One Does Not Simply Estimate State: Comparing Model-based and Model-free Reinforcement Learning on the Partially Observable MordorHike Benchmark

Sai Prasanna **André Biedenkapp*** **Raghu Rajan***
{ramans, biedenka, rajanr}@cs.uni-freiburg.de
University of Freiburg

Abstract

Evaluating reinforcement learning agents on partially observable Markov decision processes remains lacking as common benchmarks often do not require complex state estimation under non-linear dynamics and noise. We introduce *MordorHike*, a benchmark suite for rigorous state estimation testing, revealing performance gaps which would not be possible on other benchmarks. We present an evaluation framework assessing task performance and state estimation quality via probing. Using this framework, we empirically compare model-based (Dreamer, R2I) and model-free (DRQN) agents for sophisticated state estimation. The analysis reveals that Dreamer excels in sample efficiency and achieves superior performance in the hardest setting while R2I underperforms, suggesting its linear recurrent architecture may be a bottleneck. Further analysis reveals links between state estimation quality and task performance. Finally, out-of-distribution analysis shows a generalization gap for all algorithms, although Dreamer maintains an edge in the most challenging setting. The results highlight the need for robust state estimation and the need for proper evaluation benchmarks while validating the usefulness of *MordorHike* for future POMDP research.

1 Introduction

Learning optimal agents in partially observable Markov decision processes (POMDPs) [Åström, 1965] is a significant challenge. In POMDPs, agents perceive the world through noisy and/or incomplete sensors and must estimate the true state from a history of observations and actions to act optimally [Kaelbling et al., 1998]. This history is often compressed into a belief representation by a recursive state estimator, typically using recurrent neural networks (RNNs) [Ni et al., 2021].

While existing benchmarks like POPGym [Pašukonis et al., 2022] and MemoryMaze [Morad et al., 2023] have advanced partially observable RL evaluation, they fall short in testing sophisticated non-linear belief filtering under uncertainty. POPGym focuses on computational efficiency to test various different memorization strategies providing mostly small-scale memorization tasks, whereas MemoryMaze was designed to evaluate long-term memory through spatial navigation tasks. In contrast, we introduce *MordorHike* – an extension of the MountainHike environment [Igl et al., 2018, Lambrechts et al., 2022], a POMDP benchmark specifically designed to require sophisticated state estimation. The tasks in *MordorHike* feature persistent multi-modality in the true belief state, preventing simple state recovery strategies such as memorization. Successful solution strategies will thus likely require non-linear filtering approaches which are critical in many real world applications, ranging from underwater robotics navigation in ocean currents [Petillot et al., 2019] to legged robots handling terrain uncertainty due to unreliable sensors [Lee et al., 2024]. *MordorHike* isolates the challenge of state estimation by minimizing confounding factors like high-dimensional inputs or

*Equal contribution.

18th European Workshop on Reinforcement Learning (EWRL 2025).

long-horizon credit assignment and provides oracle belief states for evaluation of belief filtering via probing. Thus, our benchmark fills a crucial gap in the POMDP landscape.

An emerging, interesting area of research has been whether we need non-linear recurrence for state tracking, with work challenging the usefulness of linear RNNs [Merrill et al., 2024]. To highlight the utility of our benchmark, we shed light on this topic by comparing approaches using linear and non-linear RNNs.

Lastly, previous work has largely focused on model-free RL for POMDPs [Hausknecht and Stone, 2015, Zhu et al., 2018, Lambrechts et al., 2022]. We analyze such approaches for model-based RL (MBRL) and model-free RL (MFRL), with an additional advantage of MBRL being an explicit world model which allows more interpretability into the kinds of state estimation being performed.

Our experiments[1] provide a targeted evaluation of learning objectives and architectures for complex state estimation in POMDPs, offering insights for developing robust agents for real-world scenarios. Our main contributions are: (1) investigating MBRL (Dreamer [Hafner et al., 2023] (non-linear recurrence) and R2I [Samsami et al., 2024] (linear recurrence)) and MFRL (DRQN [Hausknecht and Stone, 2015, Zhu et al., 2018]) approaches to analyze performance and sample efficiency across varying state estimation difficulties; (2) analyzing state estimation performance and studying its correlation with task performance; (3) analyzing out-of-distribution generalization.

2 Related Work

Reinforcement learning (RL) in partially observable Markov decision processes (POMDPs) often critically hinges on effective state estimation [Sondik, 1978, Monahan, 1982, Kaelbling et al., 1998]. Existing POMDP benchmarks often have limitations for focused state estimation research. Some, like Atari [Mnih et al., 2013], become near fully observable with simple frame stacking [Mnih et al., 2013]. Others are computationally prohibitive or confounded by factors beyond state estimation [Vinyals et al., 2019]. Benchmarks like Memory Maze [Pašukonis et al., 2022] and POPGym [Morad et al., 2023] primarily test memory or information retrieval rather than continuous, non-linear belief filtering under uncertainty. *MordorHike* fills this gap by demanding sophisticated state estimation from low-dimensional, noisy observations leading to persistent multi-modal beliefs, and uniquely *provides oracle belief states* for direct evaluation.

As mentioned, our work builds on previous work evaluating MFRL for POMDPs [Hausknecht and Stone, 2015, Zhu et al., 2018, Lambrechts et al., 2022] by evaluating state-of-the-art MBRL agents with explicit world models that lend themselves better to the kind of probing analysis we employ.

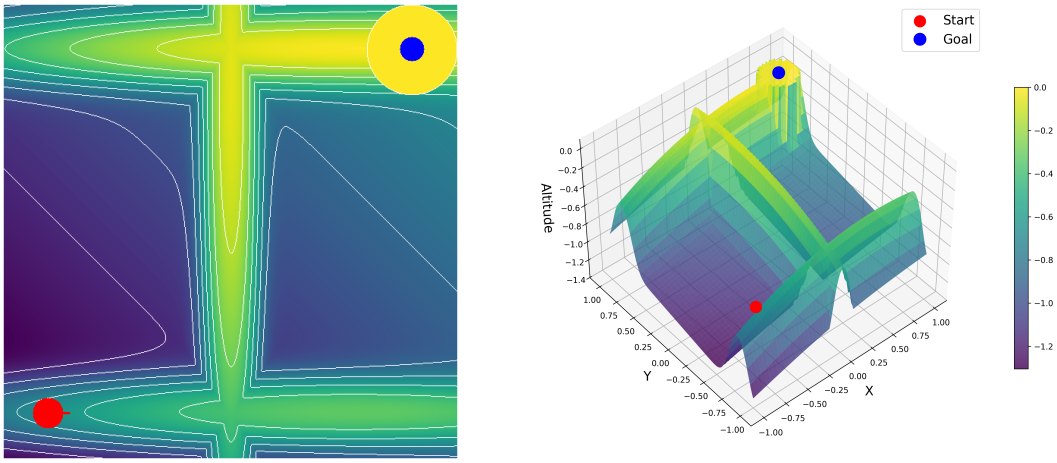
We focus on latent state models like Dreamer [Hafner et al., 2023], which uses non-linear recurrence (GRU [Cho et al., 2014]), and Recall-to-Imagine (R2I) [Samsami et al., 2024], featuring a linear recurrent architecture (S4 [Gu et al., 2022]). This architectural distinction is important to gain a deeper understanding of potential limitations of linear recurrence in complex state tracking [Merrill et al., 2024], which *MordorHike* is designed to test.

3 Background

POMDPs We define a POMDP as $(\mathcal{S}, \mathcal{A}, \mathcal{O}, R, T, O, p(s_0), \gamma)$, representing state, action, and observation spaces, reward, transition, and observation models, initial state distribution, and discount factor, respectively [Kaelbling et al., 1998]. At each timestep t, an agent takes action a_t leading to state $s_{t+1} \sim T(s_{t+1}|s_t, a_t)$, observation $o_{t+1} \sim O(o_{t+1}|s_{t+1}, a_t)$ and reward r_{t+1}. The agent's objective is to maximize the cumulative discounted future reward.

Belief state and Policy: The agent maintains a belief state $b_t(s)$, a posterior over states given history, updated via Bayesian filtering: $b_{t+1}(s') \propto O(o_{t+1}|s') \int T(s'|s, a_t) b_t(s) ds$. This integral is often intractable, necessitating approximations like particle filters which use weighted samples to represent $b_t(s)$ [Gustafsson, 2013]. RL agents for POMDPs learn a policy $\pi(a_t|o_{\leq t}, a_{<t})$ by typically using a state estimator f_θ (e.g., an RNN) to map a history to a belief representation $b_t = f_\theta(b_{t-1}, o_t, a_{t-1})$, which then conditions the policy $\pi_\theta(a_t|b_t)$. Model-free methods, such as DRQN, learn f_θ implicitly

[1]The code is available at https://github.com/sai-prasanna/mordorhike



(a) Top-down view with contour lines showing altitude.

(b) 3D visualization of the terrain with altitude represented as height.

Figure 1: *MordorHike* environment visualization. The agent navigates a terrain to reach a goal position. The red dot indicates the start position and the blue dot indicates the center of the goal position. The small line protruding from the red dot signifies the discretized orientation of the agent. Partial observability arises from noisy and indirect observations.

via the policy or value learning objective, while model-based methods (e.g., Dreamer or R2I) learn f_θ explicitly by modeling environment dynamics and observation/reward predictions.

4 MordorHike Benchmark

MordorHike[2] is a benchmark designed to isolate and test an agent's ability to infer its latent state. It consists of one environment with different variations designed to make it progressively difficult. In line with the MountainHike environment Lambrechts et al. [2022], Igl et al. [2018], an agent has to navigate a mountainous landscape, depicted in Figure 1, to reach a peak by making decisions in the presence of noisy and indirect observations. The agent's state comprises its continuous 2D position (x, y) on a map and its discrete orientation θ (e.g., North, East, South, West). The agent observes the state through noisy (Gaussian) altitude readings, $h(s) + \epsilon_o$ where $s = (x, y, \theta)$, and does not directly have access to s The altitude $h(s)$ is the maximum of a mixture of three Gaussians over the 2D coordinates (x, y). The terrain (Gaussian mixture) parameters remain fixed across all episodes to ensure consistent evaluation. The agent can either only translate or translate and rotate depending on the current environment's difficulty. The result of taking an action is also subject to Gaussian noise in the $(x, y)-$space but not in the $\theta-$space, i.e., translations are noisy and rotations are deterministic. The movements are clipped to stay within the map boundaries. Translation moves the agent 1 fixed step at a time after which the noise is applied. Altitude does not affect the agent's movement capability. The agent moves freely in the 2D $(x, y)-$space with altitude serving only as an indirect position sensor. As such, the altitude can be seen as a proxy reading of the (x, y) position of the agent and since multiple (x, y)s correspond to the same altitude, estimating the true underlying state from the observations is challenging.

The agent receives a *dense reward* at each step, which is the negative of its current altitude relative to the highest point (the goal); thus, higher altitudes yield better rewards. When the agent successfully navigates to a designated goal region around the peak defined by the altitude, the agent the episode terminates with the agent receiving zero reward. The negative rewards everywhere but the goal region ensure that the reward maximization objective encourages the agent to reach the goal as fast as possible.

[2]The name is inspired by the fictional region of Mordor from The Lord of the Rings. Just as Mordor makes navigation difficult for the Hobbits, our benchmark presents a significant navigational challenge to RL agents.

4.1 Environment Variations: Difficulty Progression

Three difficulty levels systematically increase the challenge of state estimation by modifying elements of the underlying MDP (specifically, the action set $\mathcal{A}$ and the initial state distribution $p(s_0)$ concerning orientation). The easier settings are also present in MountainHike. The three difficulty settings are:

Easy: The initial orientation of the agent is fixed ($0°$) and the action space includes only translation (forward, backward, left, right). *Resulting POMDP Challenge:* Estimate (x, y) position knowing the fixed orientation. Primarily involves filtering noisy altitude observations while accounting for stochastic transitions.

Medium: The initial orientation is random (uniform over $\{0°, 90°, 180°, 270°\}$) and the action space is the same as for the easy setting (translation only). *Resulting POMDP Challenge:* Estimate (x, y) and infer the unknown but fixed initial orientation θ_0. Requires using sequences of ambiguous observations to disambiguate the initial orientation, adding complexity to belief tracking.

Hard: The initial orientation is random and the action space includes both translation and rotation actions (rotate left, rotate right) in addition to the translation actions from easier settings. Importantly, in the hard setting, the translation actions are relative to the current orientation (forward moves in the direction the agent is facing), making rotation actions non-redundant as they change the reference frame for subsequent movements. *Resulting POMDP Challenge:* Estimate (x, y) while simultaneously tracking the changing orientation θ_t, starting from an unknown θ_0. Requires disentangling altitude changes caused by noisy observations, stochastic translations (dependent on current θ_t), and deterministic rotations (which change θ_t). This is the most complex belief tracking scenario for *MordorHike*.

4.2 Formal POMDP Description

The *MordorHike* environment is a Partially Observable Markov Decision Process (POMDP). The underlying Markov Decision Process (MDP) is defined by the tuple $(\mathcal{S}, \mathcal{A}, T, R, p(s_0), \gamma)$, where γ is the discount factor (set to 0.99 in our experiments).

State Space ($\mathcal{S}$). The state $s_t = (x_t, y_t, \theta_t)$ at timestep t includes the agent's continuous 2D position $(x_t, y_t) \in [-1, 1] \times [-1, 1]$ and its discrete orientation $\theta_t \in \{0°, 90°, 180°, 270°\}$.

Action Space ($\mathcal{A}$). The agent selects actions from a discrete set $\mathcal{A}$. The specific actions available depend on the difficulty level (see Section 4.1) but generally involve either translations (moving forward, backward, left, or right relative to θ_t) or rotations (changing θ_t by $\pm 90°$). For example, in the 'easy' and 'medium' settings, $\mathcal{A} = \{\text{forward}, \text{backward}, \text{left}, \text{right}\}$, and in the 'hard' setting, $\mathcal{A} = \{\text{forward}, \text{backward}, \text{rotate left}, \text{rotate right}\}$. Each translation step is a fixed distance (e.g., 0.1 units).

Initial State Distribution ($p(s_0)$). The agent's starting 2D position (x_0, y_0) is fixed at $(-0.8, -0.8)$. The initial orientation θ_0 depends on the difficulty level:

$$p(s_0 = ((-0.8, -0.8), \theta)) = \begin{cases} 1 & \text{if } \theta = 0° \text{ (Easy difficulty)} \\ 1/4 & \text{if } \theta \in \{0°, 90°, 180°, 270°\} \text{ (Medium, Hard difficulties)} \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

Transition Dynamics ($T(s_{t+1}|s_t, a_t)$). The next state s_{t+1} is sampled from the transition distribution T. Given the current state $s_t = (x_t, y_t, \theta_t)$ and action a_t, the environment transitions stochastically as follows: Let $(\Delta x_{intended}, \Delta y_{intended})$ be the intended position change based on a_t and θ_t, and $\Delta\theta_{intended}$ be the intended orientation change. Noise is added to the position update:

$$x_{t+1} = \text{clip}(x_t + \Delta x_{intended} + \epsilon_x, -1, 1) \tag{4.2}$$

$$y_{t+1} = \text{clip}(y_t + \Delta y_{intended} + \epsilon_y, -1, 1) \tag{4.3}$$

$$\theta_{t+1} = (\theta_t + \Delta\theta_{intended}) \bmod 360° \tag{4.4}$$

where $\epsilon_x, \epsilon_y \sim \mathcal{N}(0, \sigma_T^2)$ are independent Gaussian noise variables with $\sigma_T = 0.05$. Movement is clipped to the map boundaries $[-1, 1] \times [-1, 1]$. Orientation changes are deterministic given the chosen rotation action.

Reward Model ($R(r_{t+1}|s_{t+1}, a_t)$). A deterministic reward r_{t+1} is received after transitioning into state s_{t+1}. It is based on the agent's altitude $h(s_{t+1})$ relative to the goal peak (defined by a specific terrain function):

$$r_{t+1} = h(s_{t+1}) \tag{4.5}$$

The function $h(s)$ is designed such that $h(s) = 0$ at the goal peak (e.g., at position $(0.8, 0.8)$) and is negative elsewhere on the map.

Termination. An episode ends if the agent enters the goal region, defined as a circle of radius 0.2 around the goal peak position:

$$\sqrt{(x_{t+1} - 0.8)^2 + (y_{t+1} - 0.8)^2} \leq 0.2 \tag{4.6}$$

Episodes also terminate after a maximum number of steps (e.g., 200).

Observation Space ($\mathcal{O}$). At each step $t + 1$, after transitioning to state s_{t+1}, the agent receives a scalar observation o_{t+1} from a continuous space $\mathcal{O} \subset \mathbb{R}$.

Observation Model ($O(o_{t+1}|s_{t+1}, a_t)$). The observation is the true altitude $h(s_{t+1})$ at the agent's position in state s_{t+1}, corrupted by Gaussian noise:

$$o_{t+1} \sim O(\cdot|s_{t+1}, a_t) = \mathcal{N}(h(s_{t+1}), \sigma_O^2) \tag{4.7}$$

This means $o_{t+1} = h(s_{t+1}) + \epsilon_o$, where $\epsilon_o \sim \mathcal{N}(0, \sigma_O^2)$ with $\sigma_O = 0.1$. The agent does not observe its (x, y) coordinates or its orientation θ directly.

4.3 Oracle Belief States

Importantly, to advance research on state estimation capabilities and belief representation, our environment distinctively offers access to approximate oracle belief states via particle filtering. Refer to Appendix A for a description of how particle filtering is done when we have Oracle access to the POMDP structure. Figure 2 shows an example of particle filtering in the *MordorHike* medium difficulty.

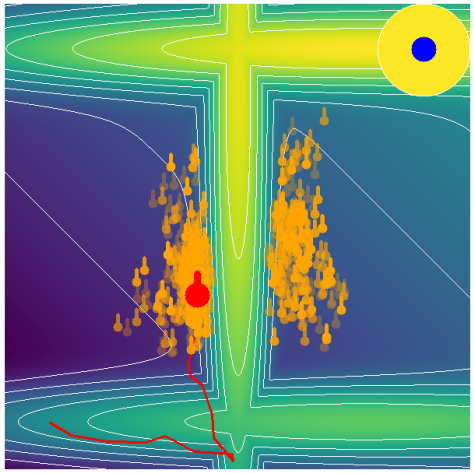


Figure 2: Example particle filter visualization in *MordorHike*. Orange dots with protruding lines represent particles forming samples from the belief distribution over possible agent positions and orientations. White contour lines show the terrain elevation. The red dot indicates the agent's true position, the small line shows its orientation and the blue dot shows the goal. Note the multi-modal nature of the belief distribution, with particles clustered in multiple locations despite using the true dynamics and observation models, demonstrating the inherent ambiguity of state estimation in *MordorHike*.

4.4 The Challenge of MordorHike

MordorHike specifically tests **belief filtering** under key technical challenges that distinguish it from existing memory-focused POMDP benchmarks:

Persistent Multi-modal Beliefs. The altitude function $h(s) = \max(\text{Gaussian}_1, \text{Gaussian}_2, \text{Gaussian}_3)$ creates observational ambiguity where multiple (x, y) positions yield identical readings. As Figure 2 shows, this leads to persistent multi-modal beliefs that prevent simple memorization strategies. **Learning to Approximate Bayes Filtering.** Neural agents must learn to mimic Bayesian belief updates without knowing true dynamics. With both transition noise and observation noise, agents cannot trust observations blindly but must learn to weight them against historical evidence, discovering that identical altitude readings can correspond to different locations.

This focus on learning sophisticated probabilistic filtering under persistent uncertainty positions distinguishes *MordorHike* from other memory-focused environments like POPGym and MemoryMaze.

5 Experiments

In this section, we present our experimental evaluation of three reinforcement learning algorithms across the *MordorHike* environments. We describe our training setup and evaluation protocol and present results addressing various research questions. Our evaluation goes beyond standard performance metrics to assess the quality of belief state representations and their generalization to out-of-distribution scenarios.

5.1 Algorithms and Training Procedures

We evaluate three RL algorithms with distinct approaches to state estimation in POMDPs. Each must learn a state estimator to form a belief representation from history and a policy that maps these beliefs to actions.

Model-based Approaches: These methods learn an explicit world model. We study latent state models where the learned representations are shared with the policy. *Dreamer* [Hafner et al., 2023] learns a generative world model using a Recurrent State Space Model (RSSM). Its belief representation combines a deterministic state h_t from a GRU and a stochastic latent state z_t, learned via variational inference to predict future observations and rewards. Policy learning occurs via actor-critic methods in the learned latent space. *Recall to Imagine (R2I)* [Samsami et al., 2024] adapts Dreamer's RSSM framework but replaces the GRU with a linear recurrent architecture (S4 model [Gu et al., 2022]) and uses a non-recurrent posterior for z_t based only on the current observation o_t to enable parallelization. Contrasting these methods allows us to investigate if linear recurrence limits complex state estimation. **Model-free Approach:** *DRQN* [Hausknecht and Stone, 2015, Zhu et al., 2018] learns belief representations implicitly. Following Lambrechts et al. [2022], we use a GRU to update a deterministic belief representation $b_t = f_\theta(b_{t-1}, o_t, a_{t-1})$. This b_t directly inputs to an MLP estimating Q-values, optimizing f_θ end-to-end via temporal difference learning [Sutton and Barto, 2018].

Each algorithm is trained for $500\,000$ environment steps across three difficulty levels of *MordorHike*. For each setting, we use five different random seeds. This differs from the evaluation protocol of Lambrechts et al. [2022], where they evaluate over $10\,000$ episodes, which can have a differing number of environment steps and may be less accurate.

We performed hyperparameter tuning using *random search with priors* [Mallik et al., 2023] with 50 configurations per algorithm on the medium difficulty level, then applied the best configurations across all difficulty levels. Details of the hyperparameters used are provided in the Appendix B.

Throughout the training phase, we systematically record checkpoints for each model after every $100\,000$ steps taken in the environment. This results in the generation of five distinct checkpoints over the course of an entire training session for each model. By evaluating the models at these specific checkpoints, we obtain insights into the agents' learning dynamics.

5.2 Evaluation

For each checkpoint of the RL agents saved during training, we collect 300 episodes by executing the agent's policy and recording the desired evaluation metrics and visualizations during these rollouts.

Task Performance For each episode we calculate the **score**. The score is the undiscounted sum of rewards collected in an episode until truncation. We average the scores across episodes to get the average score for a given checkpoint and seed. To summarize this average score across seeds, we follow the recommendation of Agarwal et al. [2021] and report the average scores' Inter-Quartile Mean (IQM).

Belief Representation Probing To directly assess the quality of learned belief representations (b_t), we employ linear probing. For each algorithm, we train a linear model to predict the approximate oracle belief state from b_t. The oracle belief states are approximated using a particle filter given the true POMDP model, run on trajectories collected from the agent's policy. These particle distributions are discretized into a $D_x \times D_y \times D_\theta$ grid ($20 \times 20 \times 4$) to serve as the target P_{true} for the linear probe. The probe, $f_\psi(b_t)$, is then trained to minimize the average KL divergence $\mathcal{L}(\psi) = \mathrm{E}[]\mathrm{KL}(P_{\text{true}} \parallel f_\psi(b_t))$ with early stopping on a held-out portion of the collected trajectories.

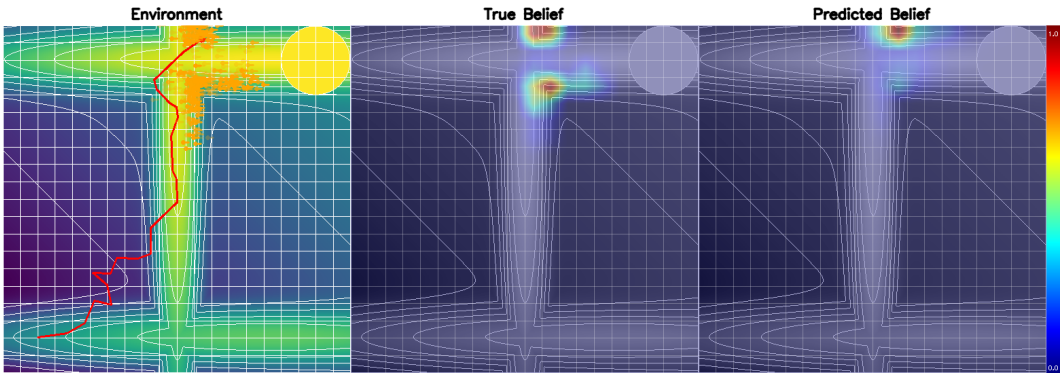


Figure 3: Illustration of probing: In left plot, a particular timestep in the agent's trajectory is discretized. The discretized grid is used to marginalize over the particles of a particle filter that employs the true POMDP model which gives us the heatmap in the center plot illustrating the discrete probability mass over the X and Y coordinates (we also marginalize the orientation for obtaining this two-dimensional illustration). In the rightmost plot, a linear probe is used to predict this oracle belief using the agent's belief representation.

The quality of an agent's belief representation is quantified by the average KL divergence on an *unseen* test set of trajectories. Lower KL divergence (in nats) indicates that the agent's internal representation b_t more effectively encodes the true belief state in a linearly decodable manner. Figure 3 provides an overview of the process.

Generalization The state estimator of the baseline agents compresses the history of observations and actions into a belief state representation. The training data is, however, dependent on the current policy and this data distribution gets increasingly narrow as the policy converges to the optimal policy [see also the discussion by; Suau et al., 2024].

As discussed previously, we can compute the optimal belief state using Bayes filtering or its approximation for any trajectory. In case the state estimator of our RL agents also learns some form of actual Bayes filtering (implicitly or explicitly), it should be able to use the learned Bayes filter to generalize and perform better on out-of-distribution (OOD) trajectories. To test this hypothesis, we generate novel trajectories that the state estimator has not encountered during training. We take two different approaches to generate such OOD trajectories.

1. Noisy Actions: Here, we employ random actions with a probability of 0.25 and the trained policy with a probability of 0.75. This creates trajectories that differ from those seen by the state estimator and the policy during training while remaining relatively close to on-policy trajectories. This allows us to assess how well the belief state estimator and the policy generalize to novel situations that do not deviate drastically from the training distribution.

2. Waypoint Navigation: While action noise can generate OOD trajectores, they can resemble training trajectories as the policy potentially covers similar regions of the state space. To have even lower overlap with the training distribution, we propose propose to sample waypoint sequences, in our particular instantiation, each with three waypoints. The first two waypoints are sampled randomly with a fixed distance (1 unit) apart, while the final waypoint is always at the center of the map. We use an oracle state to derive an oracle action that guides the agent at each step to trace the path between waypoints. This is necessary, due to the stochasticity of transitions in MordorHike. After reaching the final waypoint, we let the agent follow its learnt policy. We sample 30 such waypoint sequences which are fixed across all tasks and algorithms to make the OOD trajectories comparable. We collect 10 episodes per waypoint sequence, which leads to 300 episodes per algorithm and difficulty level.

5.3 Results

In this section, we answer the following three research questions: How do our approaches comprising of different objectives (model-based vs model-free) and architectures (GRU vs S4) compare in the three levels of *MordorHike* (easy/medium/hard) in terms of

1. Final performance and sample efficiency.
2. Estimating the true belief, and if this correlates with task performance.
3. Generalizing to out-of-distribution trajectories.

5.3.1 Task Performance

Table 1: **Consolidated average score comparison (IQM across seeds)** across in-distribution, action noise out-of-distribution (OOD), and waypoint OOD conditions, and difficulty levels. The agent is evaluated at 500 000 steps. Higher values are better.

	In-distribution			Action noise OOD			Waypoint OOD		
Algorithm	**Easy**	**Medium**	**Hard**	**Easy**	**Medium**	**Hard**	**Easy**	**Medium**	**Hard**
Dreamer	**-15.18**	**-28.51**	**-28.32**	-24.96	-52.44	**-80.07**	-38.68	-65.81	**-76.77**
R2I	-27.97	-73.11	-110.51	-29.91	-82.22	-116.78	-40.88	-81.90	-100.61
DRQN	**-15.18**	-29.09	-109.77	**-23.04**	**-40.61**	-118.79	**-35.20**	**-59.58**	-106.11

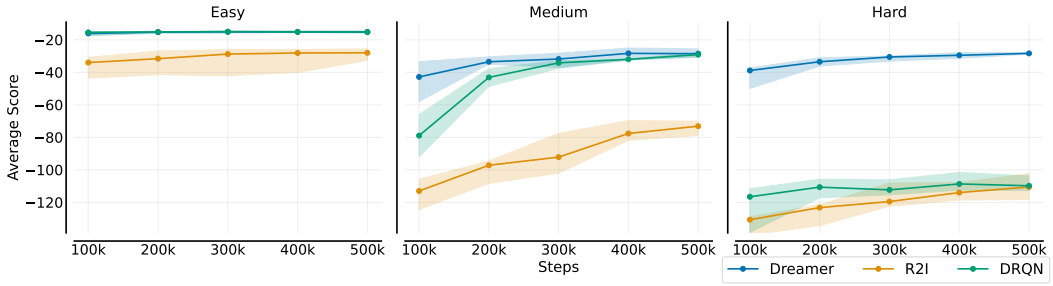


Figure 4: **In-distribution Performance** evaluation of the RL algorithms across difficulty levels. We evaluate the agent at each 100 000 environment step by averaging scores over 300 episode rollouts. Lines show IQM of the average scores across five seeds, with shaded regions indicating 95%-confidence intervals computed using bootstrap sampling. Higher scores are better.

The results for final performance (Table 1) show that model-free (DRQN) and model-based (Dreamer) approaches perform similarly in settings with simpler partial observability challenges (easy and medium). However, as the challenge increases in the hard setting, only the non-linear world model approach (Dreamer) maintains performance while DRQN fails to learn effectively. R2I underperforms across all settings despite sharing the world modelling objective with Dreamer, suggesting that architectural choices (linear vs non-linear recurrence) significantly impact performance beyond learning objectives. In terms of sample efficiency (Figure 4), Dreamer is consistently strong within 100 000 steps across all levels and superior to DRQN.

We present an additional metric of interest, specifically the success rate which quantifies the proportion of instances where the objective is achieved within the specified truncation timeframe. Detailed outcomes are consistent with the analysis presented here and are provided in Appendix C.1.

5.3.2 State Estimation Quality

For linear probing, if we look at the results for on-policy trajectories in Table 2 (in-distribution column) and progress during training in Figure 5, we observe that DRQN performs the best followed closely by Dreamer in the easy and medium settings, while R2I significantly lags behind. However, in the hardest setting Dreamer once again significantly outperforms both DRQN and R2I. These results also point to a clear correlation in the state estimation performance judged by the KL-divergence and the final performance. While Dreamer's superior performance in the hard setting aligns with its significantly better state estimation, interestingly, Dreamer's state estimation quality in the hard setting is better than even in the medium setting and merits further investigation.

Table 2: Consolidated KL divergence comparison (IQM across seeds) for in-distribution, Action noise OOD, and Waypoint OOD evaluations across difficulty levels. The agent is evaluated at 500 000 steps. Lower values indicate better state estimation.

	In-distribution			Action noise OOD			Waypoint OOD		
Algorithm	**Easy**	**Medium**	**Hard**	**Easy**	**Medium**	**Hard**	**Easy**	**Medium**	**Hard**
Dreamer	0.410	0.904	**0.689**	0.732	1.537	**2.173**	1.124	2.159	**2.390**
R2I	0.685	1.838	2.955	1.000	2.521	3.074	1.304	2.757	3.351
DRQN	**0.370**	**0.838**	2.528	**0.641**	**1.270**	2.853	**0.925**	**1.979**	2.864

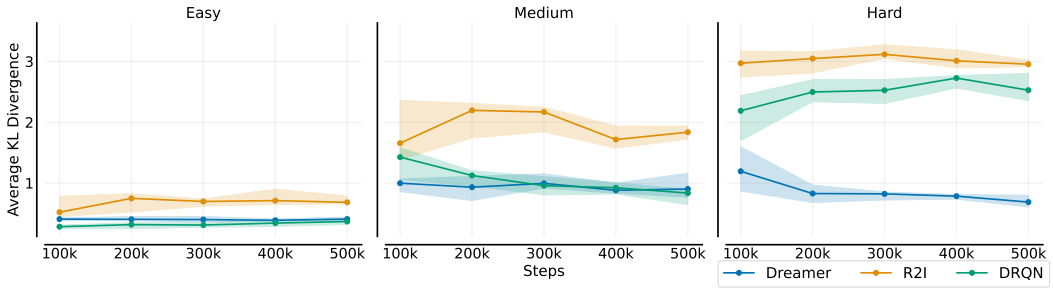


Figure 5: **In-distribution KL divergence** between predicted and true belief distributions across difficulty levels. At each 100 000 environment step, we evaluate the agent by averaging per-step KL divergence within each episode, then averaging these episodic means across 300 episodes. Lines show IQM of the average scores across five seeds, with shaded regions indicating 95%-confidence intervals computed using bootstrap sampling. Lower values are better.

5.3.3 Generalization

We report the generalization of the policy Table 1 and state estimation Table 2 to the two OOD settings. While we focus on the final agent's numbers in the table here, we report the scores and state estimation quality throughout training with learning curve plots in Appendices C.2 and C.3.

All algorithms show performance degradation in the OOD settings. As for the state estimation in the previous sub-section, DRQN generalizes best in the easy and medium settings with Dreamer close behind and R2I lagging behind. However, once again, Dreamer shows its advantages in the hard setting outperforming the others by a significant margin – this despite also suffering a large drop in performance in this setting.

The KL-divergence values follow similar patterns, with all approaches showing substantial increases in OOD settings. DRQN maintains the lowest KL divergence in easy and medium settings, while Dreamer maintains an advantage in the hard setting despite its substantial degradation.

These results reveal an important trade-off: DRQN exhibits better relative generalization in simpler environments with smaller performance drops from in-distribution to OOD settings. However, despite experiencing more significant absolute performance drops in the hard setting (from -28 to -80), Dreamer still maintains significantly better absolute performance than its competitors.

The important takeaways from these experiments are the following:

Correlation with belief quality: Lower KL divergence consistently correlates with better absolute performance across all settings and algorithms.

All approaches struggle with distribution shift: Every approach shows substantial degradation when faced with OOD trajectories, highlighting the challenging nature of generalization in POMDP environments.

Relative vs. absolute generalization: DRQN shows better relative generalization (smaller performance drops) in easy/medium settings. At the same time, Dreamer maintains better absolute performance in the most complex setting despite more significant drops.

These findings suggest that no approach has fully solved the generalization challenge in complex POMDPs, with all algorithms struggling to maintain consistent performance across different trajectory distributions.

5.4 Discussion

Our empirical findings revealed some unexpected patterns that warrant a more extensive examination. Notably, DRQN demonstrated superior generalizability in less complex scenarios when juxtaposed with MBRL methods. Additionally, there was a considerable disparity in the performance levels of R2I across all tested conditions.

The Impact of Exploration Strategies on Generalization We posit that the superior relative generalization ability of the DRQN in easy and medium settings may be attributed to its application of epsilon-greedy exploration during the training phase. This exploration strategy inherently subjects the agent to a more diverse set of trajectories, thereby potentially enhancing its resilience to action noise and waypoint navigation challenges. Conversely, Dreamer's entropy-regularized policy might lack sufficient exploration capability, resulting in a more fragile state estimation.

We further propose that, in the most challenging setting, where the agent is required to manage its orientation as well, the epsilon-greedy exploration may induce excessive erratic trajectories. Consequently, the DRQN agent may struggle to meaningfully learn from the exploration patterns in this setting, complicating the estimation of the belief state.

Understanding R2I's Underperformance Dreamer and R2I show striking performance differences despite a shared world modelling objective, which is surprising given R2I's success in environments from POPGym/MemoryMaze [Samsami et al., 2024]. We speculate two reasons for this disparity:
1. Non-recurrent posterior R2I's posterior depends only on the current observation to allow for parallelization. While this might work for memory tasks, our task which needs recursive Bayes-like updates, may require a recursive posterior dependent on prior belief. This could be crucial for real-world noisy sensor applications.
2. Limitations of Linear RNNs Merrill et al. [2024] show limitations of linear RNNs (like S4) without sequence non-linearity in state-tracking tasks, where S4 fails to generalize beyond the training length. We speculate that state estimation under stochasticity without supervision may demand similar generalization, which S4 (even trained sequentially) might fail to provide. Our findings might be initial evidence that parallelizable architectures like S4, despite computational benefits, may fail in certain kinds of POMDPs.

6 Conclusions and Future Work

We introduced *MordorHike*, a benchmark that provides ground-truth estimates for rigorous POMDP state estimation evaluation, comparing model-based (Dreamer, R2I) and model-free (DRQN) RL. Overall, Dreamer showed superior sample efficiency and performance in complex state estimation while R2I's underperformance suggests its linear recurrent architecture may be a bottleneck. We also showed links between belief representation quality (via linear probing) and task performance. *MordorHike* emphasizes state estimation capabilities whereas other existing benchmarks are memory-focused. Finally, our out-of-distribution generalization analysis revealed performance gaps, though Dreamer maintained the best absolute performance under the most significant distributional shifts.

Future work includes: (1) Developing robust architectures and advanced variational inference for better estimation and generalization (e.g., modern sequence models, structured particle filtering). (2) Taxonomizing POMDP challenges (belief filtering vs. memory retention) to guide algorithm development. (3) Creating better POMDP exploration mechanisms for robust state estimators. Our work advances understanding of state estimation in complex POMDPs, highlights limits, and offers *MordorHike* as a tool for progress.

Acknowledgments

The authors acknowledge funding through the research network "Responsive and Scalable Learning for Robots Assisting Humans" (ReScaLe) of the University of Freiburg. The ReScaLe project is funded by the Carl Zeiss Foundation. Sai Prasanna also acknowledges funding by the Konrad Zuse School of Excellence in Learning and Intelligent Systems (ELIZA) scholarship.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 2021.

Karl Johan Åström. Optimal control of markov processes with incomplete state information i. *Journal of mathematical analysis and applications*, 10:174–205, 1965.

Kyunghyun Cho, B. V. Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *Conference on Empirical Methods in Natural Language Processing*, 2014. doi: 10.3115/v1/D14-1179.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *The International Conference on Learning Representations (ICLR)*, 2022.

F. Gustafsson. Particle filters. *Encyclopedia of Systems and Control*, 2013. doi: 10.3150/12-BEJSP07.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv: 2301.04104*, 2023.

Matthew J. Hausknecht and P. Stone. Deep recurrent q-learning for partially observable mdps. *AAAI Fall Symposia*, 2015.

Maximilian Igl, Luisa Zintgraf, Tuan Anh Le, Frank Wood, and Shimon Whiteson. Deep variational reinforcement learning for POMDPs. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2117–2126. PMLR, 10–15 Jul 2018.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artif. Intell.*, 101:99–134, 1998.

Rudolph Emil Kalman and Others. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

Gaspard Lambrechts, Adrien Bolland, and D. Ernst. Recurrent networks, hidden states and beliefs in partially observable environments. *Transactions on Machine Learning Research*, 2022. doi: 10.48550/arXiv.2208.03520.

Joonho Lee, Marko Bjelonic, Alexander Reske, Lorenz Wellhausen, Takahiro Miki, and Marco Hutter. Learning robust autonomous navigation and locomotion for wheeled-legged robots. *Science Robotics*, 9(84), 2024.

Neeratyoy Mallik, Eddie Bergman, Carl Hvarfner, Danny Stoll, Maciej Janowski, Marius Lindauer, Luigi Nardi, and Frank Hutter. Priorband: Practical hyperparameter optimization in the age of deep learning. In *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS'23)*, 2023.

William Merrill, Jackson Petty, and Ashish Sabharwal. The illusion of state in state-space models. *International Conference on Machine Learning*, 2024. doi: 10.48550/arXiv.2404.08819.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv: 1312.5602*, 2013.

George E. Monahan. A survey of Partially Observable Markov Decision Processes: Theory, models, and algorithms. *Management Science*, 28(1):1–16, January 1982.

Steven D. Morad, Ryan Kortvelesy, Matteo Bettini, Stephan Liwicki, and Amanda Prorok. Popgym: Benchmarking partially observable reinforcement learning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

Tianwei Ni, Benjamin Eysenbach, and R. Salakhutdinov. Recurrent model-free rl can be a strong baseline for many pomdps. *International Conference on Machine Learning*, 2021.

Jurgis Pašukonis, Timothy Lillicrap, and Danijar Hafner. Evaluating long-term memory in 3d mazes. In *International Conference on Learning Representations*, 2022. doi: 10.48550/arXiv.2210.13383.

Yvan R. Petillot, Gianluca Antonelli, Giuseppe Casalino, and Fausto Ferreira. Underwater robots: From remotely operated vehicles to intervention-autonomous underwater vehicles. *IEEE Robotics & Automation Magazine*, 26(2):94–101, 2019. doi: 10.1109/MRA.2019.2908063.

Mohammad Reza Samsami, Artem Zholus, Janarthanan Rajendran, and Sarath Chandar. Mastering memory tasks with world models. In *The Twelfth International Conference on Learning Representations*, 2024.

Edward J. Sondik. The optimal control of partially observable markov processes over the infinite horizon: Discounted costs. *Oper. Res.*, 26(2):282–304, 1978. doi: 10.1287/OPRE.26.2.282.

Miguel Suau, Matthijs T. J. Spaan, and Frans A. Oliehoek. Bad habits: Policy confounding and out-of-trajectory generalization in RL. *RLJ*, 4:1711–1732, 2024.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0262039249.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *nature*, 575(7782):350–354, 2019.

Pengfei Zhu, Xin Li, Pascal Poupart, and Guanghui Miao. On improving deep reinforcement learning for pomdps. *arXiv preprint arXiv: 1804.06309*, 2018.

A Particle Filters

The belief state can be computed in closed form only in cases where the dynamics and observation models belong to certain families of distributions, for example, Kalman filter [Kalman and Others, 1960] assumes that the transition and the observation distributions are gaussian. When this is not the case, we must use approximate methods such as sequential Monte Carlo sampling or particle filters [Gustafsson, 2013].

Particle filters approximate the belief state $b_t(s)$ using a set of weighted particles $\{(s_t^i, w_t^i)\}_{i=1}^N$, where s_t^i represents the state of particle i and w_t^i its normalized weight ($\sum_i w_t^i = 1$). The belief is approximated as follows:

$$b_t(s) \approx \sum_{i=1}^{N} w_t^i \delta(s - s_t^i) \tag{A.1}$$

where δ is the Dirac delta function. At each timestep, particles are propagated through the transition model and reweighted based on the observation likelihood:

$$w_t^i \propto w_{t-1}^i O(o_t | s_t^i) \tag{A.2}$$

A resampling step is performed regularly to prevent particle degeneracy, where most particles end up with near-zero weights. During resampling, particles with low weights are eliminated while particles with high weights are duplicated, maintaining a fixed number of particles N. This process creates an unweighted particle set that better represents the high-probability regions of the belief state.

B Hyperparameters

We performed hyperparameter tuning using random search with a budget of 50 configurations for all approaches (DRQN, Dreamer, R2I) on the medium-difficulty environment. Due to limited computational resources, we used the best hyperparameters found for the medium environment across all difficulty levels (easy, medium, and hard). This hyperparameter tuning process led to significant performance improvements, particularly for DRQN in the medium environment.

The following subsections detail the final hyperparameters used for each algorithm and the linear probe.

B.1 DRQN

We choose all the default hyperparameters from Samsami et al. [2024] and modify the following.

Name	Value
Batch size	126
Train interval	35 (environment steps)
Target Network Update interval	280 (environment steps)
Updates per Train step	8
ϵ exploration factor	0.249
GRU Hidden Units	54
GRU Layers	2
Gradient Steps	23
Learning rate	$1.79 * 10^{-4}$

Table 3: DRQN hyperparameters. The same values are used across all experiments.

B.2 Dreamer

We choose the *small* variant of DreamerV3 with all hyperparameters taken from Hafner et al. [2023] and modify the following hyperparameters.

Name	Value
General	
Train ratio	262
Batch size	8
World Model	
Deterministic State model (GRU) units	256
Hidden Size	
MLP units	165
Number of latents	32
Classes per latent	11
Learning rate	$2.3 * 10^{-4}$
Actor Critic	
MLP units	165
Discount horizon	100
Learning rate	$2.38 * 10^{-4}$

Table 4: DreamerV3 tuned hyperparameters. The same values are used across all experiments.

B.3 R2I

We choose all the default hyperparameters from Samsami et al. [2024] and modify the following.

Name	Value
General	
Train ratio	457
Batch size	4
World Model	
Deterministic State model (S4) units	1024
Hidden Size	512
S4 Layers	4
MLP Layers	1
MLP units	165
Number of latents	32
Classes per latent	32
Learning rate	$1.79 * 10^{-4}$
Actor Critic	
MLP units	165
MLP Layers	1
Discount horizon	100
Learning rate	$2.34 * 10^{-4}$

Table 5: R2I tuned hyperparameters. The same values are used across all experiments.

B.4 Linear Probe

The linear probe uses a single linear layer that maps the belief representation from different environments to the discretized belief state distribution with the output size of $1600 = (20 * 20 * 4)$. We save the checkpoint with the best validation loss and use that to calculate the KL divergence on the test set.

Name	Value
General	
Epochs	300
Adam Learning Rate	$3 * 10^{-4}$
Batch Size	64

Table 6: Linear Probe Parameters

C Additional Results

C.1 Success Rate

Algorithm	In-distribution			Action noise OOD			Waypoint OOD		
	Easy	Medium	Hard	Easy	Medium	Hard	Easy	Medium	Hard
Dreamer	**100.0%**	**99.2%**	**100.0%**	97.3%	85.9%	**71.7%**	90.4%	79.7%	**92.2%**
R2I	54.3%	41.7%	5.8%	55.4%	39.8%	4.7%	39.7%	36.4%	14.9%
DRQN	**100.0%**	98.7%	16.7%	**99.7%**	**97.9%**	11.9%	**97.1%**	**83.3%**	16.6%

Table 7: Final average success rate comparison (IQM across seeds) across different evaluation conditions and difficulty levels. The agent is evaluated at the 500k steps. Higher values are better.

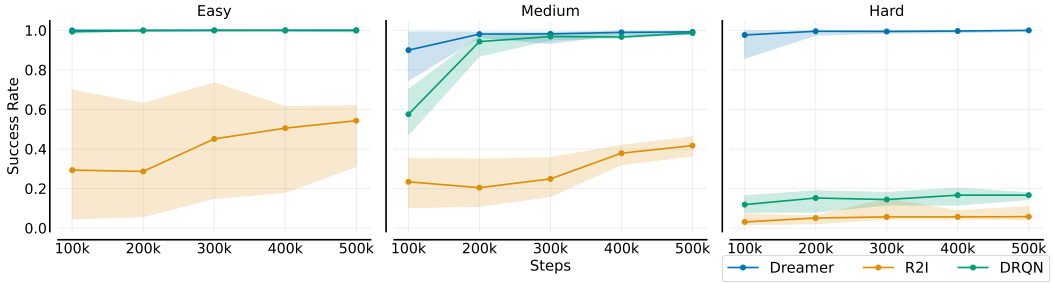


Figure 6: **In-Distribution Policy: Success Rate** evaluation of the RL algorithms across difficulty levels. We evaluate the agent at each 100k environment step by averaging scores over 300 episode rollouts. Lines show IQM of the average scores across five seeds, with shaded regions indicating 95% confidence intervals computed using bootstrap sampling. Higher scores are better.

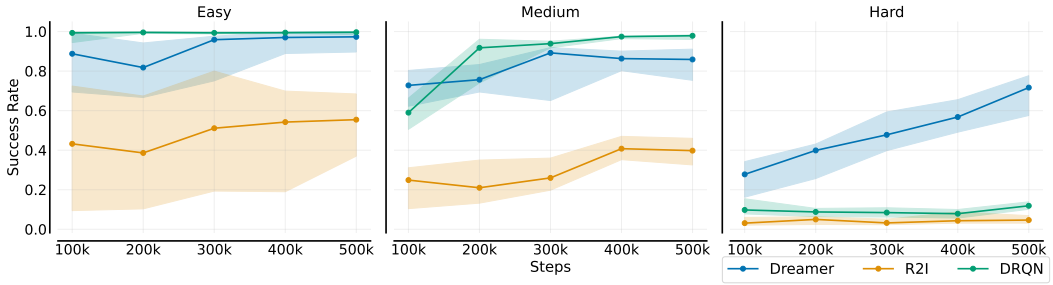


Figure 7: **Noisy Policy: Success Rate** evaluation of the RL algorithms across difficulty levels. We evaluate the agent at each 100k environment step by averaging scores over 300 episode rollouts. Lines show IQM of the average scores across five seeds, with shaded regions indicating 95% confidence intervals computed using bootstrap sampling. Higher scores are better.

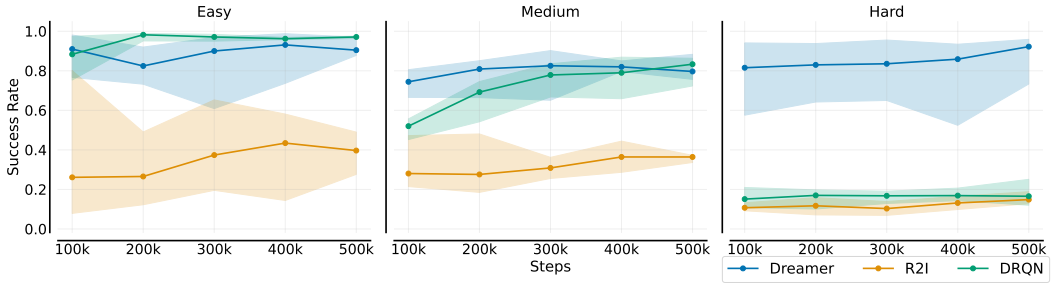


Figure 8: **Waypoint Policy: Success Rate** between predicted and true belief distributions across difficulty levels. At each 100k environment step, we evaluate the agent by averaging per-step KL divergence within each episode, then averaging these episodic means across 300 episodes. Lines show IQM of the average scores across five seeds, with shaded regions indicating 95% confidence intervals computed using bootstrap sampling. Lower values are better.

C.2 Noisy Actions

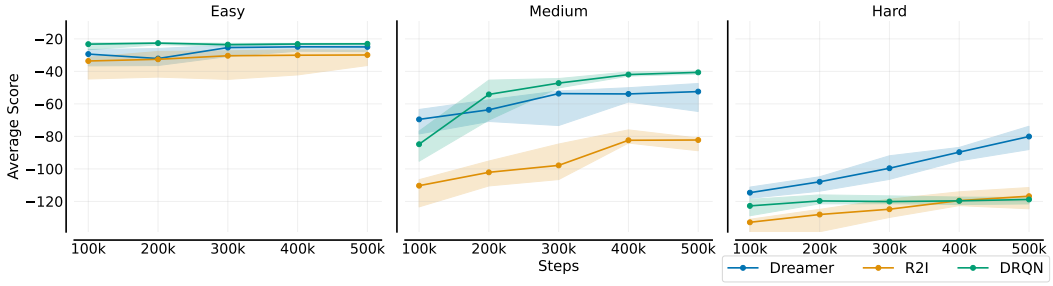


Figure 9: **Noisy Policy: Average Score** evaluation of the RL algorithms across difficulty levels. We evaluate the agent at each 100k environment step by averaging scores over 300 episode rollouts. Lines show IQM of the average scores across five seeds, with shaded regions indicating 95% confidence intervals computed using bootstrap sampling. Higher scores are better.

C.3 Waypoint Navigation

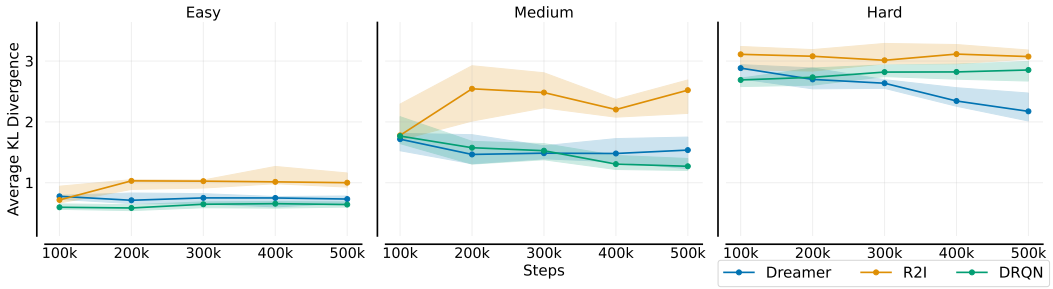


Figure 10: **Noisy Policy: KL divergence** between the predicted and true belief distributions across difficulty levels. At each 100k environment step, we evaluate the agent by averaging per-step KL divergence within each episode, then averaging this across the episode and finally averaging the episodic means across 300 episodes. Lines show IQM of the average scores across five seeds, with shaded regions indicating 95% confidence intervals computed using bootstrap sampling. Lower values are better.

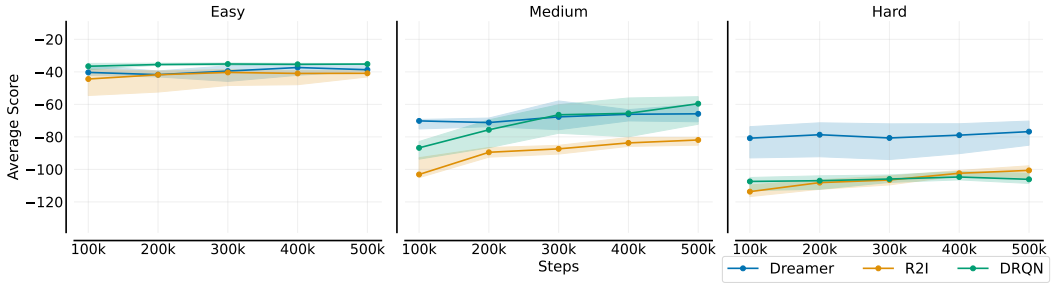


Figure 11: **Waypoint Policy: Average Score** evaluation of the RL algorithms across difficulty levels. We evaluate the agent at each 100k environment step by averaging scores over 300 episode rollouts. Lines show IQM of the average scores across five seeds, with shaded regions indicating 95% confidence intervals computed using bootstrap sampling. Higher scores are better.

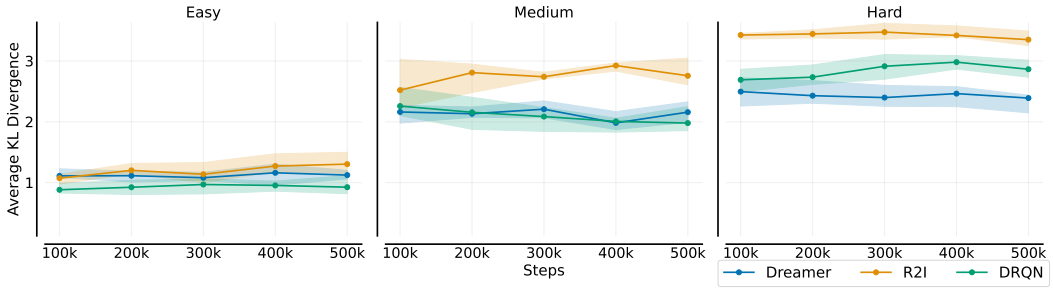


Figure 12: **Waypoint Policy: KL divergence** between predicted and true belief distributions across difficulty levels. At each 100k environment step, we evaluate the agent by averaging per-step KL divergence within each episode, then averaging these episodic means across 300 episodes. Lines show IQM of the average scores across five seeds, with shaded regions indicating 95% confidence intervals computed using bootstrap sampling. Lower values are better.