Unifying Model-Based and Model-Free Reinforcement Learning with Equivalent Policy Sets

Benjamin Freed
bfreed@cs.cmu.edu
Robotics Institute
Carnegie Mellon University

Thomas Wei
thomaswe@cs.cmu.edu
Robotics Institute
Carnegie Mellon University

Roberto Calandra
roberto.calandra@tu-dresden.de
Centre for Tactile Internet
with Human-in-the-Loop
TU Dresden

Jeff Schneider
jeff.schneider@cs.cmu.edu
Robotics Institute
Carnegie Mellon University

Howie Choset
choset@cs.cmu.edu
Robotics Institute
Carnegie Mellon University

Abstract

Model-based and model-free reinforcement learning (RL) each possess relative strengths that prevent either algorithm from strictly outperforming the other. Model-based RL often offers greater data efficiency, as it can use models to evaluate many possible behaviors before choosing one to enact. However, because models cannot perfectly represent complex environments, agents that rely too heavily on models may suffer from poor asymptotic performance. Model-free RL, on the other hand, avoids this problem at the expense of data efficiency. In this work, we seek a unified approach to RL that combines the strengths of both approaches. To this end, we introduce the concept of *equivalent policy sets* (EPS), which quantify the limitations of models for the purposes of decision-making, *i.e.*, action selection. Based on this concept, we propose *Unified RL*, a novel RL algorithm that uses models to constrain model-free RL to the set of policies that are not provably suboptimal, according to model-based bounds on policy performance. We demonstrate across a range of benchmarks that Unified RL effectively combines the relative strengths of both model-based and model-free RL, in that it achieves comparable data efficiency to model-based RL, while achieving asymptotic performance similar or superior to that of model-free RL. Additionally, we show that Unified RL often outperforms a number of existing state-of-the-art model-based and model-free RL algorithms, and *can learn effective policies in situations where either model-based or model-free RL alone fail.*

1 Introduction

Recent successes in model-based reinforcement learning (MBRL) have demonstrated the enormous value that learned representations of environmental dynamics (*i.e.*, models) can confer to autonomous decision-making. For example, models allow agents to evaluate many possible future behaviors, without requiring additional expensive and potentially dangerous environmental interactions. This process is referred to as *planning*, and is a cornerstone of autonomous decision-making.

Models also hold the potential to facilitate cross-task knowledge transfer (Killian et al., 2017) and intelligent exploration (Lowrey et al., 2018; Sekar et al., 2020; Mehta et al., 2021; 2022). In practice, MBRL algorithms often achieve higher data efficiency than model-free algorithms (Deisenroth & Rasmussen, 2011; Heess et al., 2015; Gal et al., 2016a; Chua et al., 2018; Janner et al., 2019; Hafner et al., 2019; 2020; Lin et al., 2023). That being said, models come with their own set of limitations.

Due to their limited representational capacity, models will typically fall short of capturing the full complexity of the real environmental dynamics, which may help explain why MBRL often fails to match the asymptotic performance of model-free RL (MFRL) (Wang et al., 2019).

This limitation of models is exacerbated by the *objective mismatch problem* (Wei et al., 2023): model-learning objectives typically used in MBRL, which are based on some generic measure of accuracy, are often misaligned with the overall goal of increasing reward. Objective mismatch has been shown to negatively impact MBRL performance in practice (Lambert et al., 2020). Several recent approaches have attempted to address objective mismatch by deriving model-learning objectives that are more aligned with the overall RL objective, to enable learned models to be more useful for policy improvement (Joseph et al., 2013; Luo et al., 2018; Rajeswaran et al., 2020; Chow et al., 2020; Grimm et al., 2020; D'Oro et al., 2020; Eysenbach et al., 2022; Ghugare et al., 2022; Wei et al., 2023).

Since practical models will always differ from the true dynamics by some degree, we argue that over-reliance on models will invariably result in some degree of suboptimality. For this reason, we take an alternative approach to addressing the objective mismatch problem. We seek to develop agents that understand the limitations of their models, allowing them to switch to an alternative (*e.g.*, a model-free) learning paradigm in situations where models are not useful for policy improvement. We believe that such an agent would enjoy the benefits of both model-based and model-free learning. To this end, we propose *equivalent policy sets* (EPS), a novel concept for quantifying the limitations of a model for estimating optimal behaviors. We define the EPS as the set of policies that are not *provably suboptimal*, using bounds on the performance of candidate policies, computed using a model. Intuitively, the EPS captures the usefulness of a particular model class for discerning optimal from suboptimal policies.

Based on the concept of the EPS, we propose *Unified RL*, a principled approach to combining MBRL and MFRL that takes advantage of their relative strengths. Unified RL constrains the policy found by MFRL (*e.g.*, soft actor-critic) to lie within the set of non-provably suboptimal policies (the EPS). Here, models are used as a sort of "pre-filtering" step that eliminates provably suboptimal policies from consideration by MFRL. Unified RL leverages the ability of models to rapidly rule-out suboptimal candidate behaviors, while avoiding limitations on asymptotic performance that they introduce.

Unified RL takes a principled approach to dealing with modeling error, because in situations where modeling error is large (*e.g.*, when the environmental dynamics are too complex to be represented accurately, or model learning fails to converge), the lower bounds used to construct the EPS will be loose. As a result, the EPS will be large, and Unified RL will mostly resemble model-free RL. In other words, Unified RL avoids overreliance on its model by avoiding the elimination of policies that the model cannot accurately evaluate due to modeling error.

We show empirically that Unified RL is able to combine the benefits of both model-based and model-free RL on a range of challenging continuous control benchmarks. Furthermore, we show that Unified RL outperforms a wide range of state-of-the-art model-based and model-free RL algorithms. Finally, we show that Unified RL is robust to failure of either its model-based or model-free components. Specifically, when distractors are introduced that prevent the agent from learning well-aligned models, Unified RL continues to make learning progress using model-free policy updates. On the other hand, when poorly selected model-free hyperparameters are used that cause MFRL to fail, Unified RL resorts to MBRL.

2 Background

We represent the environment with which the agent interacts as a Markov decision process (MDP) with initial state distribution $s_0 \sim p_0(s_0)$, state transition dynamics $s_{t+1} \sim T(s_{t+1}|s_t, a_t)$, reward function $r_t \sim R(r_t|s_t, a_t)$ for $t \in \{0, ..., T\}$, and discount factor $\gamma \in [0, 1]$. For simplicity, we assume $\gamma = 1$ and hence ignore it in future exposition. We consider continuous control problems, wherein

the agent learns a policy $\pi \in \Pi$ where $\pi : \mathcal{S} \times \mathcal{A} \to [0, \infty)$ is a state-dependent probability density function over a real-valued action space.

In this work, we formulate the RL problem in Bayesian terms, although the approach is not restricted to Bayesian algorithms. We are concerned with the Bayesian posterior over state transition and reward functions, given by $p(w|D) = {}^{p(D|w)p(w)}/_{p(D)}$, where D is all data observed thus far in the environment, w denotes a parameter vector that parameterizes both the state transition and reward functions, and $p(w)$ is our prior. The prior represents our belief about the dynamics before observing data D, and can be informed by domain-specific knowledge or from previous tasks. Here, we do not assume that we possess any prior knowledge, and therefore choose a generic prior (a Gaussian over w, see Sec. 3.2). We denote our models of the state transition function and reward function, conditioned on a certain parameter vector w, as $p(s'|s, a, w)$ and $p(r|s, a, w)$, respectively. The distribution of trajectories τ given a particular policy π and parameters w is given by $p(\tau|\pi, w) = p(s_0)\pi(a_0|s_0)p(r_0|s_0, a_0, w)\prod_{t=1}^{T} p(s_t|s_{t-1}, a_{t-1}, w)\pi(a_t|s_t)p(r_t|s_t, a_t, w)$. Our inferred posterior distribution over trajectories given the available data D and a policy π is given by $p(\tau|D) = \mathbb{E}_{p(w|D)}p(\tau|\pi, w)$. We denote the expected return of π given a particular parameter vector w as $J(\pi|w) = \mathbb{E}_{p(\tau|\pi,w)}\left[\sum_{t=0}^{T} r_t \middle| \pi, w\right]$. Finally, we define the *Bayesian return* of a policy π to be the expected sum of rewards achieved by π, in expectation over our Bayesian posterior over trajectories This is the quantity that our approach to Bayesian RL attempts to maximize. We refer to a policy that maximizes the Bayesian return $\pi^* \in \arg\max_{\pi\in\Pi} J(\pi|D)$ as the *Bayes-optimal policy*. Note that, by this definition, the Bayes-optimal policy does not depend explicitly on history outside the current episode, as is the case for some definitions Duff (2002). Similarly, we refer to any policy $\pi \notin \arg\max_{\pi\in\Pi} J(\pi|D)$ as *Bayes-suboptimal*.

For many interesting model classes, exact Bayesian posteriors are intractable, and must therefore be approximated with some tractable distribution family. We denote approximate posteriors with $q(w; \theta) \in \mathcal{Q}$, where θ denotes the parameters of the distribution. For example, if q is a multivariate normal distribution, θ may contain the mean vector and variance matrix. We henceforth refer to q as our *model*, because it encodes our learned representation of (our posterior over) the environmental dynamics. In practice, we use a dropout Bayesian neural network (Gal & Ghahramani, 2016b) to represent q, as these have been shown to work well in MBRL (Gal et al., 2016a; Depeweg et al., 2017; Gamboa Higuera et al., 2018).

3 Unifying Model-Based and Model-Free Reinforcement Learning

Here, we introduce the concept of *equivalent policy sets* (EPS) as a tool for quantifying the limitations of models for the purposes of approximating optimal policies. Subsequently, we describe *Unified Reinforcement Learning*, which builds on the concept of the EPS to combine the strengths of model-based and model-free RL.

3.1 Equivalent Policy Sets

To achieve our ultimate goal of developing agents that can flexibly switch between model-based and model-free learning, agents must understand the limitations of models for evaluating and improving policies. To this end, we propose *equivalent policy sets* (EPS) as a tool for quantifying the usefulness of a model for discerning optimal from suboptimal policies. More precisely, we define the EPS $\Pi_E(\theta, D) \subseteq \Pi$ to be the set of all policies that are not provably Bayes-suboptimal, using a model with parameters θ and available data D. To prove the suboptimality of a particular policy π, we use our model to compute a lower bound on (a function f of) the improvement in Bayesian return of a *new* policy π' over π,

$$\mathcal{L}(\pi, \pi', \theta, D) \leq f((J(\pi'|D) - J(\pi|D))), \tag{1}$$

where f is a monotonically increasing function. Although one could use any such $\mathcal{L}$, in this work we take $\mathcal{L}$ to be of the form

$$\mathcal{L}(\pi, \pi', \theta, D) = \mathbb{E}_q \left[f\left(\frac{p(D|w)p(w)}{q(w;\theta)} \left(J(\pi'|w) - J(\pi|w) \right) \right) \right], \tag{2}$$

which we derive in the Sec. A.1 of the Appendix using Jensen's inequality. This particular form of $\mathcal{L}$ is a variational lower bound and requires f to be concave. It is closely related to f-divergences, a generalization of the widely used KL and Rényi divergences (Li & Turner, 2016; Wan et al., 2020). In the closely-related field of variational inference, the effect of the choice of f is an active area of research, and gives rise to various divergence metrics (Kingma & Welling, 2013; Burda et al., 2015; Li & Turner, 2016; Dieng et al., 2017; Chen et al., 2018; Wan et al., 2020). In this work, we consider $f = \log$, as this is the most well-studied choice of f (Blei et al., 2017). $\mathcal{L}$ is tight (*i.e.*, inequality 1 holds with equality) when $q(w;\theta) \propto p(D|w)p(w)(J(\pi'|w) - J(\pi|w))$. Note that, although $\mathcal{L}$ depends on the parameters θ of the *approximate* posterior q, inequality 1 bounds the *exact* difference in Bayesian return between π' and π.

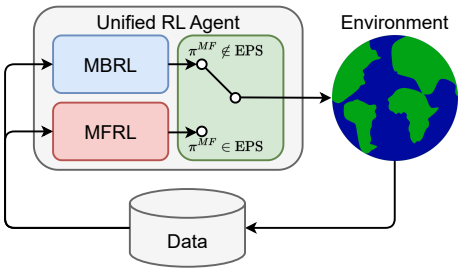

Figure 1: Unified RL combines model-based and model-free RL using the equivalent policy set (EPS). At each iteration, data from a shared buffer are used to update a model-based policy and a model-free policy. We then check whether the model-free policy is contained within the EPS, that is, the set of policies that cannot be proven to be sub-optimal, according to bounds on policy performance computed using the model. If the model-free policy is within the EPS, it is used to collect another episode of data in the environment, which is added to the data buffer. Otherwise, the model-based policy is used to collect more data.

Inequality 1 allows us to prove the suboptimality of any policy π for which there exists a new policy $\pi' \in \Pi$ such that $\mathcal{L}(\pi, \pi', \theta, D) > f(0)$, because this condition implies that π' achieves higher Bayesian return than π, and therefore π is not Bayes-optimal. We can therefore use $\mathcal{L}$ to construct the EPS, which we define to be *the set of policies π for which there does not exist a provably better $\pi' \in \Pi$*, using model parameters θ and data D,

$$\Pi_E(\theta, D) = \{\pi : \max_{\pi' \in \Pi} \mathcal{L}(\pi, \pi', \theta, D) \leq f(0)\}.$$

Given the choice $f = \log$, note that $\mathcal{L}(\pi, \pi', \theta, D) \leq -\infty$ if and only if there exists some $w \in \text{supp}(q)$ such that $J(\pi'|w) - J(\pi|w) \leq 0$. Therefore, the equivalent set definition can be simplified to

$$\Pi_E(\theta, D) = \{\pi : \exists w \in \text{supp}(q) \text{ s.t. } J(\pi'|w) - J(\pi|w) \leq 0\},$$

which is convenient because it allows the data-dependent term in $\mathcal{L}$ to be ignored.

Equivalent Policy Sets for Understanding the Limitations of Models The EPS provides a principled approach to dealing with modeling error, by avoiding the elimination of policies that the model cannot accurately evaluate. In the case where model error can be driven to zero (that is, the approximate posterior matches the ideal posterior, as would be the case with an infinitely expressive posterior class), $\mathcal{L}$ is tight, *i.e.*, inequality (1) holds with equality (see Sec. A.1 for more details). This causes the EPS to reduce to a singleton set containing only the Bayes-optimal policy. In this situation, the agent should fully trust its model because it can correctly identify the optimal policy, which is reflected by the fact that the EPS contains only this policy. However, limitations in modeling resources make this practically infeasible, and in general the model will always contain

some inaccuracies. Existing approaches to MBRL largely have not dealt with this problem, and instead treat the model's approximation of the optimal policy as ground-truth. This can result in highly suboptimal policies, especially when the model is misaligned (Wei et al., 2023; Agarwal et al., 2021). The EPS addresses this problem by quantifying how inaccuracies in our imperfect model translate into uncertainty about the optimal policy, where this uncertainty is represented as a *set* of policies that our model cannot prove are suboptimal. Limitations in model class prevent q from matching the ideal posterior, causing $\mathcal{L}$ to be loose and thereby increasing the size of the EPS. The EPS is guaranteed to contain the Bayes-optimal policy, but will also include other policies that the model is not accurate enough to rule out as suboptimal. This leaves open the possibility of using an alternative learning paradigm such as MFRL to choose a policy from within the EPS to deploy. We therefore argue that the EPS helps fundamentally address problems in MBRL caused by model inaccuracies. This intuition provides the basis for Unified RL, which we describe in the next section.

3.2 Unified Reinforcement Learning

Unified RL builds on the concept of the EPS introduced in the previous section, and is summarized in Alg. 1 and Fig. 1. Unified RL can be thought of as a model-free RL algorithm, where the policy is constrained to lie within the EPS. Through this constraint, Unified RL is able to eliminate many provably suboptimal policies from consideration, thus retaining the data-efficiency benefits of MBRL. However, because Unified RL uses the model only to identify the set of policies that *may* be optimal rather than to estimate a single optimal policy, it avoids over-reliance on the model, and thus avoids the objective mismatch problem associated with typical MBRL approaches. Constraining the model-free policy to lie within the EPS does not in principle prevent MFRL from discovering the Bayes-optimal policy, as the Bayes-optimal policy will always lie within the EPS regardless of the model used to compute the EPS.

Algorithm 1 Unified RL

Given: initial dataset D
for each iteration **do**
  π^{MB}, θ =MBRL(D)
  π^{MF} =SAC(D)
  Estimate $\hat{\mathcal{L}}(\pi^{MF}, \pi^{MB}, \theta, D)$
  if $\hat{\mathcal{L}} > -\infty$ **then**
    $\pi = \pi^{MB}$
  else
    $\pi = \pi^{MF}$
  end if
  for time step t=0,...,T **do**
    $a_t \sim \pi(a_t|s_t)$
    $s_{t+1}, r_t =$ `env.step`(a_t)
    $D \leftarrow D \cup \{s_t, a_t, r_t, s_{t+1}\}$
  end for
end for

We take a simple approach to combining model-based and model-free RL using the EPS, and leave more complex variants to future work. Before each episode, an MBRL and an off-policy MFRL algorithm use the available data D to compute what we refer to as the *model-based policy* π^{MB} and the *model-free policy* π^{MF}, respectively. Subsequently, the agent checks whether the model-free policy is within the EPS; that is, it checks whether a lower bound can be constructed using the model that proves that the model-based policy achieves higher Bayesian return than the model-free policy. If the model-free policy is within the EPS, the agent executes it in the real environment to collect one episode of new data. If not, the agent instead executes the model-based policy, which is guaranteed to be within the EPS. The new data are then added to the shared data buffer, and the entire process repeats. Note that this approach does not require the EPS to be represented explicitly. Instead, the EPS is maintained *implicitly*, in the sense that the lower bound in (1) provides a condition that allows one to check whether a given policy is within the EPS. We describe the individual components of our approach in more detail below, with additional details in Sec. A.2 of the Appendix.

Model-Based RL The MBRL component of our algorithm proceeds in two distinct steps: model training and policy training. During the model training step, we estimate the posterior parameters θ by fitting a Bayesian LSTM dynamics model to our environmental data D, by maximizing an evidence lower bound on data log likelihood (Kingma et al., 2015; Gal & Ghahramani, 2016b;a),

$$\mathcal{L}_{\text{model}}(\theta, D) = \mathbb{E}_{w\sim q(w;\theta)}\left[\sum_{i=1}^{|D|}\sum_{t=1}^{T}\log p(s_{t+1}^{(i)}, r_t^{(i)}|s_{\leq t}^{(i)}, a_{\leq t}^{(i)}, w)\right] - D_{KL}\big(q(w;\theta)||p(w)\big)\,.$$

Specifically, we use the *binary dropout* formulation of Bayesian LSTMs (Gal & Ghahramani, 2016a), wherein sampling a weight from the posterior $w \sim q(w;\theta)$ is accomplished by sampling a binary dropout mask from a fixed Bernoulli distribution (Gal & Ghahramani, 2016b). In this formulation, the prior $p(w)$ is approximately a Normal distribution, while the posterior is a Bernoulli (Gal et al., 2016b). Our dynamics model $p(s_{t+1}^{(i)}, r_t^{(i)}|s_{\leq t}^{(i)}, a_{\leq t}, w)$ is a Gaussian distribution over the next state $s_{t+1}^{(i)}$ and reward $r_t^{(i)}$ with a diagonal covariance matrix, given the states $s_{\leq t}^{(i)}$ and actions $a_{\leq t}^{(i)}$ at all previous timesteps. The choice to represent state transition dynamics as a Gaussian with a diagonal covariance matrix is similar to past work (Gal et al., 2016a; Chua et al., 2018; Gamboa Higuera et al., 2018; Chow et al., 2020; Eysenbach et al., 2022; Freed et al., 2023), with the primary difference being that our dynamics model is recurrent. Specifically, we use an LSMT dynamics model, as we found this to yield more stable gradient-based policy optimization compared to a simple feed-forward MLP.

During the policy training step, we train a Tanh-Gaussian policy (Haarnoja et al., 2018) to maximize the expected cumulative reward predicted by our model. Depending on the environment, we found that one of two methods yielded the best results. In both methods, we start by sampling a set of weights from our approximate posterior (which corresponds to sampling a set of dropout masks). In the first method, for each weight, we sample a set of initial states from the initial state distribution, which we assume to be known. Subsequently, we sample a full T-length trajectory, starting from each initial state, by iteratively sampling actions from the policy, followed by a reward and state transition from the model. Given a batch of sampled trajectories, we compute the policy loss as the negative total reward along the trajectory averaged across sampled trajectories, plus a policy entropy bonus. Similar to Gamboa Higuera et al. (2018), we found that gradient clipping stabilized policy optimization and improved results. We refer to this method as full-trajectory policy training, because full-length trajectories are rolled out.

The second method of policy training that we employ is identical to that used by Hafner et al. (2019), with the slight modification that trajectories are sampled using various dropout masks, and trajectories are sampled in raw state space as opposed to latent space. In summary, states are sampled uniformly from the data buffer, and trajectory segments of length $H = 16$ are sampled starting from those states. Value estimates are then computed using a critic network and the predicted trajectory rewards. The critic is then updated to produce more accurate value estimates, and the policy is updated to produce higher value estimates. In either case, the dropout mask that we use to sample a particular trajectory is held constant during the entire trajectory; this is to reflect the fact that even though there is uncertainty in the dynamics model parameters w, the parameters do not change during a single trajectory (Gal & Ghahramani, 2016a).

Model-Free RL We use Soft Actor-Critic (Haarnoja et al., 2018) as the off-policy MFRL component of our algorithm. We found that standard SAC performed poorly when run off-policy; therefore, we incorporate two modifications suggested by Ball et al. (2023) that we found yielded superior off-policy performance while preserving SAC's on-policy performance. Specifically, we used layer normalization in our Q networks, and omit the entropy term from the Q network loss.

Lower Bound Estimation Using the posterior parameters θ obtained during the model learning process and $f = \log$, it is possible to compute a Monte-Carlo estimate of $\mathcal{L}$ as

$$\hat{\mathcal{L}}(\pi^{MF}, \pi^{MB}, \theta, D) = \sum_{i=1}^{K}\left(\log\frac{p(D|w_i)p(w_i)}{q(w_i;\theta)} + \log\left(\hat{J}(\pi^{MB}|w_i) - \hat{J}(\pi^{MF}|w_i)\right)\right), \tag{3}$$

for $w_1, ..., w_K \sim q(w;\theta)$. Here, $p(D|w_i)$ is the probability of all state transitions and rewards in the dataset given parameters w_i, and $\hat{J}(\pi^{MB}|w)_i$ and $\hat{J}(\pi^{MF}|w_i)$ are themselves Monte-Carlo estimates

of the expected return for the model-based and model-free policies respectively, computed by rolling out a batch of M trajectories from the model using parameters w_i and policies π^{MB} and π^{MF}, respectively. More details on the estimation of this bound are provided in Sec. A.2 of the Appendix.

We review some useful properties of $\hat{\mathcal{L}}$. As $K \to \infty$ and $M \to \infty$, by the law of large numbers, $\hat{\mathcal{L}} \to \mathcal{L}$. However, for $K \to \infty$ and finite M, by Jensen's inequality, $\hat{\mathcal{L}} \leq \mathcal{L}$. This property does not change our algorithm in principle, because $\hat{\mathcal{L}}$ for finite M is still a lower bound on $\log(J(\pi^{MB}|D) - J(\pi^{MF}|D))$. The only practical implication in using $\hat{\mathcal{L}}$ in place of $\mathcal{L}$ is that the algorithm becomes more conservative, preferring model-free RL more often, as it becomes more difficult to prove that the model-based policy achieves a higher Bayesian return. When K is also finite, $\hat{\mathcal{L}}$ is stochastic, and we can no longer say it is strictly a lower bound on $\log(J(\pi^{MB}|D) - J(\pi^{MF}|D))$, though on average it is. Practically, the stochasticity of $\hat{\mathcal{L}}$ injects some randomness into policy selection. We did not find this to be an issue as long as a large enough value of K and M were used.

To check if the model-free policy is in the EPS, we must check whether $\hat{\mathcal{L}}(\pi^{MB}, \pi^{MF}, \theta, D) > \log(0) = -\infty$. Note that in (3), all terms except $\log\left(\hat{J}(\pi^{MB}|w_i) - \hat{J}(\pi^{MF}|w_i)\right)\Big)$ will be defined and finite. However, as $\hat{J}(\pi^{MB}|w_i) - \hat{J}(\pi^{MF}|w_i) \to 0$ from the right, $\log\left(J(\pi^{MB}|w_i) - J(\pi^{MF}|w_i)\right) \to -\infty$. Therefore, this term dominates $\hat{\mathcal{L}}$ when the model-free policy is on or near the boundary of the relevant set, allowing us to simply check whether $\hat{J}(\pi^{MB}|w_i) - \hat{J}(\pi^{MF}|w_i) > 0, \quad \forall i = 1, ..., K$. This property is particularly convenient because it allows us to ignore the $\log p(D|w_i)$ term, which would normally require calling the model on the entire dataset.

4 Experiments

Our experiments seek to answer two questions:

1. Can Unified RL successfully combine the strengths of model-based and model-free RL? If so, can Unified RL perform favorably compared to state-of-the-art prior work?
2. Is Unified RL effective in situations where either MBRL or MFRL alone fail?

We address question 1 in Sec. 4.1, and question 2 in Sec. 4.2. In our experiments, we consider a range of challenging continuous control tasks from the OpenAI gym benchmark suite (Brockman et al., 2016), Deepmind Control Suite (DMC) (Tassa et al., 2018), and the ROBEL robotics benchmark suite (Ahn et al., 2020). Specifically, we consider OpenAI gym Hopper, Walker, Ant, and Half-Cheetah, as well as DMC Cartpole Swingup and ROBEL DClawTurnFixed. We make two modifications to the standard environments for the sake of simplicity. First, we disabled early episode termination in the OpenAI gym tasks, as early termination has been shown to cause issues for MBRL (Wang et al., 2019). Second, because we are not focused on long time-horizon planning in this work, we consider short time horizon tasks; specifically, we consider episode lengths of $T = 100$ for all OpenAI gym and DMC tasks, except for Hopper and Cartpole, which we considered episode lengths of $T = 200$. We use the default episode lengths of $T = 40$ for DClawTurnFixed. We found that these episodes were sufficiently long to allow agents to learn the desired behaviors.

4.1 Data Efficiency and Asymptotic Performance

To empirically evaluate the effectiveness of Unified RL at combining the strengths of model-based and model-free RL, we compare Unified RL to its constituent model-based and model-free components alone. For the fairest possible comparison, our SAC baseline includes the modifications discussed in Sec. 3.2, though we found these to make little difference in SAC's on-policy performance. We also compare to several prior state-of-the-art approaches: Aligned Latent Models (ALM) (Ghugare et al., 2022), Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2015), Twin Delayed DDPG (TD3) (Fujimoto et al., 2018), Proximal Policy Optimization (PPO) (Schulman et al., 2017), Stochastic Value Gradient (SVG) (Heess et al., 2015), and Hybrid Learning (HL) (Pinosky et al., 2023). DDPG, TD3, and PPO are state-of-the-art model-free methods. ALM, SVG and HL are

high-performing algorithms that combine aspects of model-based and model-free RL. For each algorithm and each environment, hyperparameters were manually tuned, using the recommended hyperparameters for that algorithm and environment as a starting point.

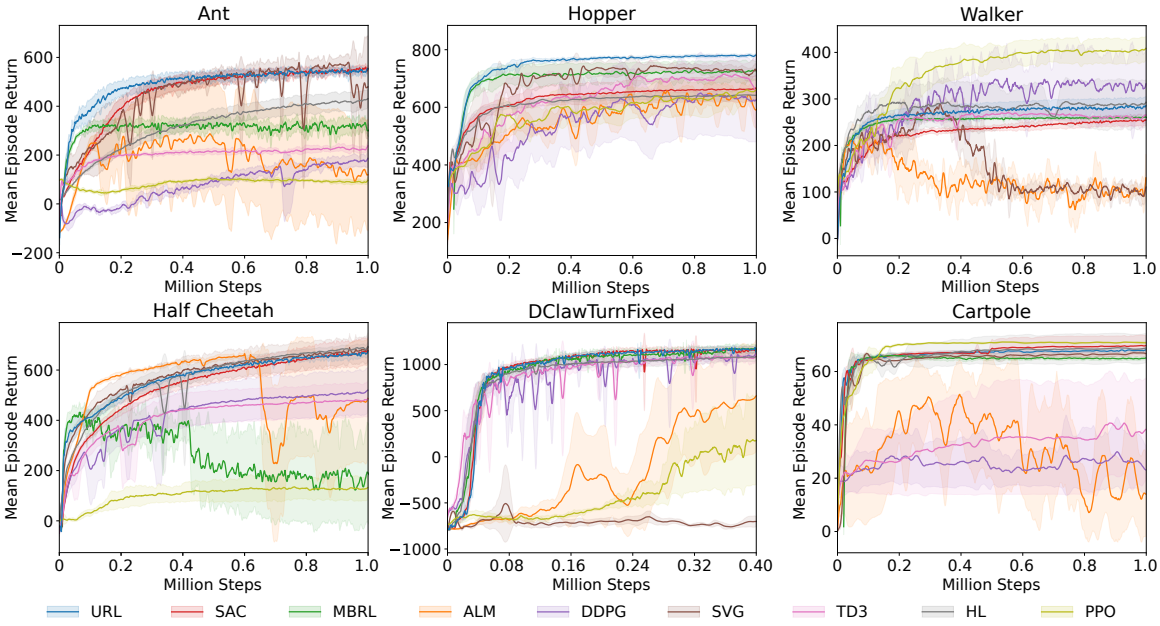

Figure 2: Training curves on benchmark tasks. Solid lines indicate the average return per episode across 5 runs, while shaded regions denote 95% confidence intervals. We find that Unified RL successfully combines the strengths of both model-based and model-free RL. In environments where either MBRL or SAC strictly dominates the other, Unified RL at least matches the better of these two algorithms. In situations where MBRL learns faster initially but is eventually surpassed by SAC, Unified RL achieves higher performance than either algorithm alone. Additionally, Unified RL also performs favorably compared to the other baselines, and is the only algorithm we tested that consistently performs well across all tasks.

We report our results in two ways. First, we show mean episode return vs. the number of environmental steps in Fig. 2. Here, solid lines indicate average episode return averaged across 5 independent random seeds, while shaded regions denote 95% confidence intervals. Second, we report the average episode return across the entire training process for each algorithm, which is equivalent to area under the learning curve divided by number of training steps, in Table 1. This statistic is relevant because it blends both data efficiency and asymptotic performance into a single scalar performance metric. Here again, we report the average return across 5 random seeds, with 95% confidence intervals. We add an additional row to Table 1, labeled Max(SAC,MBRL), where we report the maximum average episode return achieved between either SAC and MBRL. This row corresponds to a naive strategy wherein both SAC and MBRL are run concurrently, with the best policy taken at the end of training.

To determine the statistical significance of the results in Table 1, we performed a student t-test, comparing each algorithm to the best-performing algorithm in each environment. Bold numbers indicate algorithms that either achieved the highest average return in each environment, or algorithms that were not significantly worse than the best-performing algorithm, according to our t-test. We additionally performed a t-test to determine in which environments Unified RL significantly outperforms Max(SAC,MBRL), and marked these results with asterisks. The final column in Table 1 is averaged normalized episode return, which is computed by normalizing all mean episode returns for each environment to the interval between 0 and 1 to arrive at a value that is comparable across environments, and averaging across all environments for each algorithm.

Table 1: Mean episode return and 95% confidence intervals on benchmark tasks. Episodes return was averaged across the entire training process to provide a quantitative performance metric that balances data efficiency and asymptotic performance. Bold numbers indicate the highest-performing algorithm for each environment, or any result that was not significantly worse than the best algorithm. Asterisks (∗) denote environments in which Unified RL significantly outperformed both SAC and MBRL. The final column denotes average normalized return across all environments. We find that in 4 of the 6 environments tested (Ant, Hopper, Walker, and Half Cheetah), Unified RL achieves significantly higher mean episode return compared to either MBRL or SAC, indicating that Unified RL enables a synergy between both algorithms. Additionally, Unified RL is the only algorithm we tested that performed consistently well across all tasks, and achieves the highest average normalized return.

	Ant	Hopper	Walker	Half-Cheetah	Cartpole	DClaw-Turn-Fixed	Avg. norm. return
Unified RL (ours)	$\mathbf{493.1 \pm 9.9}$∗	$\mathbf{750.3 \pm 2.4}$∗	267.2 ± 5.9∗	$\mathbf{571.4 \pm 3.5}$∗	66.7 ± 0.3	940.5 ± 9.7	**0.9211**
SAC	457.3 ± 7.9	632.8 ± 13.6	229.4 ± 2.1	540.3 ± 11.5	67.5 ± 0.9	$\mathbf{949.2 \pm 10.7}$	0.7786
MBRL	310.8 ± 10.4	704.9 ± 16.0	253.2 ± 0.8	263.3 ± 48.3	64.6 ± 0.2	$\mathbf{944.0 \pm 22.6}$	0.6868
max(MBRL,SAC)	457.3 ± 7.9	704.9 ± 16.0	253.2 ± 0.8	540.3 ± 11.5	67.5 ± 0.9	$\mathbf{949.2 \pm 10.7}$	0.8535
ALM	187.7 ± 76.6	562.8 ± 6.2	127.2 ± 7.6	520.4 ± 26.6	31.5 ± 2.9	-252.1 ± 97.5	0.2760
DDPG	76.2 ± 6.4	541.6 ± 42.2	293.0 ± 21.2	421.9 ± 34.0	25.3 ± 3.7	855.9 ± 18.9	0.3888
SVG	443.8 ± 15.3	683.7 ± 13.8	163.4 ± 6.6	$\mathbf{581.8 \pm 21.4}$	65.0 ± 0.3	-704.5 ± 15.9	0.6041
TD3	203.8 ± 3.8	631.8 ± 27.4	249.8 ± 4.2	411.9 ± 22.5	32.6 ± 7.8	882.4 ± 26.5	0.5073
HL	318.6 ± 11.7	616.4 ± 4.5	280.1 ± 18.5	568.9 ± 8.2	66.6 ± 1.8	914.3 ± 40.4	0.7500
PPO	84.5 ± 1.6	582.2 ± 17.9	$\mathbf{356.4 \pm 11.9}$	103.7 ± 16.2	$\mathbf{69.2 \pm 0.1}$	-403.6 ± 56.2	0.3936

Our first observation is that Unified RL succeeds at combining the strengths of its two constituent algorithms. Unified RL significantly outperforms both SAC and MBRL in 4 of the 6 environments we tested, and achieves a higher average normalized return than SAC and MBRL. Furthermore, Unified RL achieves a higher average normalized return than Max(SAC,MBRL), indicating that Unified RL is a better strategy than simply running both SAC and MBRL concurrently, even ignoring the fact that running both algorithms requires twice as many environmental interactions. We find that in environments such as Ant and Half-Cheetah, where MBRL learns rapidly initially but is eventually surpassed by SAC, Unified RL is able to learn as rapidly as MBRL and achieve an asymptotic performance that is at least equivalent to SAC, indicating that Unified RL enables a synergy between MBRL and MFRL. Finally, observe that of all the algorithms we tested, Unified RL was unique in that it performed well across all tasks, and achieves the highest average normalized return. Interestingly, in our experiments, ALM suffered from instability, which we found to be caused by the shorter episode lengths (see Sec. A.2.7).

To gain a deeper insight into the policy-switching behavior of Unified RL, we visualize the fraction of episodes that the agent chooses its model-free policy as training progresses in Fig. 3. We find that in environments where MBRL offers high data efficiency, but MFRL offers higher asymptotic performance, such as Ant, Half Cheetah, and Cartpole, Unified RL uses MBRL for a significant fraction of episodes early in training, but switches almost exclusively to MFRL in later episodes. In Walker, where MBRL outperforms MFRL, Unified RL very briefly uses MFRL, switching to predominantly MBRL for the majority of training. Interestingly, in the Hopper environment, the Unified RL uses a mixture of MFRL and MBRL throughout training. This may explain how Unified RL is capable of outperforming both MFRL and MBRL alone.

4.2 Robustness to Failures of Model-Based and Model-Free RL

One of our central claims is that Unified RL helps avoid the objective mismatch problem by allowing the agent to switch to MFRL when the model is misaligned (that is, ill-suited to helping the agent improve its policy). To test this claim, we evaluate Unified RL on a task that we designed to induce model misalignment in MBRL. Recall that distractors are components of the observation that are predictable but task-irrelevant. Distractors exacerbate model misalignment, because typical model-learning objectives do not prioritize the modeling of task-relevant observation components over

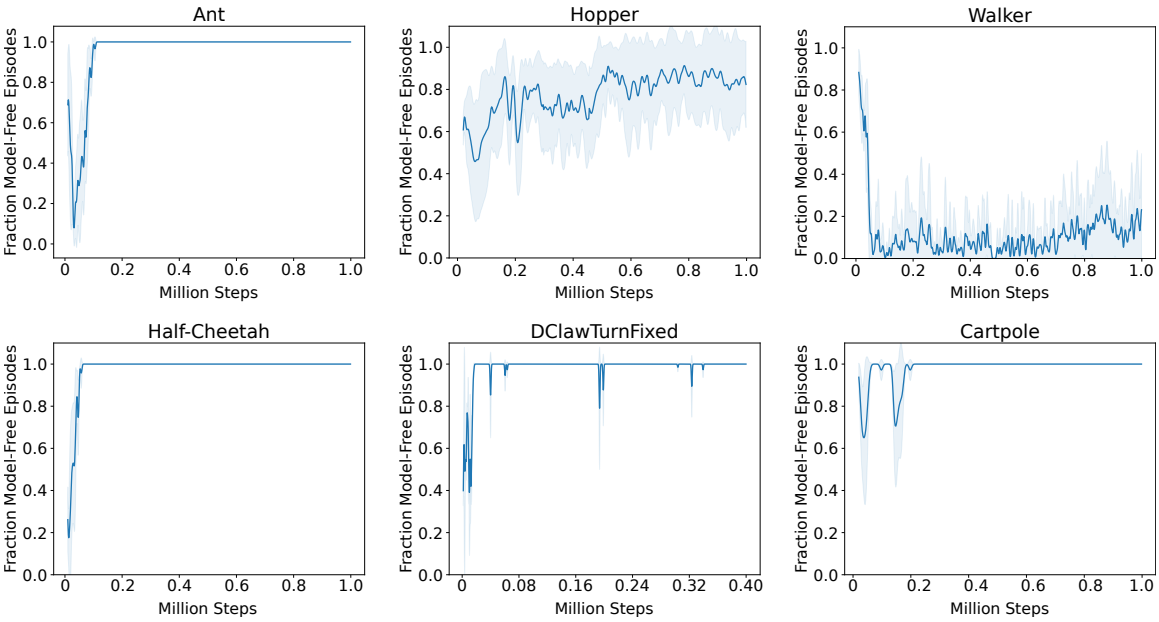

Figure 3: Fraction of model-free episodes vs. training steps.

the task-irrelevant distractors. This results in models that do not accurately represent the task-relevant components. In our experiments, we appended time-dependent sinusoids of fixed frequency to the observations. Sinusoids were grouped together into groups of 10, where all 10 sinusoids in a group had the same phase. Each group was assigned a random phase, preventing the model from simply memorizing the distractors. Five such groups were appended to the observations. The hyperparameters used for SAC, MBRL, and Unified RL for this experiment were identical to those used in the original Ant environment.

The reward curves for this experiment are shown in Fig. 4. We observe that MBRL utterly fails to make learning progress in the presence of distractors, while MFRL is much less affected. In this experiment, Unified RL performed slightly better than MFRL (although not significantly so), indicating that it is able to effectively fall back on MFRL when its model is misaligned.

We do not expect MFRL to always achieve higher asymptotic performance than MBRL in all environments; for example, MFRL may fail to escape a poor local minimum or have poorly tuned hyperparameters. Unified RL has the advantage over other approaches such as MBRL with Model-Free Fine Tuning (Nagabandi et al., 2018), which runs MBRL for a manually specified number of episodes before switching to MFRL, in that Unified RL only switches to MFRL when the model-based policy isn't provably superior. Therefore, in situations where MFRL fails to learn effectively, we expect Unified RL to utilize model-based learning exclusively. To test this claim, we compare the performance of Unified RL to MBRL and SAC in the Ant environment, where the entropy penalty for both SAC and the SAC component of Unified RL was set far higher than its ideal value. As expected, this prevented SAC from learning effectively, both alone and within Unified RL. Indeed, we found that Unified RL recognized that SAC was ineffective at solving the task, instead relied exclusively on MBRL.

5 Related Work

Similar to Duff (2002); Deisenroth & Rasmussen (2011); Gal et al. (2016a); Chua et al. (2018); Gamboa Higuera et al. (2018); Mehta et al. (2021; 2022), we consider a Bayesian formulation of MBRL. The characteristic feature of these approaches is an explicit representation of uncertainty in their estimate of the environmental dynamics. (Gal et al., 2016a), (Depeweg et al., 2017), and

(Gamboa Higuera et al., 2018) are most similar to our approach, in that they use Bayesian neural networks (BNNs) to represent beliefs over dynamics, and learn policies by backpropagating gradients through model rollouts.

Several recent approaches have been proposed for combining model-based and model-free RL. For example, Hybrid Learning (Pinosky et al., 2023) used a learned dynamics model to determine an optimal time to switch between a planned action sequence and a policy learned using MFRL. Stochastic Value Gradients (Heess et al., 2015) proposed a spectrum of policy gradient algorithms that range from model-free methods with value functions to model-based methods without value functions. Finally, Model-Based RL with Model-Free Fine-Tuning initialized MFRL with a policy trained for a fixed number of episodes using MBRL. The primary drawback to these approaches that is addressed in our work is that they use hard-coded or heuristic methods for selecting which learning modality to use in a given situation, rather than switching based on a measure of the model's ability to contribute to policy improvement.

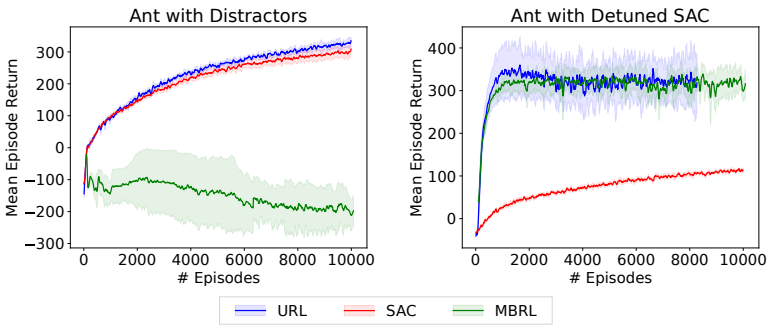

Figure 4: Unified RL is robust to both failures in model-based and model-free RL. (Left) We induce model misalignment by introducing distractors into the observations of the Ant environment, which prevents model-based RL from learning effectively. We find that, in this situation, Unified RL matches the performance of model-free RL, indicating that Unified RL is robust to failure of model-based RL. (Right) We induce failure of model-free RL (SAC) by increasing the entropy bonus to a suboptimal value. In this situation, we find that Unified RL matches the performance of model-based RL, indicating that Unified RL is robust to failure of model-free RL.

Recent approaches for improving model alignment in MBRL optimized policies with respect to lower bounds similar to $\mathcal{L}$. For example, Luo et al. (2018) considered iteratively constructing lower bounds that hold locally in policy space, which are then optimized jointly with respect to both the model and policy. Eysenbach et al. (2022) and Ghugare et al. (2022) considered jointly optimizing a global lower bound on policy performance with respect to both the model and policy parameters. Chow et al. (2020) proposed an EM algorithm to jointly improve the model and the policy with respect to a variational lower bound. One fundamental limitation of these approaches is that they do not address the suboptimalities introduced by the fact that models have limited representational capacity. In environments with complex dynamics that the model class is ill-suited to represent, a lower bound on policy performance may differ significantly from the true objective we wish to optimize (*i.e.,* $\mathcal{L}$ will be a loose bound for the true objective J), resulting in a poorly aligned policy-learning objective and suboptimal policies. Our approach builds on these ideas, but takes a fundamentally different approach: rather than using the model to approximate a single optimal policy, we maintain a set of policies that may be optimal, which is then refined by model-free RL, thereby avoiding over-reliance on potentially inaccurate models.

6 Discussion and Limitations

Our approach has a few limitations that are worth noting. First, Unified RL introduces additional computational overhead over either MBRL or MFRL alone, as it must perform both model-based and model-free learning updates in each iteration. However, we argue that the data efficiency improvements from Unified RL more than outweigh the additional computational cost, particularly in real-world environments where most of the learning time is spent gathering data. For instance,

in the Ant environment, Unified RL requires only 57% of the data of SAC to achieve 80% of its maximum reward. SAC learning updates require only 0.008 s for every 10s of real-world data (assuming an interaction rate of 10 Hz). Therefore, Unified RL will still require less wall-clock time even if it increases the computational overhead by a factor of 945, ignoring the fact that model-based and model-free updates can be parallelized. The details of this calculation are included in Sec. A.2.6 of the appendix.

Second, our approach does not incorporate intelligent exploration, and simply assume that the best policy at any given iteration is the ideal policy to collect new data, be it model-based or model-free. This assumption is potentially disadvantageous in environments that require extensive exploration, where short-term reward should be sacrificed for the purposes of information gain. This limitation could potentially be circumvented with a slight modification to the bound in (2) to include an exploration bonus corresponding to an approximation of the amount of information gained by executing a particular policy, similar to that used by Houthooft et al. (2016).

Finally, because Unified RL maintains separate model-based and model-free policies, but only collects data from one in a given episode, at least one of the two policies will be performing some amount of off-policy learning. This restricts our choice of model-free RL algorithm to off-policy algorithms, such as SAC or Q-learning. Even though SAC is in principle an off-policy algorithm, we found standard SAC to perform poorly when learning off-policy, requiring modifications to the Q learning process (Sec. 3.2) (Ball et al., 2023). This limitation could potentially be avoided by modifying the Unified RL algorithm to maintain one policy, that is updated with model-free RL, but constrained to lie within the equivalent policy set. This could be accomplished by incorporating a constraint into the model-free policy updates, similar to trust regions used in TRPO (Schulman et al., 2015) and PPO (Schulman et al., 2017).

7 Conclusions and Future Work

In this work, we propose *equivalent policy sets* (EPS), which we define as the set of policies that are not provably Bayes-suboptimal, according to bounds on policy performance constructed using a model. The EPS provides a valuable tool for quantifying how inaccuracies in the model translate into uncertainty in their estimate of the optimal policy. Using this tool, agents can better understand in what situations models are useful, and when models should be abandoned in favor of model-free learning updates. Based on this concept, we proposed *Unified RL*, a novel RL algorithm that combines the relative strengths of model-based and model-free RL. Unified RL can be thought of as a model-free RL algorithm, where the enacted policy is constrained to lie within the EPS. Unified RL retains the data-efficiency benefits of model-based approaches by leveraging models to rule out provably suboptimal policies. However, Unified RL avoids over-reliance on models and leverages the asymptotic performance benefits of MFRL by using the MFRL whenever it is not provably suboptimal. We show empirically on a wide range of challenging continuous control RL benchmarks that Unified RL successfully combines the strengths of both MBRL and MFRL, significantly exceeding the performance of either algorithm alone in 4 out of the 6 environments that we tested. We also find that Unified RL outperforms a number of state-of-the-art model-based and model-free prior approaches. Finally, we show that Unified RL learns effective policies in situations where either model-based or model-free RL alone fail.

In future work, we plan to explore using latent dynamics models, similar to those used in Dreamer, for Unified RL, as they have been shown to scale well to high-dimensional observation spaces and complex dynamics (Hafner et al., 2019; 2020; Lin et al., 2023). Additionally, we plan to explore variants of Unified RL that combine more than two RL algorithms. For example, we may simultaneously consider *multiple* MFRL algorithms, and select the best one for the situation, thereby combining the strengths of a wider array of algorithms. This may allow Unified RL to more thoroughly explore the entire set of policies contained within the EPS, rather than choosing between only two policies.

Acknowledgments

Partially funded by the German Research Foundation (DFG, Deutsche Forschungsgemeinschaft) as part of Germany's Excellence Strategy – EXC 2050/1 – Project ID 390696704 – Cluster of Excellence "Centre for Tactile Internet with Human-in-the-Loop" (CeTI) of Technische Universität Dresden, and by the Bundesministerium für Bildung und Forschung (BMBF), German Academic Exchange Service (DAAD) in project 57616814 (SECAI, School of Embedded and Composite AI).

References

Rishabh Agarwal, Marlos C Machado, Pablo Samuel Castro, and Marc G Bellemare. Contrastive behavioral similarity embeddings for generalization in reinforcement learning. *arXiv preprint arXiv:2101.05265*, 2021.

Michael Ahn, Henry Zhu, Kristian Hartikainen, Hugo Ponte, Abhishek Gupta, Sergey Levine, and Vikash Kumar. ROBEL: Robotics benchmarks for learning with low-cost robots. In *Conference on robot learning*, pp. 1300–1313. PMLR, 2020.

Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. *arXiv preprint arXiv:2302.02948*, 2023.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

Liqun Chen, Chenyang Tao, Ruiyi Zhang, Ricardo Henao, and Lawrence Carin Duke. Variational inference and model selection with generalized evidence bounds. In *International conference on machine learning*, pp. 893–902. PMLR, 2018.

Yinlam Chow, Brandon Cui, MoonKyung Ryu, and Mohammad Ghavamzadeh. Variational model-based policy optimization. *arXiv preprint arXiv:2006.05443*, 2020.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

Marc Deisenroth and Carl E Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pp. 465–472, 2011.

Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udluft. Learning and policy search in stochastic dynamical systems with Bayesian neural networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=H1fl8S9ee.

Adji Bousso Dieng, Dustin Tran, Rajesh Ranganath, John Paisley, and David Blei. Variational inference via χ upper bound minimization. *Advances in Neural Information Processing Systems*, 30, 2017.

Pierluca D'Oro, Alberto Maria Metelli, Andrea Tirinzoni, Matteo Papini, and Marcello Restelli. Gradient-aware model-based policy search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 3801–3808, 2020.

Michael O'Gordon Duff. *Optimal Learning: Computational procedures for Bayes-adaptive Markov decision processes*. University of Massachusetts Amherst, 2002.

Benjamin Eysenbach, Alexander Khazatsky, Sergey Levine, and Russ R Salakhutdinov. Mismatched no more: Joint model-policy optimization for model-based rl. *Advances in Neural Information Processing Systems*, 35:23230–23243, 2022.

Benjamin Freed, Siddarth Venkatraman, Guillaume Adrien Sartoretti, Jeff Schneider, and Howie Choset. Learning temporally abstractworld models without online experimentation. 2023.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. *Advances in neural information processing systems*, 29, 2016a.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059. PMLR, 2016b.

Yarin Gal, Rowan McAllister, and Carl Edward Rasmussen. Improving PILCO with Bayesian neural network dynamics models. In *Data-efficient machine learning workshop, ICML*, volume 4, pp. 25, 2016a.

Yarin Gal et al. Uncertainty in deep learning, 2016b.

Juan Camilo Gamboa Higuera, David Meger, and Gregory Dudek. Synthesizing neural network controllers with probabilistic model-based reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2538–2544, 2018. doi: 10.1109/IROS.2018.8594018.

Raj Ghugare, Homanga Bharadhwaj, Benjamin Eysenbach, Sergey Levine, and Russ Salakhutdinov. Simplifying model-based rl: Learning representations, latent-space models, and policies with one objective. In *The Eleventh International Conference on Learning Representations*, 2022.

Christopher Grimm, André Barreto, Satinder Singh, and David Silver. The value equivalence principle for model-based reinforcement learning. *Advances in Neural Information Processing Systems*, 33:5541–5552, 2020.

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

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2019.

Danijar Hafner, Timothy P Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2020.

Nicolas Heess, Gregory Wayne, David Silver, Timothy Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. *Advances in neural information processing systems*, 28, 2015.

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

Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. *Advances in neural information processing systems*, 29, 2016.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019.

Joshua Joseph, Alborz Geramifard, John W. Roberts, Jonathan P. How, and Nicholas Roy. Reinforcement learning with misspecified model classes. In *2013 IEEE International Conference on Robotics and Automation*, pp. 939–946, 2013. doi: 10.1109/ICRA.2013.6630686.

Taylor W Killian, Samuel Daulton, George Konidaris, and Finale Doshi-Velez. Robust and efficient transfer learning with hidden parameter markov decision processes. *Advances in neural information processing systems*, 30, 2017.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. *Advances in neural information processing systems*, 28, 2015.

Nathan Lambert, Brandon Amos, Omry Yadan, and Roberto Calandra. Objective mismatch in model-based reinforcement learning. *arXiv preprint arXiv:2002.04523*, 2020.

Yingzhen Li and Richard E Turner. Rényi divergence variational inference. *Advances in neural information processing systems*, 29, 2016.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Jessy Lin, Yuqing Du, Olivia Watkins, Danijar Hafner, Pieter Abbeel, Dan Klein, and Anca Dragan. Learning to model the world with language. *arXiv preprint arXiv:2308.01399*, 2023.

Kendall Lowrey, Aravind Rajeswaran, Sham Kakade, Emanuel Todorov, and Igor Mordatch. Plan online, learn offline: Efficient learning and exploration via model-based control. *arXiv preprint arXiv:1811.01848*, 2018.

Yuping Luo, Huazhe Xu, Yuanzhi Li, Yuandong Tian, Trevor Darrell, and Tengyu Ma. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. In *International Conference on Learning Representations*, 2018.

Viraj Mehta, Biswajit Paria, Jeff Schneider, Stefano Ermon, and Willie Neiswanger. An experimental design perspective on model-based reinforcement learning. In *International Conference on Learning Representations*, 2021.

Viraj Mehta, Ian Char, Joseph Abbate, Rory Conlin, Mark Boyer, Stefano Ermon, Jeff Schneider, and Willie Neiswanger. Exploration via planning for information about the optimal trajectory. *Advances in Neural Information Processing Systems*, 35:28761–28775, 2022.

Anusha Nagabandi, Gregory Kahn, Ronald S. Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7559–7566, 2018. doi: 10.1109/ICRA.2018.8463189.

Allison Pinosky, Ian Abraham, Alexander Broad, Brenna Argall, and Todd D Murphey. Hybrid control for combining model-based and model-free reinforcement learning. *The International Journal of Robotics Research*, 42(6):337–355, 2023.

Aravind Rajeswaran, Igor Mordatch, and Vikash Kumar. A game theoretic framework for model based reinforcement learning. In *International conference on machine learning*, pp. 7953–7963. PMLR, 2020.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In *International Conference on Machine Learning*, pp. 8583–8592. PMLR, 2020.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Neng Wan, Dapeng Li, and Naira Hovakimyan. f-divergence variational inference. *Advances in neural information processing systems*, 33:17370–17379, 2020.

Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. Benchmarking model-based reinforcement learning. *arXiv preprint arXiv:1907.02057*, 2019.

Ran Wei, Nathan Lambert, Anthony D McDonald, Alfredo Garcia, and Roberto Calandra. A unified view on solving objective mismatch in model-based reinforcement learning. *Transactions on Machine Learning Research*, 2023.

A Appendix

A.1 Derivation of Lower Bound in eq. (2)

The difference in Bayesian return between policies π' and π is given by

$$J(\pi'|D) - J(\pi|D) = \int_{\mathcal{W}} \frac{p(D|w)p(w)}{p(D)} \left(J(\pi'|w) - J(\pi|w)\right) dw.$$

By introducing an approximate posterior $q(w;\theta)$, we can write the above expression as an expectation over q,

$$J(\pi'|D) - J(\pi|D) = \mathbb{E}_q \left[\frac{p(D|w)p(w)}{q(w;\theta)p(D)} \left(J(\pi'|w) - J(\pi|w)\right) \right].$$

Let $\tilde{f}$ be a concave, monotonically increasing function. Taking $\tilde{f}$ of both sides and applying Jensen's inequality, we arrive at a lower bound on $\tilde{f}(J(\pi'|D) - J(\pi|D))$,

$$\begin{aligned} \tilde{f}(J(\pi'|D) - J(\pi|D)) &= \tilde{f}\left(\mathbb{E}_q \left[\frac{p(D|w)p(w)}{q(w;\theta)p(D)} \left(J(\pi'|w) - J(\pi|w)\right) \right] \right) \\ &\geq \mathbb{E}_q \left[\tilde{f}\left(\frac{p(D|w)p(w)}{q(w;\theta)p(D)} \left(J(\pi'|w) - J(\pi|w)\right) \right) \right]. \end{aligned}$$

Finally, to arrive at $\mathcal{L}$, we define a new concave monotonically increasing function $f(x) = \tilde{f}(p(D)x)$ and substitute this into the above expression to eliminate the constant $p(D)$ term,

$$\tilde{f}(J(\pi'|D) - J(\pi|D)) \geq \mathbb{E}_q \left[f\left(\frac{p(D|w)p(w)}{q(w;\theta)} \left(J(\pi'|w) - J(\pi|w)\right)\right)\right]$$
$$= \mathcal{L}(\pi, \pi', \theta, D).$$

To prove that π' achieves higher Bayesian return than π, it is sufficient to show that $\mathcal{L}(\pi, \pi', \theta, D) > \tilde{f}(0) = f(0)$, thus the data likelihood term $p(D)$ is irrelevant in constructing the EPS.

A.2 Implementation Details

A.2.1 Model Architecture and Training

The dynamics model we used for all tasks consisted of a single linear input layer, followed by a single-directional, single-layer LSTM cell (Hochreiter & Schmidhuber, 1997), followed by two linear layers, and an output layer. The output layer consisted of four separate output heads, one each for state mean, reward mean, state standard deviation, and reward standard deviation. The standard deviation output heads used softplus activations to ensure their output was positive, while the mean layers did not an activation function. ReLU activations were used for all other layers other than the LSTM cell. State means were represented as learned deltas from previous states. That is, the state mean output predicts the mean in the *difference* between the current and last state. All inputs (states and actions) and outputs (state deltas and rewards) of the dynamics model were normalized before each period of model training to be of mean zero and unit variance.

Before each layer other than the initial input layer, including each internal layer within the LSTM cell, a binary dropout mask (Srivastava et al., 2014) was applied, which was used by Gal & Ghahramani (2016a) and Gal & Ghahramani (2016b) to represent uncertainty in neural network parameters. Crucially, both in training and when sampling rollouts, the dropout mask is held fixed across all timesteps in a trajectory (Gal & Ghahramani, 2016b), while different dropout masks are sampled across trajectories. The dynamics model was trained with the following loss computed on a batch of trajectories sampled from the data buffer:

$$\mathcal{L}_{\text{model}} = \frac{1}{B} \sum_{i=1}^{B} \sum_{t=0}^{T} \left(\log p(s_{t+1}|s_t, a_t, w_i) + \log p(r_t|s_t, a_t, w_i)\right) + \frac{\eta}{N}||W||_2^2$$

where $B = 100$ is the batch size, T is the episode length, w_i is the dropout mask corresponding to the ith trajectory, η is a factor that determines the length-scale of the prior (Gal & Ghahramani, 2016b), N is the number of trajectories in the training dataset, and W is the set of all learnable parameters in the network.

A.2.2 Policy Architecture

Both model-based RL and SAC use a Tanh-Gaussian MLP policy with three layers, with Tanh activations between layers. Policies used in MBRL have 1024 units in their hidden layers, while policies used for SAC have 256. The policies have two output heads, one for mean and one for standard deviation. The mean output head uses no activation function, while the standard deviation head uses either a softplus activation to ensure that the standard deviation is positive, or a sigmoid activation to force the standard deviation to be bounded. To force samples from the policy to fall within the specified action range of the environment, samples are passed through a tanh function.

A.2.3 Critic Architecture

The critic network used for SAC was a state-action value function, while the critic used for MBRL was a state value function. In either case, critics consisted of 3 layers with ReLU activations between layers, with 256 units in each hidden layer.

A.2.4 Lower Bound Estimation

As discussed in Sec. A.2.4, to check whether the model-free policy π^{MF} is within the EPS, we need only check whether $\hat{J}(\pi^{MB}|w_i) - \hat{J}(\pi^{MF}|w_i) > 0, \quad \forall i = 1, ..., K$, where $\hat{J}(\pi|w_i)$ is a Monte-Carlo estimate of $J(\pi|w_i)$, the expected return for policy π given dynamics model parameters w_i. In the dropout formulation of BNNs, sampling w_i corresponds to sampling a dropout mask, so we use w_i to denote a particular dropout mask. Therefore, to compute $\hat{J}(\pi|w_i)$, we sample one dropout mask, and sample M state-action-reward trajectories from our dynamics model and policy, from timesteps $t = 0$ to T using that dropout mask, and average the return across those trajectories:

$$\hat{J}(\pi|w_i) = \frac{1}{M} \sum_{t=0}^{T} r_t,$$

for $a_t \sim \pi(a_t|s_t)$, $r_t \sim p(r_t|s_t, a_t, w_i)$, and $s_{t+1} \sim p(s_{t+1}|s_t, a_t, w_i)$. Note that the dropout mask w_i is held constant across timesteps.

A.2.5 Hyperparameters

Table 2 contains the hyperparameters used for Unified RL for each task.

- K=Number of dropout masks sampled when computing $\hat{\mathcal{L}}$ ((3))
- M=Number of trajectories sampled when computing $\hat{\mathcal{L}}$ ((3))
- Policy training: whether the model-based policy is trained using full-trajectory policy training or Dreamer-style policy training, as described in Sec. 3.
- σ_{max}: In some cases, we found it useful to bound the maximum value that the policy standard deviation could take, by placing a sigmoid activation on the standard deviation output of the policy and multiplying by a constant. We refer to this upper bound as σ_{max}
- α_{MB}: entropy bonus used for the model-based policy training
- α_{MF}: entropy bonus used for SAC
- Automatic Entropy Tuning (MB policy): whether automatic entropy tuning is used for the model-based policy (makes α_{MB} irrelevant)
- Automatic Entropy Tuning (MF policy): whether automatic entropy tuning is used for the SAC policy (makes α_{MF} irrelevant)
- T: episode length

We additionally found it necessary to provide SAC with enough on-policy data by enforcing that at least one out of every 10 episodes was run using the MFRL policy.

Table 2: Hyperparameters used in Unified RL

Environment	K	M	Policy Training	σ_{max}	α_{MB}	α_{MF}	Auto Ent Tuning (π^{MB})	Auto Ent Tuning (π^{MF})	T	η
Ant	50	5	Dreamer	None	0.2	-	False	True	100	200
Hopper	50	100	Full trajectory	None	0.2	0.2	False	False	200	100
Walker	50	100	Full trajectory	0.1	0.2	0.2	False	False	100	100
Half Cheetah	50	5	Dreamer	None	0.1	-	False	True	100	100
Cartpole	50	default	Dreamer	None	0.2	-	False	True	200	100
DClaw-TurnFixed	50	10	Full Trajectory	None	0.2	0.2	False	False	40	200

A.2.6 Overhead Calculation

Here, we calculate the maximum permissible computational overhead increase of Unified RL over SAC that would result in both algorithms reaching a particular threshold of performance in the same amount of wall-clock time. We consider a threshold of 80% of the maximum performance of SAC, and that Unified RL and SAC follow the same respective learning curves as the Ant environment. Finally, we assume a realistic "real-world" interaction speed of 10 Hz, and an episode length of 100 timesteps, meaning that episodes last 10 seconds. We assume no parallelization, *i.e.,* SAC updates, model-based updates, and data collection all occur sequentially.

T_x = wall-clock time to reach performance threshold for algorithm x,

e_x = number of learning episodes required to reach performance threshold for algorithm x ($e_{URL} = 0.57 e_{SAC}$ because Unified RL requires 57% of the data of SAC),

c = factor of computational overhead increase from SAC to Unified RL,

u_x = update time for algorithm x ($u_{SAC} = 0.008$, $u_{URL} = c u_{SAC}$).

Setting the wall-clock times for each algorithm equal:

$$T_{SAC} = T_{URL},$$

$$(10 + u_{SAC}) e_{SAC} = (10 + u_{URL}) e_{URL}.$$

Substituting in expressions from above and solving for c,

$$(10 + 0.008) e_{SAC} = (10 + c(0.008))(0.57) e_{SAC}.$$

$$c = 944.7.$$

This indicates that Unified RL updates can be up to 944.7x slower than SAC updates and still require less wall-clock time to achieve the performance threshold.

A.2.7 ALM Results on Full-Length Trajectories

To determine the cause of the differences in results on Half-Cheetah and Ant reported by Ghugare et al. (2022) and our ALM results, reported in Sec. 4, we ran ALM experiments in Half-Cheetah and Ant with full, 1000-length episodes. We found that in Half-Cheetah, ALM performed better than the results reported by Ghugare et al. (2022) (Fig. 5, left). In Ant, we found the results to be comparable, although less stable, than those reported by Ghugare et al. (2022) (Fig. 5, left). Because the only difference between these experiments and the ones we report in 4 is episode length, we can conclude that this is the cause of the discrepancy in the results.

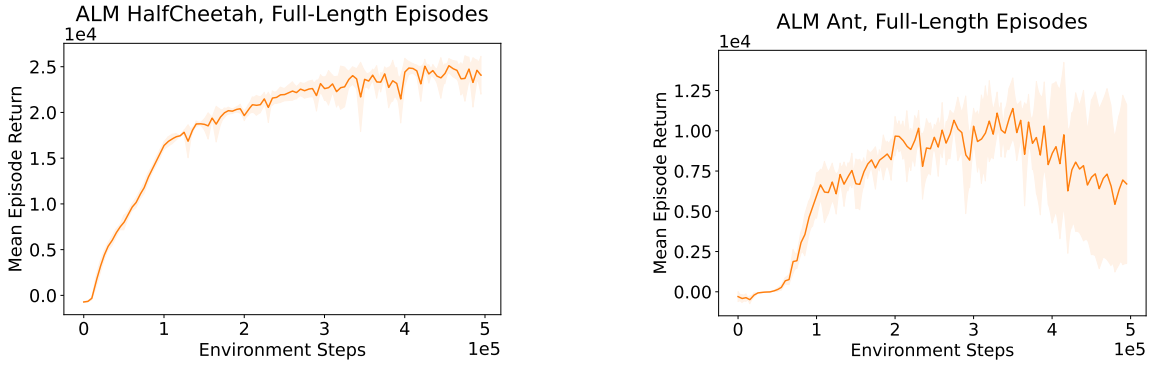

Figure 5: Reward Curves for ALM in Half-Cheetah (left) and Ant (Right), with Full-Length Episodes.