C., Kumar, S., Buckman, J., Nachum, O., and Bellemare, M. G. Deepmdp: Learning continuous latent space models for representation learning. In *International Conference on Machine Learning*, pp. 2170–2179. PMLR, 2019.

Gregor, K., Jimenez Rezende, D., Besse, F., Wu, Y., Merzic, H., and van den Oord, A. Shaping belief states with generative environment models for RL. In *Advances in Neural Information Processing Systems 32*, 2019a.

Gregor, K., Papamakarios, G., Besse, F., Buesing, L., and Weber, T. Temporal difference variational auto-encoder. In *International Conference on Learning Representations*, 2019b. URL https://openreview.net/forum?id=S1x4ghC9tQ.

Guo, Z. D., Azar, M. G., Piot, B., Pires, B. A., and Munos, R. Neural predictive belief representations. *arXiv preprint arXiv:1811.06407*, 2018.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pp. 2555–2565. PMLR, 2019.

Han, D., Doya, K., and Tani, J. Variational recurrent models for solving partially observable control tasks. In *8th International Conference on Learning Representations*, 2020.

Hausknecht, M. and Stone, P. Deep recurrent q-learning for partially observable MDPs. In *2015 AAAI fall symposium series*, 2015.

Heess, N., Hunt, J. J., Lillicrap, T. P., and Silver, D. Memory-based control with recurrent neural networks. *arXiv preprint arXiv:1512.04455*, 2015.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Humplik, J., Galashov, A., Hasenclever, L., Ortega, P. A., Teh, Y. W., and Heess, N. Meta reinforcement learning as task inference. *arXiv preprint arXiv:1905.06424*, 2019.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

Katt, S., Oliehoek, F. A., and Amato, C. Learning in POMDPs with Monte Carlo tree search. In *International Conference on Machine Learning*, pp. 1819–1827. PMLR, 2017.

Kemertas, M. and Aumentado-Armstrong, T. Towards robust bisimulation metric learning. In *Advances in Neural Information Processing Systems 34*, pp. 4764–4777, 2021.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In Bengio, Y. and LeCun, Y. (eds.), *2nd International Conference on Learning Representations*, 2014.

Kumar, A., Fu, Z., Pathak, D., and Malik, J. RMA: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021.

Lauri, M. and Ritala, R. Planning for robotic exploration based on forward simulation. *Robotics and Autonomous Systems*, 83:15–31, 2016.

Lauri, M., Hsu, D., and Pajarinen, J. Partially observable markov decision processes in robotics: A survey. *IEEE Transactions on Robotics*, 2022.

Lee, A. X., Nagabandi, A., Abbeel, P., and Levine, S. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. In *Advances in Neural Information Processing Systems 33*, pp. 741–752, 2020.

Li, A. C., Chen, Z., Vaezipoor, P., Klassen, T. Q., Icarte, R. T., and McIlraith, S. A. Noisy symbolic abstractions for deep rl: A case study with reward machines. In *Deep Reinforcement Learning Workshop NeurIPS 2022*.

Littman, M. L. The witness algorithm: Solving partially observable Markov decision processes. *Brown University, Providence, RI*, 1994.

Liu, E. Z., Raghunathan, A., Liang, P., and Finn, C. Decoupling exploration and exploitation for meta-reinforcement learning without sacrifices. In *International Conference on Machine Learning*, pp. 6925–6935. PMLR, 2021.

Meng, L., Gorbet, R., and Kulić, D. Memory-based deep reinforcement learning for POMDPs. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5619–5626. IEEE, 2021.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. A. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. URL http://arxiv.org/abs/1312.5602.

Monahan, G. E. State of the art—a survey of partially observable Markov decision processes: theory, models, and algorithms. *Management science*, 28(1):1–16, 1982.

Pinto, L., Andrychowicz, M., Welinder, P., Zaremba, W., and Abbeel, P. Asymmetric actor critic for image-based robot learning. In *14th Robotics: Science and Systems, RSS 2018*. MIT Press Journals, 2018.

Pleines, M., Pallasch, M., Zimmer, F., and Preuss, M. Memory gym: Partially observable challenges to memory-based agents. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=jHc8dCx6DDr.

Sharma, P. K., Fernandez, R., Zaroukian, E., Dorothy, M., Basak, A., and Asher, D. E. Survey of recent multi-agent reinforcement learning algorithms utilizing centralized training. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications III*, volume 11746, pp. 665–676. SPIE, 2021.

Silver, D. and Veness, J. Monte-Carlo planning in large POMDPs. In *Advances in Neural Information Processing Systems 23*, 2010.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, pp. 387–395. PMLR, 2014.

Spaan, M. T., Veiga, T. S., and Lima, P. U. Active cooperative perception in network robot systems using POMDPs. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4800–4805. IEEE, 2010.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Thrun, S. Monte Carlo POMDPSs. In *Advances in Neural Information Processing Systems 12*, 1999.

Tishby, N., Pereira, F. C., and Bialek, W. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

Toro Icarte, R., Waldie, E., Klassen, T. Q., Valenzano, R., Castro, M. P., and McIlraith, S. A. Learning reward machines for partially observable reinforcement learning. In *Advances in Neural Information Processing Systems 32*, pp. 15497–15508, 2019.

Toro Icarte, R., Valenzano, R., Klassen, T. Q., Christoffersen, P., Farahmand, A.-m., and McIlraith, S. A. The act of remembering: A study in partially observable reinforcement learning. *arXiv preprint arXiv:2010.01753*, 2020.

Tuli, M., Li, A., Vaezipoor, P., Klassen, T., Sanner, S., and McIlraith, S. Learning to follow instructions in text-based games. *Advances in Neural Information Processing Systems*, 35:19441–19455, 2022.

Vaezipoor, P., Li, A. C., Toro Icarte, R. A., and McIlraith, S. A. LTL2Action: Generalizing LTL instructions for multi-task RL. In *International Conference on Machine Learning*, pp. 10497–10508. PMLR, 2021.

Wang, Y. and Tan, X. Deep recurrent belief propagation network for POMDPs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 10236–10244, 2021.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. Deep sets. In *Advances in Neural Information Processing Systems 30*, 2017.

Zhang, A., McAllister, R. T., Calandra, R., Gal, Y., and Levine, S. Learning invariant representations for reinforcement learning without reconstruction. In *9th International Conference on Learning Representations*, 2021.

Zhang, M., McCarthy, Z., Finn, C., Levine, S., and Abbeel, P. Learning deep neural network policies with continuous memory states. In *2016 IEEE international conference*

on robotics and automation (ICRA), pp. 520–527. IEEE, 2016.

Zhu, P., Li, X., and Poupart, P. On improving deep reinforcement learning for pomdps. *CoRR*, abs/1704.07978, 2017. URL http://arxiv.org/abs/1704.07978.

A. Environment Descriptions

A.1. Sphinx Environment

The Sphinx environment is a discrete-state, discrete-action finite horizon task in which an agent, randomly placed in a 8×8 grid, has an objective to go to one particular box out of three possible boxes, located in the top-right, bottom-right and bottom-left corners, unknown to the agent at the start of each episode, creating partial observability conditions. Each episode terminates after 100 steps. Stepping onto the tile with the correct box yields a reward of +1 and terminates the episode. Stepping onto a tile with an incorrect box yields no reward and also terminates the episode. In the top-left corner of the grid, there is a sphinx. If the agent enters a tile adjacent to the sphinx, the sphinx emits the correct box number for one step. In Sphinx (0.1), Sphinx (0.2), Sphinx (0.3), moving adjacent to the sphinx incurs a negative reward of -0.1, -0.2, and -0.3 respectively. Larger negative rewards for consulting the sphinx increase the difficulty of the task. The optimal fully observable policy is to directly head to the correct box. The optimal partially observable policy is to head to the sphinx, observe the hint, and proceed to the correct box.

The agent receives a $8 \times 8 \times 3$ dimensional image of the environment as an input observation. The first channel of the image is an 8×8 image containing information about the obstacles and items on the grid such as walls, sphinx location, the three boxes, each encoded by an integer. The second channel of the image is a 8×8 array of zeros, except at the coordinates of the sphinx, whose value is the current hint being emitted by the sphinx. When the agent is not adjacent to the sphinx, the value emitted is 3. Otherwise, the sphinx is emitting 0, 1, 2, corresponding to the top-right, bottom-right, and bottom-left tiles. The final channel of the image is a 8×8 array of zeros, except at the agent's current coordinates, where the possible values are 1, 2, 3 or 4 corresponding to the direction in which the agent is facing (east, south, west, north, respectively). The ground-truth state is encoded exactly the same as the agent's observations, except that in the first channel, the correct box is encoded with a separate value. The action space of the environment is $\{ 0, 1, 2 \}$ corresponding to actions left, right and forward in the environment.

A visualization of the environment is provided in Figure 1.

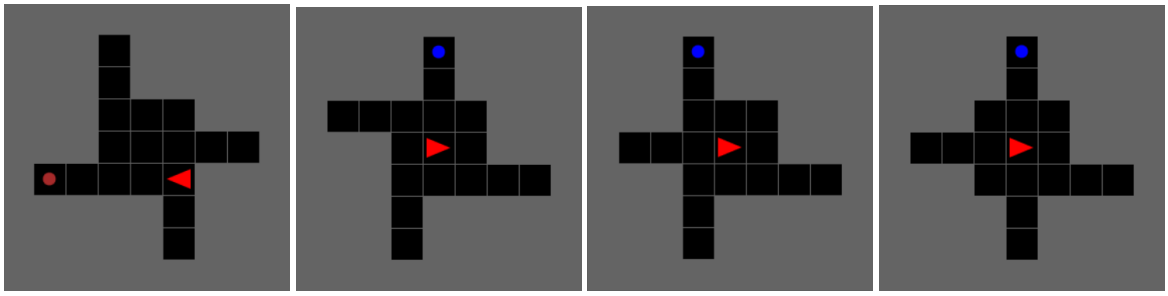

Figure 9. Examples of corridor configurations for the Cookie domain. The agent is denoted by a red triangle. The button that generates cookies is denoted by solid blue circle. The cookie is denoted by a solid brown circle.

A.2. Cookie Environment

The Cookie environment is a discrete-state, discrete-action finite horizon task in which an agent, placed in the center of a 9×9 grid containing a central room and four corridors, must repeatedly press a button in the north corridor and eat a cookie in one of the other three corridors at random. The button in the north corridor only appears if no cookie is present on the grid, and disappears once pressed, spawning a cookie in one of the other three corridors. Eating a cookie provides +1 reward. The precise attachment position of each of the north, east, south, and west corridors to the room is generated randomly at the start of the episode (see Figure 9). When in a corridor, the agent cannot observe the contents of the central room nor the contents in the other corridors. When in the central room, the agent cannot observe the contents of the corridors. This creates partially observable conditions. The optimal fully observable policy is to repeat the following actions: 1) press the button in the north corridor and 2) immediately proceed to the corridor with the cookie and eat it. The optimal partially observable policy is to repeat the following actions: 1) press the button in the north corridor and 2) check each corridor one by one, and if a cookie is observed, eat it.

The agent receives a $9 \times 9 \times 3$ dimensional image of the environment as an input observation. The first channel of the image is a 9×9 image containing information about the obstacles and items on the grid such as walls, cookies and buttons,

each encoded by an integer. When in the main room, the other areas of the grid are encoded by zeros, creating partial observability. Since the attachment points of the corridors are random and the agent cannot see the corridors when in the central room, at the start of each episode, the agent must also learn where the corridors are located and remember this information throughout the episode. The second channel of the image is always an array of zeros. The final channel of the image is a 9×9 image of zeros, except at the agent's current coordinates, where the possible values are 1, 2, 3 or 4 corresponding to the direction in which the agent is facing (east, south, west, north, respectively). The action space of the environment is $\{$ 0, 1, 2 $\}$ corresponding to actions left, right and forward in the environment.

A.3. Escape Room Environment

Escape Room is a continuous-state, continuous-action episodic environment, in which an agent must move to a portal located in a closed 400×400 square room. The room contains four statues, which point in the direction of the portal. The agent receives a +1 sparse reward for reaching the portal, as well as a small dense reward based on the change in distance to the portal from the previous timestep. Note that the reward is unobservable at deployment time (and thus cannot be exploited by the agent to infer the portal location). The agent only observes a small square area of size 100×100 centered around the agent, which creates partial observability conditions. This environment requires information-seeking behaviour (such as finding a statue) and long-term memory (such as remembering the direction in which the statue is pointing). The optimal partially observable policy is to explore and find either the portal (in which case the agent should head directly to the portal), or a statue and move in the direction the statue is pointing in to find the portal. The state space is a 17-dimensional real-valued vector containing the position and direction of the agent and statues, as well as the position of the portal. The observation space is a $1 \times 100 \times 100$ grayscale image centered around the agent, always oriented such that the agent faces up in the image. The episode ends after 200 steps or when the agent reaches the portal. The agent only needs to reach a 20 unit distance to the centroid of the portal to receive the +1 reward.

B. Model Architectures

B.1. Representation Learning

For all Sphinx environments, the *embedding size* is $8 \times 8 \times 3 = 192$.
For the Cookie environment, the *embedding size* is $9 \times 9 \times 3 = 243$.
For the Escape Room, we take image observations and states, and pass them through a convolutional neural network with the following architecture: [Conv2d(1, 8, 9, 4), ReLu, Conv2d(8, 16, 5, 2), Relu, Conv2d(16, 32, 3, 2), Relu, Linear(512, 192)]. The parameters of Conv2d are input channels, output channels, kernel size, and stride respectively. The *embedding size* is 192.

State Encoder Given the flattened state s, the State Encoder outputs a stochastic encoding of the state $\phi(s)$, modelled by a Normal distribution.
Architecture: [*embedding size*, 128, 128, $2 \times 16 = 32$] fully connected layers with ReLU activations.

Observation Encoder Given the flattened observations o, the Observation Encoder outputs a stochastic encoding of the observation $\psi(o)$, modelled by a Normal distribution.
Architecture: [*embedding size*, 128, 128, $2 \times 16 = 32$] fully connected layers with ReLU activations.

Dynamics Model Given an action a and stochastic encoding of the state $u_s \sim \phi(s)$ and observation $u_o \sim \psi(o)$, the dynamics model predicts the reward, next state encoding, and next observation encoding $\hat{r}, \hat{u}_{s'}, \hat{u}_{o'} = g(u_s, u_o, a)$.
Dynamics Model Architecture: [$2 \times 16 + 1 = 33$, 128, 128] fully connected layers with ReLU activations, followed by one of the following output layers:
Reward: [128, 1] linear layer
Next Observation Encoding: [128, 16] linear layer
Next State Encoding: [128, 16] linear layer

B.2. Belief VAE

VAE latent dimensions is 32 for all Sphinx environments, 64 for the Cookie environment, and 32 for the Escape Room environment.

VAE Encoder Given a representation of the state $\phi(s)$, and a history context h, the VAE encoder $q_\xi(z|\phi(s), h)$ outputs

a stochastic encoding of latent variable z, modelled by a Normal distribution.
VAE Encoder Architecture: [$16 + 256 = 272$, 256, 256, $2\times$*VAE latent dimensions*] fully connected layers with ReLU activations.

VAE Decoder Given a history context h, and a latent variable z, the VAE decoder $p_\theta(\phi(s)|h, z)$ outputs a stochastic encoding of the representation of the state $\phi(s)$, modelled by a Normal distribution.
VAE Decoder Architecture: [*VAE latent dimensions* $+256$, 256, 256, $2 \times 16 = 32$] fully connected layers with ReLU activations.

History Model
The history model encodes a sequence of observations. An observation is first fed through an [*embedding size*, 512] linear layer. The 512 dimensional output is fed into a three-layer stacked GRU each with hidden state size of 256. We add the next hidden state outputs of the first two GRUs, and concatenate with the 512 dimensional output, and feed the result into a [256+512, 256, 256] fully connected network with ReLU activations.

B.3. Policy Training Model Architecture

Each observation is first passed through the image encoder.

For the Escape Room, we take image observations and states, and pass them through a convolutional neural network with the following architecture: [Conv2d(1, 8, 9, 4), ReLu, Conv2d(8, 16, 5, 2), Relu, Conv2d(16, 32, 3, 2), Relu, Linear(512, 64)]. The parameters of Conv2d are input channels, output channels, kernel size, and stride respectively.

Image encoder: [64, 64, 64] fully connected layers with ReLU activations.

Belief Encoder Given a set of belief samples, the belief encoder W_{enc} encodes each 16 dimensional belief in the set.
Architecture: [16, 64, 64] fully connected layers with ReLU activations

Belief Aggregator The belief aggregator W_{agg} encodes an average of encoded beliefs $\frac{1}{n}\sum_{i=1}^{n} W_{\text{enc}}(\hat{u}_{t,i})$
Architecture: [64, 64, 64] fully connected layers with ReLU activations

The output of the image encoder, and the output of the belief aggregator are concatenated, and passed to the actor and critic.
Actor [128, 64, 64, 3] fully connected layers with ReLU activations.
Critic [128, 64, 64, 1] fully connected layers with ReLU activations.

C. Hyperparameters

Table 1. Sphinx Environment Training Hyperparameters

	Our Method	Recurrent PPO	Unbiased Asymmetric Actor Critic
Discount factor (γ)	0.99	0.99	0.99
Pretrain episodes	3000	–	–
Pretrain Representation Model			
Number of Epochs	1000	–	–
Batch Size	500	–	–
KL Coefficient	0.3	–	–
State Prediction Loss Coefficient	0.3	–	–
Observation Prediction Loss Coefficient	0.03	–	–
Reward Prediction Loss Coefficient	10	–	–
Latent Dimensions	16	–	–
Pretrain VAE			
Learning Rate	0.0003	–	–
Number of Epochs	5000	–	–
Latent Dimensions	32	–	–
Policy Learning Hyperparameters			
Learning Rate (Policy)	0.0005	0.0005	0.0005
Learning Rate (VAE)	0.0003	–	–
Minibatch Size	2048	2048	2048
Number of Epochs (PPO)	24	8	8
Entropy coefficient	0.03	0.03	0.03
Recurrence	1	16	16
Number of Frames Per Update	8192	8192	8192

Table 2. Cookie Environment Training Hyperparameters

	Our Method	Recurrent PPO	Unbiased Asymmetric Actor Critic
Discount factor (γ)	0.97	0.97	0.97
Pretrain episodes	1000	–	–
Pretrain Representation Model			
Number of Epochs	100	–	–
Batch Size	500	–	–
KL Coefficient	0.03	–	–
State Prediction Loss Coefficient	0.1	–	–
Observation Prediction Loss Coefficient	0.1	–	–
Reward Prediction Loss Coefficient	300	–	–
Latent Dimensions	16	–	–
Pretrain VAE			
Learning Rate	0.0003	–	–
Number of Epochs	3000	–	–
Latent Dimensions	64	–	–
Policy Learning Hyperparameters			
Learning Rate (Policy)	0.001	0.001	0.001
Learning Rate (VAE)	0.001	–	–
Minibatch Size	4096	4096	4096
Number of Epochs (PPO)	8	8	8
Entropy coefficient	0.003	0.003	0.003
Recurrence	1	16	16
Number of Frames Per Update	16384	16384	16384

Table 3. Lying-Sphinx Training Hyperparameters

	Our Method	**Recurrent PPO**	**Unbiased Asymmetric Actor Critic**
Discount factor (γ)	0.99	0.99	0.99
Pretrain episodes	3000	–	–
Pretrain Representation Model			
Number of Epochs	1000	–	–
Batch Size	500	–	–
KL Coefficient	0.3	–	–
State Prediction Loss Coefficient	0.3	–	–
Observation Prediction Loss Coefficient	0.03	–	–
Reward Prediction Loss Coefficient	10	–	–
Latent Dimensions	16	–	–
Pretrain VAE			
Learning Rate	0.0003	–	–
Number of Epochs	5000	–	–
Latent Dimensions	32	–	–
Policy Learning Hyperparameters			
Learning Rate (Policy)	0.0005	0.0005	0.0005
Learning Rate (VAE)	0.0003	–	–
Minibatch Size	2048	2048	2048
Number of Epochs (PPO)	24	8	8
Entropy coefficient	0.03	0.03	0.03
Recurrence	1	16	16
Number of Frames Per Update	8192	8192	8192

Table 4. Noisy-TV Sphinx Training Hyperparameters

	Our Method	**Recurrent PPO**	**Unbiased Asymmetric Actor Critic**
Discount factor (γ)	0.99	0.99	0.99
Pretrain episodes	3000	–	–
Pretrain Representation Model			
Number of Epochs	1000	–	–
Batch Size	500	–	–
KL Coefficient	0.3	–	–
State Prediction Loss Coefficient	0.3	–	–
Observation Prediction Loss Coefficient	0.03	–	–
Reward Prediction Loss Coefficient	10	–	–
Latent Dimensions	16	–	–
Pretrain VAE			
Learning Rate	0.0003	–	–
Number of Epochs	5000	–	–
Latent Dimensions	32	–	–
Policy Learning Hyperparameters			
Learning Rate (Policy)	0.0005	0.0005	0.0005
Learning Rate (VAE)	0.0003	–	–
Minibatch Size	2048	2048	2048
Number of Epochs (PPO)	24	8	8
Entropy coefficient	0.03	0.03	0.03
Recurrence	1	16	16
Number of Frames Per Update	8192	8192	8192

Table 5. Escape Room Training Hyperparameters

	Our Method	**Recurrent PPO**	**Unbiased Asymmetric Actor Critic**
Discount factor (γ)	0.99	0.99	0.99
Pretrain episodes	500	–	–
Pretrain Representation Model			
Number of Epochs	300	–	–
Batch Size	500	–	–
KL Coefficient	0.03	–	–
State Prediction Loss Coefficient	0.1	–	–
Observation Prediction Loss Coefficient	0.003	–	–
Reward Prediction Loss Coefficient	100	–	–
Latent Dimensions	16	–	–
Pretrain VAE			
Learning Rate	0.0003	–	–
Number of Epochs	5000	–	–
Latent Dimensions	32	–	–
Policy Learning Hyperparameters			
Learning Rate (Policy)	0.0005	0.0005	0.0005
Learning Rate (VAE)	0.0003	–	–
Minibatch Size	2048	2048	2048
Number of Epochs (PPO)	8	8	8
Entropy coefficient	0.01	0.01	0.01
Recurrence	1	16	16
Number of Frames Per Update	8192	8192	8192

D. Reflections on Societal Impact

Our work has focused on improving the performance of reinforcement learning in partially observable environments. In general, reinforcement learning poses multiple risks, including that RL agents may cause damage during exploration, or that the reward function will be misspecified and so even an optimal policy will have undesirable effects (Amodei et al., 2016). Partially observable environments often require especially long periods of exploration, and the RL agent may not even able to observe what aspects of the environment it is disrupting.

We did not specifically aim to make progress on general safety concerns with this work. However, the approach of training under full observability, which we (like some others) have used, may afford opportunities for avoiding some hazards during training. Also, we have seen that our approach can in some cases generate somewhat interpretable belief representations (Figure 7), which might help users to use it more safely.