Why Generalization in RL is Difficult: Epistemic POMDPs and Implicit Partial Observability

Dibya Ghosh[*] UC Berkeley
Jad Rahme[*] Princeton University
Aviral Kumar UC Berkeley
Amy Zhang UC Berkeley, Facebook AI Research
Ryan P. Adams Princeton University
Sergey Levine UC Berkeley

Abstract

Generalization is a central challenge for the deployment of reinforcement learning (RL) systems in the real world. In this paper, we show that the sequential structure of the RL problem necessitates new approaches to generalization beyond the well-studied techniques used in supervised learning. While supervised learning methods can generalize effectively without explicitly accounting for epistemic uncertainty, we show that, perhaps surprisingly, this is not the case in RL. We show that generalization to unseen test conditions from a limited number of training conditions induces implicit partial observability, effectively turning even fully-observed MDPs into POMDPs. Informed by this observation, we recast the problem of generalization in RL as solving the induced partially observed Markov decision process, which we call the epistemic POMDP. We demonstrate the failure modes of algorithms that do not appropriately handle this partial observability, and suggest a simple ensemble-based technique for approximately solving the partially observed problem. Empirically, we demonstrate that our simple algorithm derived from the epistemic POMDP achieves significant gains in generalization over current methods on the Procgen benchmark suite.

1 Introduction

Generalization is a central challenge in machine learning. However, much of the research on reinforcement learning (RL) has been concerned with the problem of optimization: how to master a specific task through online or logged interaction. Generalization to new test-time contexts has received comparatively less attention, although several works have observed empirically [1, 2, 3, 4] that generalization to new situations poses a significant challenge to RL policies learned from a fixed training set of situations. In standard supervised learning, it is known that in the absence of distribution shift and with appropriate inductive biases, optimizing for performance on the training set (i.e., empirical risk minimization) translates into good generalization performance. It is tempting to suppose that the generalization challenges in RL can be solved in the same manner as empirical risk minimization in supervised learning: when provided a training set of contexts, learn the optimal policy within these contexts and then use that policy in new contexts at test-time.

Perhaps surprisingly, we show that such "empirical risk minimization" approaches can be sub-optimal for generalizing to new contexts in RL, even when these new contexts are drawn from the same distribution as the training contexts. As an anecdotal example of why this sub-optimality arises, imagine a robotic zookeeper for feeding otters that must be trained on some set of zoos. When placed in a new zoo, the robot must find and enter the otter enclosure. It can use one of two strategies: either

[*]Contributed equally. Correspond to: dibya@berkeley.edu, jrahme@math.princeton.edu

Figure 1: **Visualization of the robotic zookeeper example.** Standard RL algorithms learn the classifier strategy, since it is optimal in every training zoo, but this strategy is sub-optimal for generalization because peeking generalizes better than the classifier at test-time. This failure occurs due to the following disconnect: while the task is *fully-observed* since the image uniquely specifies the location of the otter habitat, to an agent that has limited training data, the location is *implicitly partially observed at test-time* because of the agent's epistemic uncertainty about the parameters of the image classifier.

peek through all the habitat windows looking for otters, which succeeds with 95% probability in all zoos, or to follow an image of a hand-drawn map of the zoo that unambiguously identifies the otter enclosure, which will succeed as long as the agent is able to successfully parse the image. In every training zoo, the otters can be found more reliably using the image of the map, and so an agent trained to seek the optimal policy in the training zoos would learn a classifier to predict the identity of the otter enclosure from the map, and enter the predicted enclosure. This classification strategy is optimal on the training environments because the agent can learn to perfectly classify the training zoo maps, but it is *sub-optimal* for generalization, because the learned classifier will never be able to perfectly classify every new zoo map at test-time. Note that this task is *not* partially observed, because the map provides full state information even for a memoryless policy. However, if the learned map classifier succeeds on anything less than 95% of new zoos at test-time, the strategy of peeking through the windows, although always sub-optimal in the training environments, turns out to be a more reliable strategy for finding the otter habitat in a *new* zoo, and results in higher expected returns at test-time.

Although with enough training zoos, the zookeeper can learn a policy by solving the map classification problem, to generalize optimally when given a limited number of zoos requires a more intricate policy that is not learned by standard RL methods. How can we more generally describe the set of behaviors needed for a policy to generalize from a finite number of training contexts in the RL setting? We make the observation that, even in fully-observable domains, the agent's epistemic uncertainty renders the environment *implicitly* partially observed at test-time. In the zookeeper example, although the hand-drawn map provides the exact location of the otter enclosure (and so the enclosure's location is technically fully observed), the agent cannot identify the true parameters of the map classifier from the small set of maps seen at training time, and so the location of the otters is implicitly obfuscated from the agent. We formalize this observation, and show that generalizing optimally at test-time corresponds to solving a partially-observed Markov decision process that we call an **epistemic POMDP**, which is induced by the agent's epistemic uncertainty about the test environment.

That uncertainty about MDP parameters can be modelled as a POMDP is well-studied in Bayesian RL when training and testing on a single task in an online setting, primarily in the context of exploration [5, 6, 7, 8]. However, as we will discuss, this POMDP interpretation has significant consequences for the generalization problem in RL, where an agent cannot collect more data online, and must instead learn a policy from a fixed set of training contexts that generalizes to new contexts at test-time. We show that standard RL methods that do not explicitly account for this implicit partial observability can be arbitrarily sub-optimal for test-time generalization in theory and in practice. The epistemic POMDP underscores the difficulty of the generalization problem in RL, as compared to supervised learning, and provides an avenue for understanding how we should approach generalization under the sequential nature and non-uniform reward structure of the RL setting. Maximizing expected return in an approximation of the epistemic POMDP emerges as a principled approach to learning policies that generalize well, and we propose LEEP, an algorithm that uses an ensemble of policies to approximately learn the Bayes-optimal policy for maximizing test-time performance.

The primary contribution of this paper is to use Bayesian RL techniques to reframe generalization in RL as the problem of solving a partially observed Markov decision process, which we call the *epistemic POMDP*. The epistemic POMDP highlights the difficulty of generalizing well in RL, as compared to supervised learning. We demonstrate the practical failure modes of standard RL methods, which do not reason about this partial observability, and show that maximizing test-time performance may require algorithms to explicitly consider the agent's epistemic uncertainty during training. Our

work highlights the importance of not only finding ways to help neural networks in RL generalize better, but also on learning policies that degrade gracefully when the underlying neural network eventually does fail to generalize. Empirically, we demonstrate that LEEP, which maximizes return in an approximation to the epistemic POMDP, achieves significant gains in test-time performance over standard RL methods on several ProcGen benchmark tasks.

2 Related Work

Many empirical studies have demonstrated the tendency of RL algorithms to overfit significantly to their training environments [1, 2, 3, 4], and the more general increased difficulty of learning policies that generalize in RL as compared to seemingly similar supervised learning problems [9, 10, 11, 12]. These empirical observations have led to a newfound interest in algorithms for generalization in RL, and the development of benchmark RL environments that focus on generalization to new contexts from a limited set of training contexts sharing a similar structure (state and action spaces) but possibly different dynamics and rewards [13, 14, 15, 16, 17].

Generalization in RL. Approaches for improving generalization in RL have fallen into two main categories: improving the ability of function approximators to generalize better with inductive biases, and incentivizing behaviors that are easier to generalize to unseen contexts. To improve the representations learned in RL, prior work has considered imitating environment dynamics [18, 19], seeking bisimulation relations [20, 21], and more generally, addressing representational challenges in the RL optimization process [22, 23]. In image-based domains, inductive biases imposed via neural network design have also been proposed to improve robustness to certain factors of variation in the state [24, 25, 26]. The challenges with generalization in RL that we will describe in this paper stem from the deficiencies of MDP objectives, and cannot be fully solved by choice of representations or functional inductive biases. In the latter category, one approach is domain randomization, varying environment parameters such as coefficients of friction or textures, to obtain behaviors that are effective across many candidate parameter settings [27, 28, 29, 30, 31]. Domain randomization sits within a class of methods that seek robust policies by injecting noise into the agent-environment loop, whether in the state [32], the action (e.g., via max-entropy RL) [14], or intermediary layers of a neural network policy (e.g., through information bottlenecks) [22, 33]. In doing so, these methods effectively introduce partial observability into the problem; while not necessarily equivalent to the partial observability that is induced by the epistemic POMDP, it may indicate why these methods generalize well empirically.

Bayesian RL: Our work recasts generalization in RL within the Bayesian RL framework, the problem of acting optimally under a belief distribution over MDPs (see Ghavamzadeh et al. [8] for a survey). Bayesian uncertainty has been studied in many sub-fields of RL [34, 35, 36, 37], the most prominent being for exploration and learning efficiently in the online RL setting. Bayes-optimal behavior in RL is often reduced to acting optimally in a POMDP, or equivalently, a belief-state MDP [6], of which our epistemic POMDP is a specific instantiation. Learning the Bayes-optimal policy exactly is intractable in all but the simplest problems [38, 39], and many works in Bayesian RL have studied relaxations that remain asymptotically optimal for learning, for example with value of perfect information [5, 40] or Thompson sampling [7, 41, 42]. Our main contribution is to revisit these classic ideas in the context of generalization for RL. We find that the POMDP interpretation of Bayesian RL [5, 6, 43] provides new insights on inadequacies of current algorithms used in practice, and explains why generalization in RL can be more challenging than in supervised learning. Being Bayesian in the generalization setting also requires new tools and algorithms beyond those classically studied in Bayesian RL, since test-time generalization is measured using regret over a *single* evaluation episode, instead of throughout an online training process. As a result, algorithms and policies that minimize short-term regret (i.e., are more exploitative) are preferred over traditional algorithms like Thompson sampling that explore thoroughly to ensure asymptotic optimality at the cost of short-term regret.

3 Problem Setup

We consider the problem of learning RL policies given a set of training contexts that generalize well to new unseen contexts. This problem can be formalized in a Markov decision process (MDP) where the agent does not have full access to the MDP at training time, but only particular initial states or conditions. Before we describe what this means, we must describe the MDP $\mathcal{M}$, which is given by a tuple $(\mathcal{S}, \mathcal{A}, r, T, \rho, \gamma)$, with state space $\mathcal{S}$, action space $\mathcal{A}$, Markovian transition

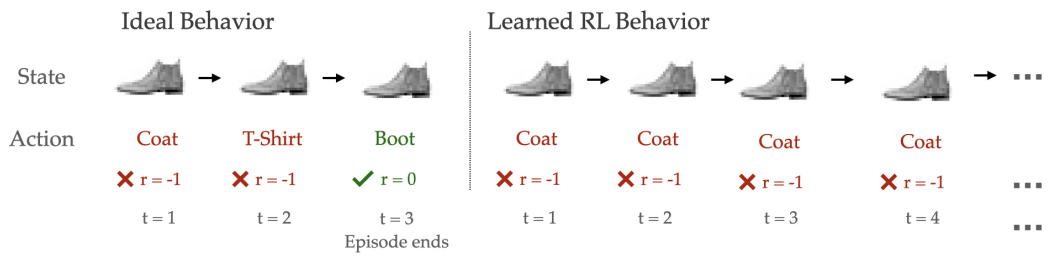

Figure 2: **Sequential Classification RL Problem.** In this task, an agent must keep guessing the label for an image until it gets it correct. To avoid low test return, policies should change actions if the label guessed was incorrect, but standard RL methods fail to do so, instead guessing the same incorrect label repeatedly.

function $T(s_{t+1}|s_t, a_t)$, bounded reward function $r(s_t, a_t)$, and initial state distribution $\rho(s_0)$. A policy π induces a discounted state distribution $d^\pi(s) = (1-\gamma)\mathbb{E}_\pi[\sum_{t\geq 0}\gamma^t 1(s_t = s)]$, and achieves return $J_{\mathcal{M}}(\pi) = \mathbb{E}_\pi[\sum_{t\geq 0}\gamma^t r(s_t, a_t)]$ in the MDP. Classical results establish that a deterministic Markovian policy π^* maximizes this objective amongst all history-dependent policies.

We focus on generalization in contextual MDPs where the agent is only trained on a subsample of contexts, and seeks to generalize well across unseen contexts. A contextual MDP is an MDP in which the state can be decomposed as $s_t = (c, s'_t)$, a context vector $c \in \mathcal{C}$ that remains constant throughout an episode, and a sub-state $s' \in \mathcal{S}'$ that may vary: $\mathcal{S} := \mathcal{C} \times \mathcal{S}'$. Each context vector corresponds to a different situation that the agent might be in, each with slightly different dynamics and rewards, but some shared structure across which an agent can generalize. During training, the agent is allowed to interact only within a sampled subset of contexts $\mathcal{C}_{\text{train}} \subset \mathcal{C}$. The generalization performance of the agent is measured by the return of the agent's policy in the full contextual MDP $J(\pi)$, corresponding to expected performance when placed in potentially new and unseen contexts. While our examples and experiments will be in contextual MDPs, our theoretical results also apply to other RL generalization settings where the full MDP cannot be inferred unambiguously from the data available during training, for example in offline reinforcement learning [44].

4 Warmup: A Sequential Classification RL Problem

We begin our study of generalization in RL with an example problem that is set up to be close to a supervised learning task where generalization is relatively well understood: image classification on the FashionMNIST dataset [45]. In this environment (visualized in Figure 2), an image from the dataset is sampled (the context) at the beginning of an episode and held fixed; the agent must identify the label of the image to complete the episode. If the agent guesses correctly, it receives a reward of 0 and the episode ends; if incorrect, it receives a reward of -1 and the episode continues, so it must attempt another guess for the *same* image at the next time step. This RL problem is near identical to supervised classification, the core distinction being that an agent may interact with the same image over several timesteps in an episode instead of only one attempt as in supervised learning. Note that since episodes may last longer than a single timestep, this problem is not a contextual bandit.

The optimal policy in both the one-step and sequential RL version of the problem deterministically outputs the correct label for the image, because the image fully determines the label (in other words, it is a fully observed MDP). However, this optimal strategy generally cannot be learned from a finite training set, since some generalization error is unavoidable. With a fixed training set, the strategy for generalizing in classification remains the

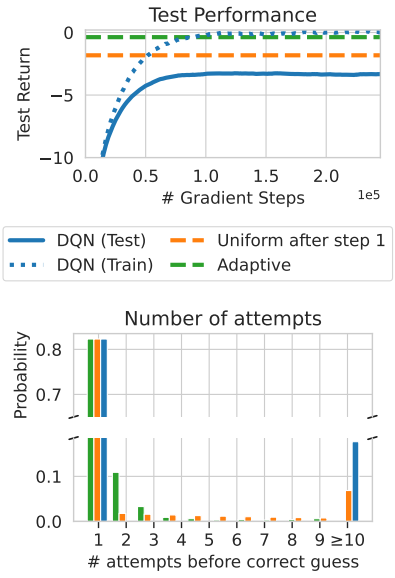

Figure 3: **DQN on RL FashionMNIST.** DQN achieves lower test performance than simple variants that leverage the structure of the RL problem.

same: deterministically choose the label the agent is most confident about. However, the RL setting introduces two new factors: the agent gets multiple tries at classifying the same image, and it knows if an attempted label is incorrect. To generalize best to new test images, an RL policy must leverage this additional structure, for example by trying many possible labels, or by changing actions if the previous guess was incorrect.

A standard RL algorithm, which estimates the optimal policy on the empirical MDP defined by the dataset of training images does not learn to leverage these factors, and instead learns behavior highly sub-optimal for generalization. We obtained a policy by running DQN [46] (experimental details in Appendix A.1), whose policy deterministically chooses the same label for the image at every timestep. Determinism is not specific to DQN, and is inevitable in any RL method that models the problem as an MDP because the optimal policy in the MDP is always deterministic and Markovian. The learned deterministic policy either guesses the correct label immediately, or guesses incorrectly and proceeds to make the same incorrect guess on every subsequent time-step. We compare performance in Figure 3 with a version of the agent that starts to guess randomly if incorrect on the first timestep, and a different agent that acts by process of elimination: first choosing the action it is most confident about, if incorrect, then the second, and so forth. Although all three versions have the same training performance, the learned RL policy generalizes more poorly than these alternative variants that exploit the sequential nature of the problem. In Section 5.2, we will see that this process-of-elimination is, in some sense, the optimal way to generalize for this task. This experiment reveals a tension: learning policies for generalization that rely on an MDP model fail, even though the underlying environment *is* an MDP. This failure holds in any MDP model with limited data, whether the empirical MDP or more sophisticated MDPs that use uncertainty estimates in their construction.

5 Modeling Generalization in RL as an Epistemic POMDP

To better understand test-time generalization in RL, we study the problem under a Bayesian perspective. We show that training on limited training contexts leads to an implicit partial observability at test-time that we describe using a formalism called the epistemic POMDP.

5.1 The Epistemic POMDP

In the Bayesian framework, when learning given a limited amount of evidence $\mathcal{D}$ from an MDP $\mathcal{M}$, we can use a prior distribution $\mathcal{P}(\mathcal{M})$ to construct a posterior belief distribution $\mathcal{P}(\mathcal{M}|\mathcal{D})$ over the identity of the MDP. For learning in a contextual MDP, $\mathcal{D}$ corresponds to the environment dynamics and reward in training contexts $\mathcal{C}_{\text{train}}$ that the agent can interact with, and the posterior belief distribution $\mathcal{P}(\mathcal{M}|\mathcal{D})$ models the agent's uncertainty about the behavior of the environment in contexts that it has not seen before (e.g. uncertainty about the label for a image from the test set in the RL image classification example).

Since the agent only has partial access to the MDP $\mathcal{M}$ during training, the agent does not know which MDP from the posterior distribution is the true environment, and must act at test-time under this uncertainty. Following a reduction common in Bayesian RL [6, 8], we model this test-time uncertainty using a partially observed MDP that we will call the **epistemic POMDP**. The epistemic POMDP is structured as follows: each new episode in the POMDP begins by sampling a single MDP $\mathcal{M} \sim \mathcal{P}(\mathcal{M}|\mathcal{D})$ from the posterior, and then the agent interacts with $\mathcal{M}$ until the episode ends in this MDP. The agent does not observe *which* MDP was sampled, and since the MDP remains fixed for the duration of the episode, this induces implicit partial observability. Effectively, each episode in the epistemic POMDP corresponds to acting in one of the possible environments that is consistent with the evidence that the agent is allowed access to at training time.

The epistemic POMDP is formally defined as the tuple $\mathcal{M}^{\text{po}} = (\mathcal{S}^{\text{po}}, \mathcal{O}^{\text{po}}, \mathcal{A}, T^{\text{po}}, r^{\text{po}}, \rho^{\text{po}}, \gamma)$. A state in this POMDP $s_t^{\text{po}} = (\mathcal{M}, s_t)$ contains the identity of the current MDP being acted in $\mathcal{M}$, and the current state in this MDP s_t; we write the state space as $\mathcal{S}^{\text{po}} = \mathbf{M} \times \mathcal{S}$, where $\mathbf{M}$ is the space of MDPs with support under the prior. The agent only gets to observe $o_t^{\text{po}} = s_t$, the state in the MDP ($\mathcal{O}^{\text{po}} = \mathcal{S}$). The initial state distribution is defined by the posterior distribution over MDPs: $\rho^{\text{po}}((\mathcal{M}, s_0)) = \mathcal{P}(\mathcal{M}|\mathcal{D})\rho_{\mathcal{M}}(s_0)$, and the transition and reward functions in the POMDP reflect the dynamics in the current MDP:

$$T^{\text{po}}((\mathcal{M}', s') \mid (\mathcal{M}, s), a) = \delta(\mathcal{M}' = \mathcal{M}) T_{\mathcal{M}}(s'|s, a) \qquad r^{\text{po}}((\mathcal{M}, s), a) = r_{\mathcal{M}}(s, a)\,. \tag{1}$$

What makes the epistemic POMDP a useful tool for understanding generalization in RL is that when the prior is well-specified, performance in the epistemic POMDP $\mathcal{M}^{\text{po}}$ corresponds exactly to the expected return of the agent at test-time.

Proposition 5.1. *If the true MDP $\mathcal{M}$ is sampled from $\mathcal{P}(\mathcal{M})$, and evidence $\mathcal{D}$ from $\mathcal{M}$ is provided to an algorithm during training, then the expected test-time return of π is equal to its performance in the epistemic POMDP $\mathcal{M}^{po}$.*

$$J_{\mathcal{M}^{po}}(\pi) = \mathbb{E}_{\mathcal{M}\sim\mathcal{P}(\mathcal{M})}[J_{\mathcal{M}}(\pi) \mid \mathcal{D}]. \tag{2}$$

In particular, the optimal policy in $\mathcal{M}^{po}$ is Bayes-optimal for generalization to the unknown MDP $\mathcal{M}$: it receives the highest expected test-time return amongst all possible policies.

The epistemic POMDP is based on well-understood concepts in Bayesian reinforcement learning, and Bayesian modeling more generally. However, in contrast to prior works on Bayesian RL, we are specifically concerned with settings where there is a training-test split, and performance is measured by a single test episode. While using Bayesian RL to accelerate exploration or minimize regret has been well-explored [8], we rather use the Bayesian lens specifically to understand generalization – a perspective that is distinct from prior work on Bayesian RL. Towards this goal, the equivalence between test-time return and expected return in the epistemic POMDP allows us to use performance in the POMDP as a proxy for understanding how well current RL methods can generalize.

5.2 Understanding Optimality in the Epistemic POMDP

We now study the structure of the epistemic POMDP, and use it to characterize properties of Bayes-optimal test-time behavior and the sub-optimality of alternative policy learning approaches. The majority of our results follow from well-known results about POMDPs, so we present them here informally, with formal statements and proofs in Appendix B. We begin by revisiting the sequential image classification problem, and describing the induced epistemic POMDP.

Example (Sequential Image Classification)**.** In the task from Section 4, the agent's uncertainty concerns how images are mapped to labels. Each MDP $\mathcal{M}$ in the posterior distribution corresponds to a different potential classifier $Y_{\mathcal{M}}$, where $Y_{\mathcal{M}}(x)$ denotes the label of x according to $\mathcal{M}$. Each episode in the epistemic POMDP requires the agent to guess the label for an image, where the label is determined by a randomly chosen classifier whose identity is unknown. Acting optimally in the epistemic POMDP requires strategies that work well in expectation over the distribution of labels that is induced by the posterior distribution $p(y|x, \mathcal{D}) = \mathbb{E}_{\mathcal{M}\sim\mathcal{P}(\mathcal{M}|\mathcal{D})}[1(Y_{\mathcal{M}}(x) = y)]$. A deterministic policy (as is learned by standard RL algorithms) is a high-risk strategy in the POMDP; it receives decent return when the guessed label matches the one from the sampled classifier, but exceedingly low return if the sampled classifier outputs a different label (-1 at every timestep). The Bayes-optimal strategy for maximizing expected test-time return corresponds to a process of elimination: first choose the most likely label $a = \arg\max p(y|x, \mathcal{D})$; if this is incorrect, eliminate it and choose the next-most likely, repeating until the correct label is finally chosen. Amongst memoryless policies, the optimal behavior is stochastic, sampling actions according to the distribution $\pi^*(a|x) \propto \sqrt{p(y|x, \mathcal{D})}$ (derivation in Appendix A.2).

The characteristics of the optimal policy in the epistemic POMDP for the image classification RL problem match well-established results that optimal POMDP policies are generally memory-based [47], and amongst memoryless policies, the optimal policy may be stochastic [48, 49]. Because of the equivalence between the epistemic POMDP and test-time behavior, these maxims are also true for Bayes-optimal behavior when maximizing test-time performance.

Remark 5.1. *The Bayes-optimal policy for maximizing test-time performance is in general non-Markovian. When restricted to Markovian policies, the Bayes-optimal policy is in general stochastic.*

The reason that Bayes-optimal behavior is often non-Markovian is that the experience collected thus far in the episode contains information about the identity of the MDP being acted in (which is hidden from the agent observation), and to maximize expected return, the agent must adapt its subsequent behavior to incorporate this new information. The fact that acting optimally at test-time formally requires adaptivity (or stochasticity for memoryless policies) highlights the difficulty of generalizing well in RL, and provides a new perspective for understanding the success various empirical studies have found in improving generalization performance using recurrent networks [50, 51] and stochastic regularization penalties [32, 14, 22, 33].

It is useful to understand to what degree the partial observability plays a role in determining Bayes-optimal behavior. When the partial observability is insignificant, the epistemic POMDP objective can coincide with a surrogate MDP approximation, and Bayes-optimal solutions can be attained with standard fully-observed RL algorithms. For example, if there is a policy that is simultaneously optimal in *every* MDP from the posterior, then an agent need not worry about the (hidden) identity of the MDP, and just follow this policy. Perhaps surprisingly, this kind of condition is difficult to relax: we show in Proposition B.1 that even if a policy is optimal in many (but not all) of the MDPs from the posterior, this seemingly "optimal" policy can generalize poorly at test-time.

Moreover, under partial observability, optimal policies for the MDPs in the posterior may differ substantially from Bayes-optimal behavior: in Proposition B.2, we show that the Bayes-optimal policy may take actions that are sub-optimal in *every* environment in the posterior. These results indicate the brittleness of learning policies based on optimizing return in an MDP model when the agent has not yet fully resolved the uncertainty about the true MDP parameters.

Remark 5.2 (Failure of MDP-Optimal Policies, Propositions B.1, B.2)**.** *The expected test-time return of policies that are learned by maximizing reward in any MDP from the posterior, as standard RL methods do, may be arbitrarily low compared to that of Bayes-optimal behavior.*

As Bayes-optimal memoryless policies are stochastic, one may wonder if simple strategies for inducing stochasticity, such as adding ϵ-greedy noise or entropy regularization, can alleviate the sub-optimality that arose with deterministic policies in the previous paragraph. In some cases, this may be true; one particularly interesting result is that in certain goal-reaching problems, entropy-regularized RL can be interpreted as optimizing an epistemic POMDP objective with a specific form of posterior distribution over reward functions (Proposition B.3) [52]. For the more general setting, we show in Proposition B.4 that entropy regularization and other general-purpose techniques can similarly catastrophically fail in epistemic POMDPs.

Remark 5.3 (Failure of Generic Stochasticity, Proposition B.4)**.** *The expected test-time return of policies learned with stochastic regularization techniques like maximum-entropy RL that are agnostic of the posterior $\mathcal{P}(\mathcal{M}|\mathcal{D})$ may be arbitrarily low compared to that of Bayes-optimal behavior.*

This failure happens because the degree of stochasticity used by the Bayes-optimal policy reflects the agent's epistemic uncertainty about the environment; since standard regularizations are agnostic to this uncertainty, the learned behaviors often do not reflect the appropriate level of stochasticity needed. A maze-solving agent acting Bayes-optimally, for example, may choose to act deterministically in mazes like those it has seen at training, and on others where it is less confident, rely on random exploration to exit the maze, inimitable behavior by regularization techniques agnostic to this uncertainty.

Our analysis of the epistemic POMDP highlights the difficulty of generalizing well in RL, in both the complexity of Bayes-optimal policies (Remark 5.1) and the deficiencies of our standard MDP-based RL algorithms (Remark 5.2, 5.3) . While MDP-based algorithms can serve as a useful starting point for acquiring generalizable skills, learning policies that perform well in new test-time scenarios may require more complex algorithms that attend to the epistemic POMDP structure that is implicitly induced by the by the agent's epistemic uncertainty.

6 Learning Policies that Generalize Well Using the Epistemic POMDP

When the epistemic POMDP $\mathcal{M}^{\text{po}}$ can be exactly obtained, we can learn RL policies that generalize well to the true (unknown) MDP $\mathcal{M}$ by learning an optimal policy in the POMDP. In this oracle setting, any POMDP-solving method will suffice, and design choices like policy function classes (e.g. recurrent vs Markovian policies) or agent representations (e.g. belief state vs PSRs) made based on the requirements of the specific domain. However, in practice, the epistemic POMDP can be challenging to approximate due to the difficulties of learning coherent MDP models and maintaining a posterior over such MDP models in high-dimensional domains.

In light of these challenges, we now focus on practical methods for learning generalizable policies when the exact posterior distribution (and therefore true epistemic POMDP) cannot be recovered exactly. We derive an algorithm for learning the optimal policy in the epistemic POMDP induced by an approximate posterior distribution $\hat{\mathcal{P}}(\mathcal{M}|\mathcal{D})$ with finite support. We use this to motivate LEEP, a simple ensemble-based algorithm for learning policies in the contextual MDP setting.

6.1 Policy Optimization in an Empirical Epistemic POMDP

Towards a tractable algorithm, we assume that instead of the true posterior $\mathcal{P}(\mathcal{M}|\mathcal{D})$, we only have access to an empirical posterior distribution $\hat{\mathcal{P}}(\mathcal{M}|\mathcal{D})$ defined by n MDP samples from the posterior distribution $\{\mathcal{M}_i\}_{i\in[n]}$. This empirical posterior distribution induces an empirical epistemic POMDP $\hat{\mathcal{M}}^{\text{po}}$; our ambition is to learn the optimal policy in this POMDP. Rather than directly learning this optimal policy as a generic POMDP solver might, we recognize that $\hat{\mathcal{M}}^{\text{po}}$ corresponds to a collection of n MDPs [2] and decompose the optimization problem to mimic this structure. We will learn n policies $\pi_1, \cdots, \pi_n$, each policy π_i in one of the MDPs $\mathcal{M}_i$ from the empirical posterior, and combine these policies together to recover a single policy π for the POMDP. Reducing the POMDP policy learning problem into a set of MDP policy learning problems can allow us to leverage the many recent advances in deep RL for scalably solving MDPs. The following theorem links the expected return of a policy π in the empirical epistemic POMDP $\hat{\mathcal{M}}^{\text{po}}$, in terms of the performance of the policies π_i on their respective MDPs $\mathcal{M}_i$, and provides a natural objective for learning policies in this decoupled manner.

Proposition 6.1. *Let $\pi, \pi_1, \cdots \pi_n$ be memoryless policies, and define $r_{\max} = \max_{i,s,a} |r_{\mathcal{M}_i}(s,a)|$. The expected return of π in $\hat{\mathcal{M}}^{po}$ is bounded below as:*

$$J_{\hat{\mathcal{M}}^{po}}(\pi) \geq \frac{1}{n}\sum_{i=1}^{n} J_{\mathcal{M}_i}(\pi_i) - \frac{\sqrt{2}r_{\max}}{(1-\gamma)^2 n}\sum_{i=1}^{n}\mathbb{E}_{s\sim d^{\pi_i}_{\mathcal{M}_i}}\left[\sqrt{D_{KL}\left(\pi_i(\cdot|s) \mid\mid \pi(\cdot|s)\right)}\right], \tag{3}$$

This proposition indicates that if the policies in the collection $\{\pi_i\}_{i\in[n]}$ all achieve high return in their respective MDPs (first term) and are imitable by a single policy π (second term), then π is guaranteed to achieve high return in the epistemic POMDP. In contrast, if the policies cannot be closely imitated by a single policy, this collection of policies may not be useful for learning in the epistemic POMDP using the lower bound. This means that it may not sufficient to naively optimize each policy π_i on its MDP $\mathcal{M}_i$ without any consideration to the other policies or MDPs, since the learned policies are likely to be different and difficult to jointly imitate. To be useful for the lower bound, each policy π_i should balance between maximizing performance on its MDP and minimizing its deviation from the other policies in the set. The following proposition shows that if the policies are trained jointly to ensure this balance, it in fact recovers the optimal policy in the empirical epistemic POMDP.

Proposition 6.2. *Let $f : \{\pi_i\}_{i\in[n]} \mapsto \pi$ be a function that maps n policies to a single policy satisfying $f(\pi, \cdots, \pi) = \pi$ for every policy π, and let α be a hyperparameter satisfying $\alpha \geq \frac{\sqrt{2}r_{max}}{(1-\gamma)^2 n}$. Then letting $\pi_1^*, \ldots \pi_n^*$ be the optimal solution to the following optimization problem:*

$$\{\pi_i^*\}_{i\in[n]} = \underset{\pi_1,\cdots,\pi_n}{\arg\max}\, \frac{1}{n}\sum_{i=1}^{n} J_{\mathcal{M}_i}(\pi_i) - \alpha\sum_{i=1}^{n}\mathbb{E}_{s\sim d^{\pi_i}_{\mathcal{M}_i}}\left[\sqrt{D_{KL}\left(\pi_i(\cdot|s) \mid\mid f(\{\pi_i\})(\cdot|s)\right)}\right], \tag{4}$$

the policy $\pi^* := f(\{\pi_i^*\}_{i\in[n]})$ is optimal for the empirical epistemic POMDP $\hat{\mathcal{M}}^{po}$.

6.2 A Practical Algorithm for Contextual MDPs: LEEP

Proposition 6.2 provides a foundation for a practical algorithm for learning policies when provided training contexts $\mathcal{C}_{\text{train}}$ from an unknown contextual MDP. In order to use the proposition in a practical algorithm, we must discuss two problems: how posterior samples $\mathcal{M}_i \sim \mathcal{P}(\mathcal{M}|\mathcal{D})$ can be approximated, and how the function f that combines policies should be chosen.

Approximating the posterior distribution: Rather than directly maintaining a posterior over transition dynamics and reward models, which is especially difficult with image-based observations, we can approximate samples from the posterior via a bootstrap sampling technique [53]. To sample a candidate MDP $\mathcal{M}_i$, we sample with replacement from the training contexts $\mathcal{C}_{\text{train}}$ to get a new set of contexts $\mathcal{C}^i_{\text{train}}$, and define $\mathcal{M}_i$ to be the empirical MDP on this subset of training contexts. Rolling out trials from the posterior sample $\mathcal{M}_i$ then corresponds to selecting a context at random from $\mathcal{C}^i_{\text{train}}$, and then rolling out that context. Crucially, note that $\mathcal{M}_i$ still corresponds to a *distribution* over contexts, not a single context, since our goal is to sample from the posterior entire contextual MDPs.

[2] Note that when the true environment is a contextual MDP, the sampled MDP $\mathcal{M}_i$ does not correspond to a single context within a contextual MDP — each MDP $\mathcal{M}_i$ is an *entire* contextual MDP with many contexts.

Algorithm 1 Linked Ensembles for the Epistemic POMDP (LEEP)

1: Receive training contexts $\mathcal{C}_{\text{train}}$, number of ensemble members n
2: Bootstrap sample training contexts to create $\mathcal{C}^1_{\text{train}}, \ldots \mathcal{C}^n_{\text{train}}$, where $\mathcal{C}^i_{\text{train}} \subset \mathcal{C}_{\text{train}}$.
3: Initialize n policies: $\pi_1, \ldots \pi_n$
4: **for** iteration $k = 1, 2, 3, \ldots$ **do**
5: **for** policy $i = 1, \ldots, n$ **do**
6: Collect environment samples in training contexts $\mathcal{C}^i_{\text{train}}$ using policy π_i
7: Take gradient steps wrt π_i on these samples with augmented RL loss:

$$\pi_i \leftarrow \pi_i - \eta \nabla_i (\mathcal{L}^{RL}(\pi_i) + \alpha \mathbb{E}_{s \sim \pi_i, \mathcal{C}^i_{\text{train}}} [D_{KL}(\pi_i(a|s) \| \max_j \pi_j(a|s))])$$

8: Return $\pi = \max_i \pi$: $\pi(a|s) = \frac{\max_i \pi_i(a|s)}{\sum_{a'} \max_i \pi_i(a'|s)}$.

Choosing a link function: The link function f in Proposition 6.2 that combines the set of policies together effectively serves as an inductive bias: since we are optimizing in an approximation to the true epistemic POMDP and policy optimization is not exact in practice, different choices can yield combined policies with different characteristics. Since optimal behavior in the epistemic POMDP must consider all actions, even those that are potentially sub-optimal in all MDPs in the posterior (as discussed in Section 5.2), we use an "optimistic" link function that does not dismiss any action that is considered by at least one of the policies, specifically $f(\{\pi_i\}_{i \in [n]}) = (\max_i \pi_i)(a|s) := \frac{\max \pi_i(a|s)}{\sum_{a'} \max \pi_i(a'|s)}$.

Algorithm: We learn a set of n policies $\{\pi_i\}_{i \in [n]}$, using a policy gradient algorithm to implement the update step. To update the parameters for π_i, we take gradient steps via the surrogate loss used for the policy gradient, augmented by a disagreement penalty between the policy and the combined policy $f(\{\pi_i\}_{i \in [n]})$ with a penalty parameter $\alpha > 0$, as in Equation 5:

$$\mathcal{L}(\pi_i) = \mathcal{L}^{RL}(\pi_i) + \alpha \mathbb{E}_{s \sim \pi_i, \mathcal{M}_i} [D_{KL}(\pi_i(a|s) \| \max_j \pi_j(a|s))]. \tag{5}$$

Combining these elements together leads to our method, LEEP, which we summarize in Algorithm 1. In our implementation, we use PPO for $\mathcal{L}^{RL}(\pi_i)$ [54]. In summary, LEEP bootstrap samples the training contexts to create overlapping sets of training contexts $\mathcal{C}^1_{\text{train}}, \ldots \mathcal{C}^n_{\text{train}}$. Every iteration, each policy π_i generates rollouts in training contexts chosen uniformly from its corresponding $\mathcal{C}^i_{\text{train}}$, and is then updated according to Equation 5, which both maximizes the expected reward and minimizes the disagreement penalty between each π_i and the combined policy $\pi = \max_j \pi_j$.

While this algorithm is structurally similar to algorithms for multi-task learning that train a separate policy for each context or group of contexts with a disagreement penalty [55, 56], the motivation and the interpretation of these approaches are completely distinct. In multi-task learning, the goal is to solve a given set of tasks, and these methods promote transfer via a loss that encourages the solutions to the tasks to be in agreement. In our setting, while we also receive a set of tasks (contexts), the goal is not to maximize performance on the training tasks, but rather to learn a policy that maximizes performance on unseen test tasks. The method also has a subtle but important distinction: each of our policies π_i acts on a sample from the contextual MDP posterior, *not* a single training context [55] or element from a disjoint partitioning [56]. This distinction is crucial, since our aim is not to make it easier to solve the training contexts, but the opposite: prevent the algorithm from overfitting to the individual contexts. Correspondingly, our experiments confirm that such multi-task learning approaches do not provide the same generalization benefits as our approach.

7 Experiments

The primary ambition of our empirical study is to test the hypothesis that policies that are learned through (approximations of) the epistemic POMDP do in fact attain better test-time performance than those learned by standard RL algorithms. We do so on the Procgen benchmark [16], a challenging suite of diverse tasks with image-based observations testing generalization to unseen contexts.

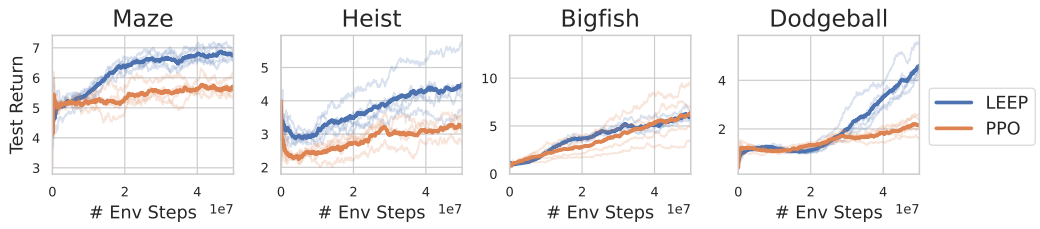

Figure 4: Test set return for LEEP and PPO throughout training in four Procgen environments (averaged across 5 random seeds). LEEP achieves higher test returns than PPO on three tasks (Maze, Heist and Dodgeball) and matches test return on Bigfish while having less variance across seeds.

1. Does LEEP derived from the epistemic POMDP lead to improved test-time performance over standard RL methods?
2. Can LEEP prevent overfitting when provided a limited number of training contexts?
3. How do different algorithmic components of LEEP affect test-time performance ?

The Procgen benchmark is a set of procedurally generated games, each with different generalization challenges. In each game, during training, the algorithm can interact with 200 training levels, before it is asked to generalize to the full distribution of levels. The agent receives a $64 \times 64 \times 3$ image observation, and must output one of 15 possible actions. We instantiate our method using an ensemble of $n = 4$ policies, a penalty parameter of $\alpha = 1$, and PPO [54] to train the individual policies (full implementation details in Appendix C).

We evaluate our method on four games in which prior work has found a large gap between training and test performance, and which we therefore conclude pose a significant generalization challenge [16, 23, 26]: Maze, Heist, BigFish, and Dodgeball. In Figure 4, we compare the test-time performance of the policies learned using our method to those learned by a PPO agent with entropy regularization. In three of these environments (Maze, Heist, and Dodgeball), our method outperforms PPO by a significant margin, and in all cases, we find that the generalization gap between training and test performance is lower for our method than PPO (Appendix D). To understand how LEEP behaves with fewer training contexts, we ran on the Maze task with only 50 levels (Figure 5 (top)); the test return of the PPO policy decreases through training, leading to final performance worse than the starting random policy, but our method avoids this degradation.

We perform an ablation study on the Maze and Heist environments (Maze in Figure 4, Heist in Appendix D) to rule out potential confounding causes for the improved generalization that our method displays on the Procgen benchmark tasks. First, to see if the performance benefit derives solely from the use of ensembles, we compare LEEP to a Bayesian model averaging strategy that trains an ensemble of policies without regularization ("Ensemble (no reg)"), and uses a mixture of these policies. This strategy does improve performance over the PPO policy, but does not match LEEP, indicating the usefulness of the regularization. Second, we compared to a version of LEEP that combines the ensemble policies together using the average $\frac{1}{n}\sum_{i=1}^{n}\pi_i(a|s)$ ("LEEP (avg)"). This link function achieves worse test-time performance than the optimistic version, which indicates that the inductive bias conferred by the $\max_i \pi_i$ link function is a useful component of the algorithm. Finally, we compare to Distral, a multi-task learning method with different motivations but similar structure to LEEP: this method helps accelerate learning on the provided training contexts (figures in Appendix D), but does not improve generalization performance as LEEP does.

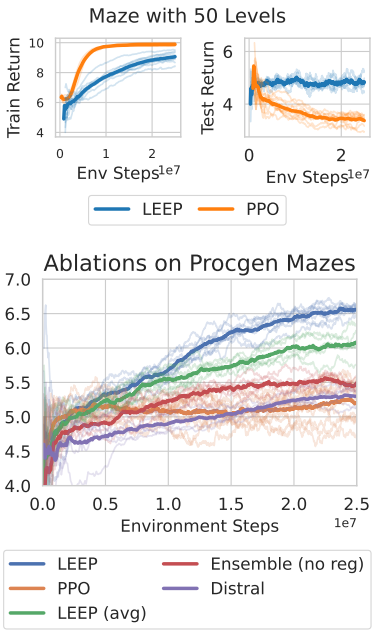

Figure 5: **(top)** Performance of LEEP and PPO with only 50 training levels on Maze. **(bottom)** Ablations of LEEP in Maze.

8 Discussion

It has often been observed experimentally that generalization in RL poses a significant challenge, but it has so far remained an open question as to whether the RL setting itself presents additional generalization challenges beyond those seen in supervised learning. In this paper, we answer this question in the affirmative, and show that, in contrast to supervised learning, generalization in RL results in a new type of problem that cannot be solved with standard MDP solution methods, due to partial observability induced by epistemic uncertainty. We call the resulting partially observed setting the epistemic POMDP, where uncertainty about the true underlying MDP results in a challenging partially observed problem. We present a practical approximate method that optimizes a bound for performance in an approximation of the epistemic POMDP, and show empirically that this approach, which we call LEEP, attains significant improvements in generalization over other RL methods that do not properly incorporate the agent's epistemic uncertainty into policy optimization. A limitation of this approach is that it optimizes a crude approximation to the epistemic POMDP with a small number of posterior samples, and may be challenging to scale to better approximations to the true objective. Developing algorithms that better model the epistemic POMDP and optimize policies within is an exciting avenue for future work, and we hope that this direction will lead to further improvements in generalization in RL.

Acknowledgements

This research was supported by an NSF graduate fellowship, the DARPA assured autonomy program, the NSF IIS-2007278 grant, a Princeton SEAS Innovation Grant and compute support from Google and Microsoft. We thank Benjamin Eysenbach, Xinyang Geng, and Justin Fu as well as members of the Princeton Laboratory for Intelligent Probabilistic Systems for helpful discussions and feedback.

References

[1] Jesse Farebrother, Marlos C. Machado, and Michael H. Bowling. Generalization and regularization in dqn. *ArXiv*, abs/1810.00123, 2018.

[2] C. Zhang, Oriol Vinyals, R. Munos, and S. Bengio. A study on overfitting in deep reinforcement learning. *ArXiv*, abs/1804.06893, 2018.

[3] Niels Justesen, R. Torrado, Philip Bontrager, Ahmed Khalifa, J. Togelius, and S. Risi. Illuminating generalization in deep reinforcement learning through procedural level generation. *arXiv: Learning*, 2018.

[4] Xingyou Song, Yiding Jiang, Stephen Tu, Yilun Du, and Behnam Neyshabur. Observational overfitting in reinforcement learning. *ArXiv*, abs/1912.02975, 2020.

[5] R. Dearden, N. Friedman, and Stuart J. Russell. Bayesian q-learning. In *AAAI/IAAI*, 1998.

[6] M. Duff and A. Barto. Optimal learning: computational procedures for bayes-adaptive markov decision processes. 2002.

[7] M. Strens. A bayesian framework for reinforcement learning. In *ICML*, 2000.

[8] M. Ghavamzadeh, Shie Mannor, Joelle Pineau, and Aviv Tamar. Bayesian reinforcement learning: A survey. *Found. Trends Mach. Learn.*, 8:359–483, 2015.

[9] A. Zhang, Yuxin Wu, and Joelle Pineau. Natural environment benchmarks for reinforcement learning. *ArXiv*, abs/1811.06032, 2018.

[10] A. Zhang, Nicolas Ballas, and Joelle Pineau. A dissection of overfitting and generalization in continuous reinforcement learning. *ArXiv*, abs/1806.07937, 2018.

[11] S. Whiteson, B. Tanner, Matthew E. Taylor, and P. Stone. Protecting against evaluation overfitting in empirical reinforcement learning. *2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pages 120–127, 2011.

[12] Zhuang Liu, Xuanlin Li, Bingyi Kang, and Trevor Darrell. Regularization matters in policy optimization – an empirical study on continuous control. *arXiv: Learning*, 2020.

[13] Alex Nichol, Vicki Pfau, Christopher Hesse, Oleg Klimov, and John Schulman. Gotta learn fast: A new benchmark for generalization in rl. *ArXiv*, abs/1804.03720, 2018.

[14] K. Cobbe, Oleg Klimov, Christopher Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *ICML*, 2019.

[15] Heinrich Kuttler, Nantas Nardelli, Alexander H. Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The nethack learning environment. *ArXiv*, abs/2006.13760, 2020.

[16] K. Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. *ArXiv*, abs/1912.01588, 2020.

[17] Austin Stone, Oscar Ramirez, K. Konolige, and Rico Jonschkowski. The distracting control suite - a challenging benchmark for reinforcement learning from pixels. *ArXiv*, abs/2101.02722, 2021.

[18] Max Jaderberg, V. Mnih, Wojciech Czarnecki, Tom Schaul, Joel Z. Leibo, D. Silver, and K. Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *ArXiv*, abs/1611.05397, 2017.

[19] Adam Stooke, Kimin Lee, P. Abbeel, and M. Laskin. Decoupling representation learning from reinforcement learning. *ArXiv*, abs/2009.08319, 2020.

[20] A. Zhang, Rowan McAllister, R. Calandra, Y. Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. *ArXiv*, abs/2006.10742, 2020.

[21] Rishabh Agarwal, Marlos C. Machado, P. S. Castro, and Marc G. Bellemare. Contrastive behavioral similarity embeddings for generalization in reinforcement learning. *ArXiv*, abs/2101.05265, 2021.

[22] Maximilian Igl, K. Ciosek, Yingzhen Li, Sebastian Tschiatschek, C. Zhang, Sam Devlin, and Katja Hofmann. Generalization in reinforcement learning with selective noise injection and information bottleneck. In *NeurIPS*, 2019.

[23] Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. *ArXiv*, abs/2010.03934, 2020.

[24] Kimin Lee, Kibok Lee, Jinwoo Shin, and H. Lee. Network randomization: A simple technique for generalization in deep reinforcement learning. In *ICLR*, 2020.

[25] Ilya Kostrikov, Denis Yarats, and R. Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *ArXiv*, abs/2004.13649, 2020.

[26] Roberta Raileanu, Maxwell Goldstein, Denis Yarats, Ilya Kostrikov, and R. Fergus. Automatic data augmentation for generalization in deep reinforcement learning. *ArXiv*, abs/2006.12862, 2020.

[27] Fereshteh Sadeghi and Sergey Levine. (cad)2rl: Real single-image flight without a single real image. *ArXiv*, abs/1611.04201, 2017.

[28] Joshua Tobin, Rachel Fong, Alex Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017.

[29] A. Rajeswaran, Sarvjeet Ghotra, Sergey Levine, and Balaraman Ravindran. Epopt: Learning robust neural network policies using model ensembles. *ArXiv*, abs/1610.01283, 2017.

[30] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8, May 2018.

[31] Katie Kang, Suneel Belkhale, Gregory Kahn, Pieter Abbeel, and Sergey Levine. Generalization through simulation: Integrating simulated and real data into deep reinforcement learning for vision-based autonomous flight. In *2019 international conference on robotics and automation (ICRA)*, pages 6008–6014. IEEE, 2019.

[32] F. Stulp, E. Theodorou, J. Buchli, and S. Schaal. Learning to grasp under uncertainty. *2011 IEEE International Conference on Robotics and Automation*, pages 5703–5708, 2011.

[33] X. Lu, Kimin Lee, P. Abbeel, and Stas Tiomkin. Dynamics generalization via information bottleneck in deep reinforcement learning. *ArXiv*, abs/2008.00614, 2020.

[34] Deepak Ramachandran and E. Amir. Bayesian inverse reinforcement learning. In *IJCAI*, 2007.

[35] A. Lazaric and M. Ghavamzadeh. Bayesian multi-task reinforcement learning. In *ICML*, 2010.

[36] Wonseok Jeon, Seokin Seo, and Kee-Eung Kim. A bayesian approach to generative adversarial imitation learning. In *NeurIPS*, 2018.

[37] Luisa M. Zintgraf, K. Shiarlis, Maximilian Igl, Sebastian Schulze, Y. Gal, Katja Hofmann, and S. Whiteson. Varibad: A very good method for bayes-adaptive deep rl via meta-learning. *ArXiv*, abs/1910.08348, 2020.

[38] R. Weber. On the gittins index for multiarmed bandits. *Annals of Applied Probability*, 2:1024–1033, 1992.

[39] P. Poupart, N. Vlassis, J. Hoey, and K. Regan. An analytic solution to discrete bayesian reinforcement learning. *Proceedings of the 23rd international conference on Machine learning*, 2006.

[40] R. Dearden, N. Friedman, and D. Andre. Model based bayesian exploration. In *UAI*, 1999.

[41] Ian Osband, Daniel Russo, and Benjamin Van Roy. (more) efficient reinforcement learning via posterior sampling. In *NIPS*, 2013.

[42] Daniel Russo and Benjamin Van Roy. Learning to optimize via posterior sampling. *Math. Oper. Res.*, 39:1221–1243, 2014.

[43] Stephane Ross, Brahim Chaib-draa, and Joelle Pineau. Bayes-adaptive pomdps. In *NIPS*, pages 1225–1232, 2007.

[44] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[45] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[46] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

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

[48] Satinder P. Singh, Tommi S. Jaakkola, and Michael I. Jordan. Learning without state-estimation in partially observable markovian decision processes. In *Proceedings of the Eleventh International Conference on International Conference on Machine Learning*, page 284–292, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[49] Guido Montúfar, K. Zahedi, and N. Ay. Geometry and determinism of optimal stationary control in partially observable markov decision processes. *ArXiv*, abs/1503.07206, 2015.

[50] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[51] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. *CoRR*, abs/1710.06537, 2017.

[52] Benjamin Eysenbach and Sergey Levine. If maxent rl is the answer, what is the question? *arXiv preprint arXiv:1910.01913*, 2019.

[53] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[54] John Schulman, F. Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017.

[55] Yee Teh, Victor Bapst, Wojciech M. Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[56] Dibya Ghosh, Avi Singh, Aravind Rajeswaran, Vikash Kumar, and Sergey Levine. Divide-and-conquer reinforcement learning. In *International Conference on Learning Representations*, 2018.

[57] Scott Fujimoto, H. V. Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. *ArXiv*, abs/1802.09477, 2018.

[58] Ilya Kostrikov. Pytorch implementations of reinforcement learning algorithms. https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail, 2018.

[59] Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *In Proc. 19th International Conference on Machine Learning*. Citeseer, 2002.

A FashionMNIST Classification

A.1 Implementation details

Environment: The RL image classification environment consists of a dataset of labelled images. At the beginning of each episode, a new image and its corresponding label are chosen from the dataset, and held fixed for the entire episode. Each time-step, the agent must pick an action corresponding to one of the labels. If the picked label is correct, the agent gets a reward of $r = 0$, and the episode ends, and if the picked label is incorrect, then the agent gets a reward of $r = -1$, and the episode continues to the next time-step (where it must guess another label for the *same* image). The total return for a trajectory corresponds to the number of incorrect guesses the agent makes for the image. We enforce a time-limit of 20 timesteps in the environment to prevent infinite-length trajectories of incorrect guessing.

We train the agent on a dataset of 10000 FashionMNIST images subsampled from the training set, and test on the FashionMNIST test dataset. Note that this task is very similar, but not exactly equivalent to maximizing predictive accuracy for supervised classification: if the episode ended regardless of whether or not the agent was correct, then it would correspond exactly to classification.

Algorithm: We train a DQN agent on the training environment using the min-Q update rule from TD3 [57]. The Q-function architecture is a convolutional neural network (CNN) with the architecture from Kostrikov et al [58]. To ensure that the agent does not suffer from poor exploration during training, the replay buffer is pre-populated with one copy of every possible transition in the training environment (that is, where every action is taken for every image in the training dataset). The variant labelled "Uniform after step 1" in Figure 3 follows the DQN policy for the first time-step, and if this was incorrect, then at all subsequent time-steps, takes a random action uniformly amongst the 10 labels. For the variant labelled "Adaptive", we train a classifier $p_\theta(y|x)$ on the training dataset of images with the same architecture as the DQN agent. The adaptive agent follows a process-of-elimination strategy; formally, the action taken by the adaptive agent at time-step t is given by $\arg\max_{a \notin \{a_1, \ldots, a_{t-1}\}} p_\theta(y = a|x)$.

A.2 Derivation of Bayes-optimal policies

In the epistemic POMDP for the RL image classification problem, each episode, an image $x \in \mathcal{X}$ is sampled randomly from the dataset, and a label $y \in \mathcal{Y}$ sampled randomly for this image from the distribution $p(y|x, \mathcal{D})$. This label is *held fixed* for the entire episode. For notation, let $Y = \{1, \ldots, d\}$, so that a label distribution $p(y|x)$ can be written as a vector in the probability simplex on $\mathbb{R}^d$. We emphasize two settings: $\gamma = 0$ (the supervised learning setting), and $\gamma = 1$ (an RL setting), where the expected return of an agent is the average number of incorrect guesses made.

A.2.1 Memory-based policy

Since the optimal memory-based policy in a POMDP is deterministic [47], we restrict ourselves to analyzing the performance of deterministic memory-based policies. In the following we will narrow the search space even further.

Since the episode ends after the agent correctly classifies an image and the reward structure incentives the agent to solve the task as quickly as possible, an agent acting optimally will never repeat the same action twice. Indeed, the agent will not have the opportunity to repeat the right action twice because the episode would have ended after the first time it tried it. Furthermore, trying a wrong action twice is also not optimal as in incurs addition negative reward. Therefore, we can limit our search space to policies that try each action once. These policies differ by the ordering in which they try each one of these d labels.

At the beginning of every episode, a image x is sampled uniformly at random among all training images and its true label y (during that episode) is sampled from $p(y|x, \mathcal{D})$. Let π be policy that tries each of the d actions exactly once in its first d trials. Let T_y^π denotes the time when policy π tries action y. Note that $(T_y^\pi)_{y \in \mathcal{Y}}$ is a permutation. When the label chosen is y, the cumulative reward of

π for that episode is given by $r = \frac{\gamma^{T_y^\pi}-1}{1-\gamma}$ and the expected cumulative reward (across episodes) is given by:

$$J(\pi) := \sum_{y\in\mathcal{Y}} p(y|x,\mathcal{D})\frac{\gamma^{T_y^\pi}-1}{1-\gamma} = \frac{1}{1-\gamma}\left(\sum_{y\in\mathcal{Y}} p(y|x,\mathcal{D})\gamma^{T_y^\pi} - 1\right) \tag{6}$$

From that expression, we see that in order to maximize its expected cumulative reward, a policy π has to maximize $\sum_{y\in\mathcal{Y}} p(y|x,\mathcal{D})\gamma^{T_y^\pi}$ which can be interpreted as the dot product of the vector $[p(y|x,\mathcal{D})]_{y\in\mathcal{Y}}$ and $[\gamma^{T_y^\pi}]_{y\in\mathcal{Y}}$. By the rearrangement inequality, we know that this dot product is maximized when the components of the vectors are arranged in the same ordering.

If we denote by $y_{(1)},\dots y_{(d)}$ be the labels sorted in order of probability under the belief distribution: $p(y_{(1)}|x,\mathcal{D}) \geq p(y_{(2)}|x,\mathcal{D}) \geq \dots \geq p(y_{(d)}|x,\mathcal{D})$. Since $0<\gamma<1$ the rearrangement inequality implies that the expected return is maximized when $T^\pi_{y_{(t)}} = t$. This corresponds to a policy that tries the labels sequentially from the most likely to the least likely.

A.2.2 Memoryless policy

In this section, we will derive the optimal memoryless policy. Consider a memoryless policy that takes actions according to the distribution $\pi(\cdot|x)$ for the image x. When the true label is y for the image x, the number of incorrect guesses is distributed as $\text{Geom}(p = \pi(y|x))$.

When the agent guesses correctly the label y at the $t-$th guess then the cumulative reward is given by $r = -\frac{1-\gamma^t}{1-\gamma}$. This happens with probability $(1-\pi(y \mid x))^t \times \pi(y \mid x)$. The expected return for policy π evaluated on image x is then given by:

$$\begin{aligned} J(\pi|x) &= -\sum_{y\in\mathcal{Y}}\sum_{t=0}^{\infty}(1-\pi(y|x))^t\pi(y|x)\frac{1-\gamma^t}{1-\gamma}p(y|x,\mathcal{D}) \\ &= \sum_{y\in\mathcal{Y}} p(y|x,\mathcal{D})\frac{\pi(y \mid x)-1}{1-\gamma(1-\pi(y|x))} \end{aligned} \tag{7}$$

When $\gamma = 0$ (supervised learning problem), $J(\pi) = \sum_{y\in\mathcal{Y}} p(y|x,\mathcal{D})\left(\pi(y \mid x)-1\right)$ is a linear function of π and as expected, the optimal policy is to deterministically choose the label with the highest probability:$\pi^*(y \mid x) = 1\left[y = \arg\max_{y\in\mathcal{Y}} p(y|x,\mathcal{D})\right]$.

When $\gamma > 0$, the optimal policy is the solution to a constrained optimization problem that can be solved with Lagrange multipliers. When $\gamma = 1$, the optimal policy can be written explicitly as:

$$\pi^*(y \mid x) = \frac{1}{\lambda}\sqrt{p(y|x,\mathcal{D})} \tag{8}$$

where λ is a normalization constant.

B Theoretical Results

Proposition 5.1. *If the true MDP $\mathcal{M}$ is sampled from $\mathcal{P}(\mathcal{M})$, and evidence $\mathcal{D}$ from $\mathcal{M}$ is provided to an algorithm during training, then the expected test-time return of π is equal to its performance in the epistemic POMDP $\mathcal{M}^{po}$.*

$$J_{\mathcal{M}^{po}}(\pi) = \mathbb{E}_{\mathcal{M}\sim\mathcal{P}(\mathcal{M})}[J_{\mathcal{M}}(\pi) \mid \mathcal{D}]. \tag{2}$$

In particular, the optimal policy in $\mathcal{M}^{po}$ is Bayes-optimal for generalization to the unknown MDP $\mathcal{M}$: it receives the highest expected test-time return amongst all possible policies.

Proof. This proposition follows directly from the definition of the epistemic POMDP. If the MDP $\mathcal{M}$ is sampled from $\mathcal{P}(\mathcal{M})$ and $\mathcal{D}$ is witnessed, then the posterior distribution over MDPs is given by $\mathcal{P}(\mathcal{M}|\mathcal{D})$, and the expected test-time return of π given the evidence is

$$\mathbb{E}_{\mathcal{M}\sim\mathcal{P}(\mathcal{M})}[J_{\mathcal{M}}(\pi)|\mathcal{D}] := \mathbb{E}_{\mathcal{M}\sim\mathcal{P}(\mathcal{M}|\mathcal{D})}[J_{\mathcal{M}}(\pi)].$$

In the epistemic POMDP, where an episode corresponds to randomly sampling an MDP from $\mathcal{P}(\mathcal{M}|\mathcal{D})$, and a single episode being evaluated in this MDP, the expected return can be expressed identically:

$$J_{\mathcal{M}^{\text{po}}}(\pi) := \mathbb{E}_{\mathcal{M}\sim\mathcal{P}(\mathcal{M}|\mathcal{D})}[\mathbb{E}_{\pi,\mathcal{M}}[\sum_{i=0}^{\infty}\gamma^t r(s_t,a_t)]] = \mathbb{E}_{\mathcal{M}\sim\mathcal{P}(\mathcal{M}|\mathcal{D})}[J_{\mathcal{M}}(\pi)].$$

□

B.1 Optimal MDP Policies can be Arbitrarily Suboptimal

Proposition B.1. *Let $\epsilon > 0$. There exists posterior distributions $\mathcal{P}(\mathcal{M}|\mathcal{D})$ where a deterministic Markov policy π is optimal with probability at least $1-\epsilon$,*

$$P_{\mathcal{M}\sim\mathcal{P}(\mathcal{M}|\mathcal{D})}\left(\pi \in \arg\max_{\pi'} J_{\mathcal{M}}(\pi')\right) \geq 1-\epsilon, \tag{9}$$

but is outperformed by a uniformly random policy in the epistemic POMDP: $J_{\mathcal{M}^{po}}(\pi) < J_{\mathcal{M}^{po}}(\pi_{unif})$.

Proof. Consider two deterministic MDPs, $\mathcal{M}_A$, and $\mathcal{M}_B$ that both have two states and two actions: "stay" and "switch". In both MDPs, the reward for the "stay" action is always zero. In $\mathcal{M}_A$ the reward for "switch" is always 1, while in $\mathcal{M}_B$ the reward for "switch" is $-c$ for $c > 0$. The probability of being in $\mathcal{M}_B$ is ϵ while the probability of being in $\mathcal{M}_A$ is $1-\epsilon$. Clearly, the policy "always switch" is optimal in $\mathcal{M}_A$ and so is ϵ-optimal under the distribution on MDPs. The expected discounted reward of the "always switch" policy is:

$$J(\pi_{\text{always switch}}) = (1-\epsilon)\frac{1}{1-\gamma} - \epsilon\frac{c}{1-\gamma} = \frac{1}{1-\gamma}(1-(c+1)\epsilon)\,. \tag{10}$$

On the other hand, we can consider a policy which selects actions uniformly at random. In this case, the expected cumulative reward is

$$J(\pi_{\text{random}}) = (1-\epsilon)\frac{1}{2}\frac{1}{1-\gamma} - \epsilon\frac{c}{2}\frac{1}{1-\gamma} = \frac{1}{2}\frac{1}{1-\gamma}(1-(c+1)\epsilon) = \frac{1}{2}J(\pi_{\text{always switch}})\,. \tag{11}$$

Thus for any ϵ we can find a $c > \frac{1}{\epsilon} - 1$ such that both policies have negative expected rewards and we prefer the random policy for being half as negative. □

B.2 Bayes-optimal Policies May Take Suboptimal Actions Everywhere

We formalize the remark that optimal policies for the MDPs in the posterior distribution may be poor guides for determining what the Bayes-optimal behavior is in the epistemic POMDP. The following proposition shows that there are epistemic POMDPs where the support of actions taken by the MDP-optimal policies is disjoint from the actions taken by the Bayes-optimal policy, so no method can "combine" the optimal policies from each MDP in the posterior to create Bayes-optimal behavior.

Proposition B.2. *There exist posterior distributions $\mathcal{P}(\mathcal{M}|\mathcal{D})$ where the support of the Bayes-optimal memoryless policy $\pi^{*po}(a|s)$ is disjoint with that of the optimal policies in each MDP in the posterior. Formally, writing $\text{supp}(\pi(a|s)) = \{a \in \mathcal{A} : \pi(a|s) > 0\}$, then $\forall\mathcal{M}$ with $\mathcal{P}(\mathcal{M}|\mathcal{D}) > 0$ and $\forall s$:*

$$\text{supp}(\pi^{*po}(a|s)) \cap \text{supp}(\pi^*_{\mathcal{M}}(a|s)) = \emptyset$$

Proof. The proof is a simple modification of the construction in Proposition 5.1. Consider two deterministic MDPs, $\mathcal{M}_A$, and $\mathcal{M}_B$ with equal support under the posterior, where both have two states and three actions: "stay", "switch 1", and "switch 2". In both MDPs, the reward for the "stay" action is always zero. In $\mathcal{M}_A$ the reward for "switch" is always 1, while in $\mathcal{M}_B$ the reward for "switch" is -2. The reward structure for "switch 2" is flipped: in $\mathcal{M}_A$, the reward for "switch 2" is -2, and in $\mathcal{M}_B$, the reward is 1. Then, the policy "always switch" is optimal in $\mathcal{M}_A$, and the policy "always switch 2" is optimal in $\mathcal{M}_B$. However, any memoryless policy that takes either of these actions receives negative reward in the epistemic POMDP, and is dominated by the Bayes-optimal memoryless policy "always stay", which achieves 0 reward. □

B.3 MaxEnt RL is Optimal for a Choice of Prior

We describe a special case of the construction of Eysenbach and Levine [52], which shows that maximum-entropy RL in a bandit problem recovers the Bayes-optimal POMDP policy in an epistemic POMDP similar to that described in the RL image classification task.

Consider the family of MDPs $\{\mathcal{M}_k\}_{k\in[n]}$ each with one state and n actions, where taking action k in MDP $\mathcal{M}_k$ yields zero reward and the episode ends, and taking any other action yields reward -1 and the episode continues. Effectively, $\mathcal{M}_k$ corresponds to a first-exit problem with "goal action" k. Note that this MDP structure is exactly what we have for the RL image classification task for a single image. Also consider the surrogate bandit MDP $\hat{\mathcal{M}}$, also with one state and n actions, but in which taking action k yields reward r_k with immediate episode termination. The following proposition shows that running max-ent RL in $\hat{\mathcal{M}}$ recovers the optimal memoryless policy in a particular epistemic POMDP supported on $\{\mathcal{M}_k\}_{k\in[n]}$.

Proposition B.3. *Let $\pi^* = \arg\max_{\pi\in\Pi} J_{\hat{\mathcal{M}}}(\pi) + \mathcal{H}(\pi)$ be the max-ent solution in the surrogate bandit MDP $\hat{\mathcal{M}}$. Define the distribution $\mathcal{P}(\mathcal{M}|\mathcal{D})$ on $\{\mathcal{M}_k\}_{k\in[n]}$ as $\mathcal{P}(\mathcal{M}_k|\mathcal{D}) = \frac{\exp(2r_k)}{\sum_j \exp(2r_j)}$. Then, π is the optimal memoryless policy in the epistemic POMDP $\mathcal{M}^{po}$ defined by $\mathcal{P}(\mathcal{M}|\mathcal{D})$.*

Proof. See Eysenbach and Levine [52, Lemma 4.1]. The optimal policy π^* is given by $\pi^*(a = k) = \frac{\exp(r_k)}{\sum_j \exp(r_j)}$. We know from Appendix A.2.2 that this policy is optimal for epistemic POMDP $\mathcal{M}^{\text{po}}$ when $\gamma = 1$. □

If allowing time-varying reward functions, this construction can be extended beyond "goal-action taking" epistemic POMDPs to the more general "goal-state reaching" setting in an MDP, where the agent seeks to reach a specific goal state, but the identity of the goal state hidden from the agent [52, Lemma 4.2].

B.4 Failure of MaxEnt RL and Uncertainty-Agnostic Regularizations

We formalize the remark made in the main text that while the Bayes-optimal memoryless policy is stochastic, methods that promote stochasticity in an uncertainty-agnostic manner can fail catastrophically. We begin by explaining the significance of this result: it is well-known that stochastic policies can be arbitrarily sub-optimal in a single MDP, and can be outperformed by deterministic policies. The result we describe is more subtle than this: there are epistemic POMDPs where any attempt at being stochastic in an uncertainty-agnostic manner is sub-optimal, and *also* any attempt at acting completely deterministically is also sub-optimal. Rather, the characteristic of Bayes-optimal behavior is to be stochastic in *some* states (where it has high uncertainty), and not stochastic in others, and a useful stochastic regularization method must modulate the level of stochasticity to calibrate with regions where it has high epistemic uncertainty.

Proposition B.4. *Let $\alpha > 0, c > 0$. There exist posterior distributions $\mathcal{P}(\mathcal{M}|\mathcal{D})$, where the Bayes-optimal memoryless policy π^{*po} is stochastic. However, every memoryless policy π_s that is "everywhere-stochastic", in that $\forall s \in \mathcal{S} : \mathcal{H}(\pi_s(a|s)) > \alpha$, can have performance arbitrarily close to the uniformly random policy:*

$$\frac{J(\pi_s) - J(\pi_{unif})}{J(\pi^{*po}) - J(\pi_{unif})} < c$$

Proof. Consider two binary tree MDP with n levels, $\mathcal{M}_1$ and $\mathcal{M}_2$. A binary tree MDP, visualized in Figure 6, has n levels, where level k has 2^k states. On any level $k < n$, the agent can take a "left" action or a "right" action, which transitions to the corresponding state in the next level. On the final level, if the state corresponds to the terminal state (in green), then the agent receives a reward of 1, and the episode exits, and otherwise a reward of 0, and the agent returns to the top of the binary tree. The two binary tree MDPs $\mathcal{M}_1$ and $\mathcal{M}_2$ are identical except for the final terminal state: in $\mathcal{M}_1$, the terminal state is the left-most state in the final level, and in $\mathcal{M}_2$, the terminal state is the right-most state. Reaching the goal in $\mathcal{M}_1$ corresponds to taking the "left" action repeatedly, and reaching the goal in $\mathcal{M}_2$ corresponds to taking the "right" action repeatedly. We consider the posterior distribution that places equal mass on $\mathcal{M}_1$ and $\mathcal{M}_2$, $\mathcal{P}(\mathcal{M}_1|\mathcal{D}) = \mathcal{P}(\mathcal{M}_2|\mathcal{D}) = \frac{1}{2}$. A policy that reaches the

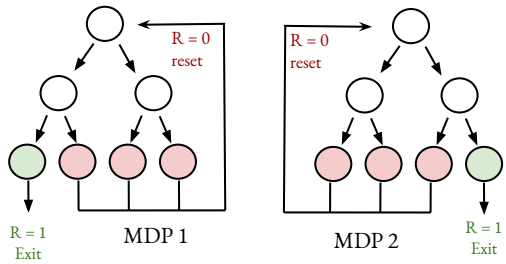

Figure 6: Visual description of Binary Tree MDPs described in proof of Proposition B.4 with depth $n = 3$.

correct terminal state with probability p (otherwise reset) will visit the initial state a Geom(p) number of times, and writing $\overline{\gamma} := \gamma^n$, will achieve return $\frac{\overline{\gamma}p}{1-\overline{\gamma}+p\overline{\gamma}} = \frac{1}{1+\frac{1}{p}\frac{1-\overline{\gamma}}{\overline{\gamma}}}$.

Uniform policy: A uniform policy randomly chooses between "left" and "right" at all states, and will reach all states in the final level equally often, so the probability it reaches the correct goal state is $\frac{1}{2^n}$. Therefore, the expected return is $J(\pi_{\text{unif}}) = \frac{1}{1+2^n\frac{1-\overline{\gamma}}{\overline{\gamma}}}$.

Bayes-optimal memoryless policy: The Bayes-optimal memoryless policy $\pi^{*\text{po}}$ chooses randomly between "left" and "right" at the top level; on every subsequent level, if the agent is in the left half of the tree, the agent deterministically picks "left" and on the right half of the tree, the agent deterministically picks "right". Effectively, this policy either visits the left-most state or the right-most state in the final level. The Bayes-optimal memoryless policy returns to the top of the tree a Geom($p = \frac{1}{2}$) number of times, and the expected return is given by $J(\pi^{*\text{po}}) = \frac{1}{1+2\frac{1-\overline{\gamma}}{\overline{\gamma}}}$.

Everywhere-stochastic policy: Unlike the Bayes-optimal policy, which is deterministic in all levels underneath the first, an everywhere-stochastic policy will sometimes take random actions at these lower levels, and therefore can reach states at the final level that are neither the left-most or right-most states (and therefore always bad). We note that if $\mathcal{H}(\pi(a|s)) > \alpha$, then there is some $\beta > 0$ such that $\max_a \pi(a|s) < 1 - \beta$. For an α-everywhere stochastic policy, the probability of taking at least one incorrect action increases as the depth of the binary tree grows, getting to the correct goal at most probability $\frac{1}{2}(1-\beta)^{n-1}$. The maximal expected return is therefore $J(\pi_s) \leq \frac{1}{1+2(\frac{1}{1-\beta})^{n-1}\frac{1-\overline{\gamma}}{\overline{\gamma}}}$

$$J(\pi^{*\text{po}}) = \frac{1}{1+2\frac{1-\overline{\gamma}}{\overline{\gamma}}} \qquad J(\pi_s) = \frac{1}{1+2(\frac{1}{1-\beta})^{n-1}\frac{1-\overline{\gamma}}{\overline{\gamma}}} \qquad J(\pi_{\text{unif}}) = \frac{1}{1+2^n\frac{1-\overline{\gamma}}{\overline{\gamma}}}$$

As $n \to \infty$, $J(\pi^{*\text{po}})$, $J(\pi_s)$ and $J(\pi_{\text{unif}})$ will converge to zero. Using asymptotic analysis we can determine their speed of convergence and find that:

$$J(\pi^{*\text{po}}) \sim \frac{\overline{\gamma}}{2} \qquad J(\pi_s) \sim \frac{\overline{\gamma}}{2(\frac{1}{1-\beta})^{n-1}} \qquad J(\pi_{\text{unif}}) \sim \frac{\overline{\gamma}}{2^n}$$

Using these asymptotics, we find that

$$\frac{J(\pi_s) - J(\pi_{\text{unif}})}{J(\pi^{*\text{po}}) - J(\pi_{\text{unif}})} \sim \frac{1}{(\frac{1}{1-\beta})^{n-1}} = (1-\beta)^{n-1},$$

which shows that this ratio can be made arbitrarily small as we increase n. □

An aside: deterministic policies While this proposition only discusses the failure mode of stochastic policies, *all* deterministic memoryless policies in this environment also fail. A deterministic policy π_d in this environment continually loops through one path in the binary tree repeatedly, and therefore will only ever reach one goal state, unlike the Bayes-optimal policy which visits both possible goal states. The best deterministic policy then either constantly takes the "left" action (which is optimal for $\mathcal{M}_1$), or constantly takes the "right" action (which is optimal for $\mathcal{M}_2$). Any other deterministic policy

reaches a final state that is neither the left-most nor the right-most state, and will always get 0 reward. The expected return of the optimal deterministic policy is $J(\pi_d) = \frac{\overline{\gamma}}{2}$, receiving $\overline{\gamma}$ reward in one of the MDPs, and 0 reward in the other. When the discount factor γ is close to 1, the maximal expected return of a deterministic policy is approximately $\frac{1}{2}$, while the expected return of the Bayes-optimal policy is approximately 1, indicating a sub-optimality gap.

□

B.5 Proof of Theorem 6.1

Proposition 6.1. *Let $\pi, \pi_1, \cdots \pi_n$ be memoryless policies, and define $r_{\max} = \max_{i,s,a} |r_{\mathcal{M}_i}(s,a)|$. The expected return of π in $\hat{\mathcal{M}}^{po}$ is bounded below as:*

$$J_{\hat{\mathcal{M}}^{po}}(\pi) \geq \frac{1}{n}\sum_{i=1}^{n} J_{\mathcal{M}_i}(\pi_i) - \frac{\sqrt{2}r_{\max}}{(1-\gamma)^2 n}\sum_{i=1}^{n}\mathbb{E}_{s\sim d^{\pi_i}_{\mathcal{M}_i}}\left[\sqrt{D_{KL}\left(\pi_i(\cdot|s) \mid\mid \pi(\cdot|s)\right)}\right], \tag{3}$$

Proof. Before we begin, we recall some basic tools from analysis of MDPs. For a memoryless policy π, the state-action value function $Q^\pi(s,a)$ is given by $Q^\pi(s,a) = \mathbb{E}_\pi[\sum_{t\geq 0}\gamma^t r(s_t,a_t)|s_0 = s, a_0 = a]$. The advantage function $A^\pi(s,a)$ is defined as $A^\pi(s,a) = Q^\pi(s,a) - \mathbb{E}_{a\sim\pi(\cdot|s)}[Q^\pi(s,a)]$. The performance difference lemma [59] relates the expected return of two policies π and π' in an MDP $\mathcal{M}$ via their advantage functions as

$$J_{\mathcal{M}}(\pi') = J_{\mathcal{M}}(\pi) + \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\pi'}_{\mathcal{M}}}[\mathbb{E}_{a\sim\pi'}[A^\pi_{\mathcal{M}}(s,a)]]. \tag{12}$$

We now begin the derivation of our lower bound:

$$\begin{aligned}
J_{\hat{\mathcal{M}}^{\text{po}}}(\pi) &= \frac{1}{n}\sum_{i=1}^{n} J_{\mathcal{M}_i}(\pi) \\
&= \frac{1}{n}\sum_{i=1}^{n} J_{\mathcal{M}_i}(\pi_i) + \frac{1}{n}\sum_{i=1}^{n}\left[J_{\mathcal{M}_i}(\pi) - J_{\mathcal{M}_i}(\pi_i)\right] \\
&= \frac{1}{n}\sum_{i=1}^{n} J_{\mathcal{M}_i}(\pi_i) - \frac{1}{n(1-\gamma)}\sum_{i=1}^{n}\mathbb{E}_{s\sim d^{\pi_i}_{\mathcal{M}_i}}\left[\mathbb{E}_{a\sim\pi_i}\left[A^\pi_{\mathcal{M}_i}(s,a)\right]\right] \\
&= \frac{1}{n}\sum_{i=1}^{n} J_{\mathcal{M}_i}(\pi_i) - \frac{1}{n(1-\gamma)}\sum_{i=1}^{n}\mathbb{E}_{s\sim d^{\pi_i}_{\mathcal{M}_i}}\left[\mathbb{E}_{a\sim\pi_i}\left[A^\pi_{\mathcal{M}_i}(s,a)\right] - \mathbb{E}_{a\sim\pi}\left[A^\pi_{\mathcal{M}_i}(s,a)\right]\right]
\end{aligned} \tag{13}$$

In the last equality we used the fact that $\mathbb{E}_{a\sim\pi}\left[A^\pi(s,a)\right] = 0$. From there we proceed to derive a lower bound:

$$\begin{aligned}
J_{\hat{\mathcal{M}}^{\text{po}}}(\pi) &= \frac{1}{n}\sum_{i=1}^{n} J_{\mathcal{M}_i}(\pi_i) - \frac{1}{n(1-\gamma)}\sum_{i=1}^{n}\mathbb{E}_{s\sim d^{\pi_i}_{\mathcal{M}_i}}\left[\mathbb{E}_{a\sim\pi_i}\left[A^\pi_{\mathcal{M}_i}(s,a)\right] - \mathbb{E}_{a\sim\pi}\left[A^\pi_{\mathcal{M}_i}(s,a)\right]\right] \\
&\geq \frac{1}{n}\sum_{i=1}^{n} J_{\mathcal{M}_i}(\pi_i) - \frac{2r_{max}}{n(1-\gamma)^2}\sum_{i=1}^{n}\mathbb{E}_{s\sim d^{\pi_i}_{\mathcal{M}_i}}\left[D_{TV}\left(\pi_i(\cdot \mid s); \pi(\cdot \mid s)\right)\right] \\
&\geq \frac{1}{n}\sum_{i=1}^{n} J_{\mathcal{M}_i}(\pi_i) - \frac{\sqrt{2}r_{max}}{(1-\gamma)^2 n}\sum_{i=1}^{n}\mathbb{E}_{s\sim d^{\pi_i}_{\mathcal{M}_i}}\left[\sqrt{D_{KL}\left(\pi_i(\cdot \mid s) \mid\mid \pi(\cdot \mid s)\right)}\right]
\end{aligned} \tag{14}$$

where the first inequality is since $|A^\pi_{\mathcal{M}_i}(s,a)| \leq \frac{r_{\max}}{1-\gamma}$ and the second from Pinsker's inequality. Our intention in this derivation is not to obtain the tighest lower bound possible, but rather to illustrate how bounding the advantage can lead to a simple lower bound on the expected return in the POMDP. The inequality can be made tighter using other bounds on $|A^\pi_{\mathcal{M}_i}(s,a)|$, for example using $A_{\max} = \max_{i,s,a}|A^\pi_{\mathcal{M}_i}(s,a)|$, or potentially a bound on the advantage that varies across state. □

B.6 Proof of Proposition 6.1

Proposition 6.2. *Let $f : \{\pi_i\}_{i\in[n]} \mapsto \pi$ be a function that maps n policies to a single policy satisfying $f(\pi, \cdots, \pi) = \pi$ for every policy π, and let α be a hyperparameter satisfying $\alpha \geq \frac{\sqrt{2}r_{max}}{(1-\gamma)^2 n}$. Then letting $\pi_1^*, \ldots \pi_n^*$ be the optimal solution to the following optimization problem:*

$$\{\pi_i^*\}_{i\in[n]} = \arg\max_{\pi_1,\cdots,\pi_n} \frac{1}{n}\sum_{i=1}^{n} J_{\mathcal{M}_i}(\pi_i) - \alpha \sum_{i=1}^{n} \mathbb{E}_{s\sim d_{\mathcal{M}_i}^{\pi_i}} \left[\sqrt{D_{KL}\left(\pi_i(\cdot|s) \mid\mid f(\{\pi_i\})(\cdot|s)\right)}\right], \quad (4)$$

the policy $\pi^* := f(\{\pi_i^*\}_{i\in[n]})$ is optimal for the empirical epistemic POMDP $\hat{\mathcal{M}}^{po}$.

Proof. By Theorem 6.1 we have that $\forall \alpha \geq \frac{\sqrt{2}r_{\max}}{(1-\gamma)^2 n}$:

$$J_{\hat{\mathcal{M}}^{\text{po}}}(f(\{\pi_i^*\})) \geq \frac{1}{n}\sum_{i=1}^{n} J_{\mathcal{M}_i}(\pi_i^*) - \alpha \sum_{i=1}^{n} \mathbb{E}_{s\sim d_{\mathcal{M}_i}^{\pi_i^*}} \left[\sqrt{D_{KL}\left(\pi_i^*(\cdot|s) \mid\mid f(\{\pi_i^*\})(\cdot|s)\right)}\right]. \quad (15)$$

Now, write $\pi'^* \in \arg\max_\pi J_{\hat{\mathcal{M}}^{\text{po}}}(\pi)$ to be an optimal policy in the empirical epistemic POMDP, and consider the collection of policies $\{\pi'^*, \pi'^*, \ldots, \pi'^*\}$. Since $\{\pi_i^*\}$ is the optimal solution to Equation 4, we have

$$\begin{aligned} J_{\hat{\mathcal{M}}^{\text{po}}}(f(\{\pi_i^*\})) &\geq \frac{1}{n}\sum_{i=1}^{n} J_{\mathcal{M}_i}(\pi'^*) - \alpha \sum_{i=1}^{n} \mathbb{E}_{s\sim d_{\mathcal{M}_i}^{\pi'^*}} \left[\sqrt{D_{KL}\left(\pi'^*(\cdot|s) \mid\mid f(\{\pi'^*\})(\cdot|s)\right)}\right] \\ &= \frac{1}{n}\sum_{i=1}^{n} J_{\mathcal{M}_i}(\pi'^*) \\ &= J_{\hat{\mathcal{M}}^{\text{po}}}(\pi'^*), \end{aligned} \quad (16)$$

where the second line here uses the fact that $f(\pi'^*, \ldots, \pi'^*) = \pi'^*$. Therefore $\pi^* := f(\{\pi_i^*\})$ is optimal for the empirical epistemic POMDP. □

C Procgen Implementation and Experimental Setup

We follow the training and testing scheme defined by Cobbe et al. [16] for the Procgen benchmarks: the agent trains on a fixed set of levels, and is tested on the full distribution of levels. Due to our limited computational budget, we train on the so-called "easy" difficulty mode using the recommended 200 training levels. Nonetheless, many prior work has found a significant generalization gap between test and train performance even in this easy setting, indicating it a useful benchmark for generalization [16, 26, 23]. We implemented LEEP on top of an existing open-source codebase released by Jiang et al. [23]. Full code is provided in the supplementary for reference.

LEEP maintains $n = 4$ policies $\{\pi_i\}_{i\in[n]}$, each parameterized by the ResNet architecture prescribed by Cobbe et al. [16]. In LEEP, each policy is optimized to maximize the entropy-regularized PPO surrogate objective alongside a one-step KL divergence penalty between itself and the linked policy $\max_i \pi_i$; gradients are not taken through the linked policy.

$$\mathbb{E}_{\pi_i}[\min(r_t(\pi)A^\pi(s,a), \text{clip}(r_t(\pi), 1-\epsilon, 1+\epsilon)A^\pi(s,a) + \beta\mathcal{H}(\pi_i(a|s)) - \alpha D_{KL}(\pi_i(a|s) \| \max_j \pi_j(a|s))]$$

The penalty hyperparameter α was obtained by performing a hyperparameter search on the Maze task for all the comparison methods (including LEEP) amongst $\alpha \in [0.01, 0.1, 1.0, 10.0]$. Since LEEP trains 4 policies using the same environment budget as a single PPO policy, we change the number of environment steps per PPO iteration from 16384 to 4096, so that the PPO baseline and each policy in our method takes the same number of PPO updates. All other PPO hyperparameters are taken directly from [23].

In our implementation, we parallelize training of the policies across GPUs, using one GPU for each policy. We found it infeasible to run more ensemble members due to GPU memory constraints without significant slowdown in wall-clock time. Running LEEP on one Procgen environment for 50 million steps requires approximately 5 hrs in our setup on a machine with four Tesla T4 GPUs.

D Procgen Results

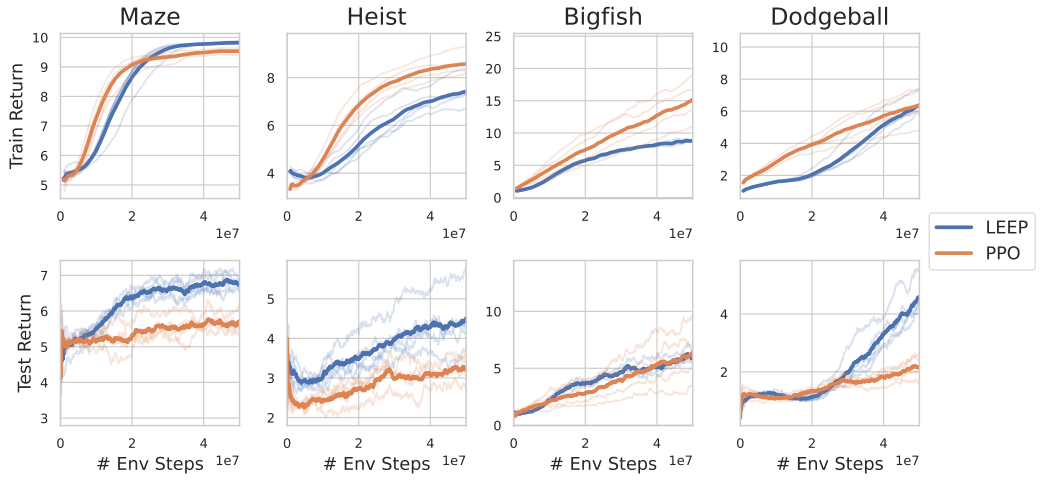

Figure 7: Training (top) and test (bottom) returns for LEEP and PPO on four Procgen environments. Results averaged across 5 random seeds. LEEP achieves equal or higher training return compared to PPO, while having a lower generalization gap between test and training returns.

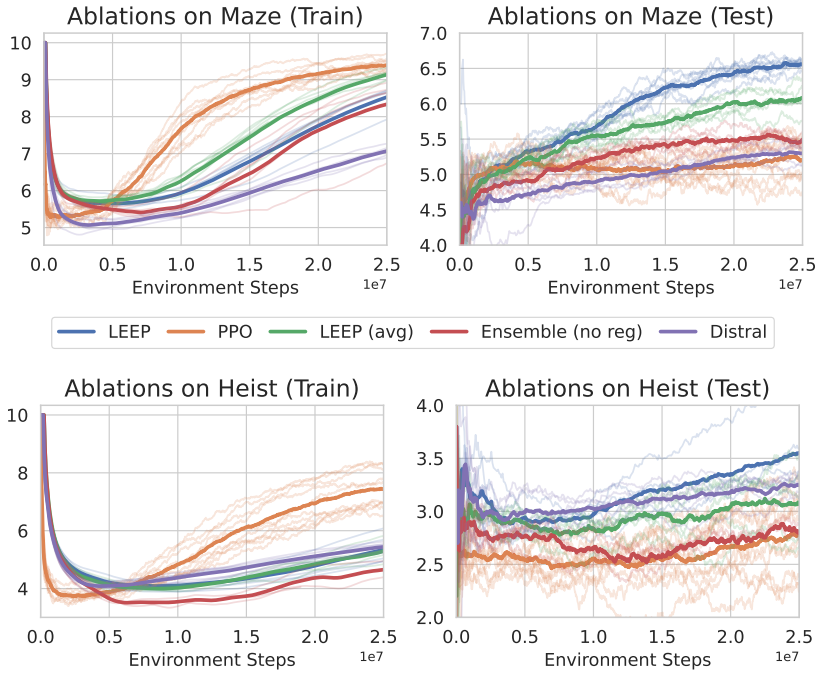

Figure 8: Training and test returns for various ablations and comparisons of LEEP.

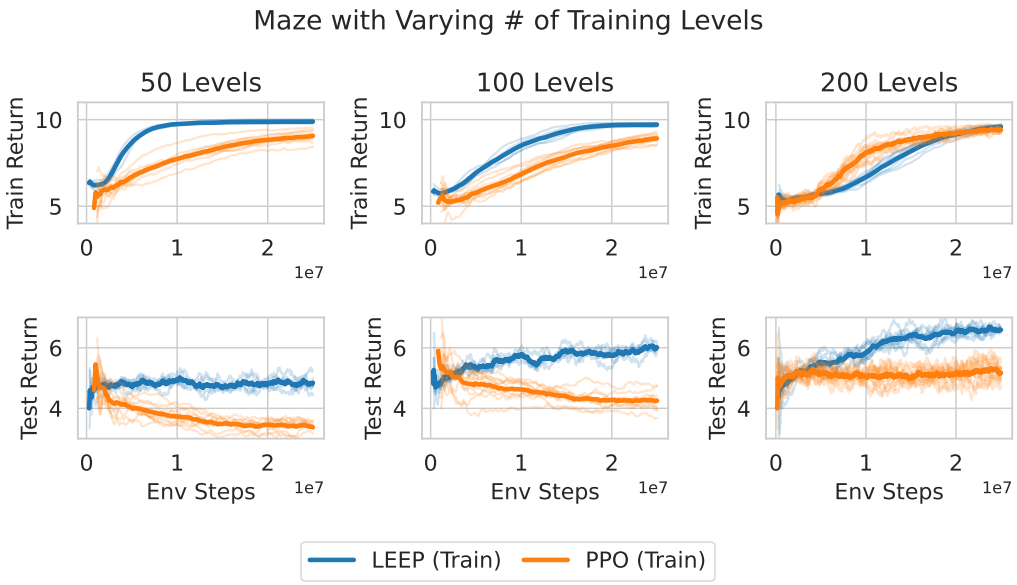

Figure 9: Performance of LEEP and PPO as the number of training levels provided varies. While the learned performance of the PPO policy is worse than a *random policy* with less training levels, LEEP avoids this overfitting and in general, demonstrates a smaller train-test performance gap than PPO.