EFFICIENT MODEL-BASED REINFORCEMENT LEARNING THROUGH OPTIMISTIC THOMPSON SAMPLING

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning complex robot behavior through interactions with the environment necessitates principled exploration. Effective strategies should prioritize exploring regions of the state-action space that maximize rewards, with optimistic exploration emerging as a promising direction aligned with this idea and enabling sample-efficient reinforcement learning. However, existing methods overlook a crucial aspect: the need for optimism to be informed by a belief connecting the reward and state. To address this, we propose a practical, theoretically grounded approach to optimistic exploration based on Thompson sampling. Our model structure is the first that allows for reasoning about *joint* uncertainty over transitions and rewards. We apply our method on a set of MuJoCo and VMAS continuous control tasks. Our experiments demonstrate that optimistic exploration significantly accelerates learning in environments with sparse rewards, action penalties, and difficult-to-explore regions. Furthermore, we provide insights into when optimism is beneficial and emphasize the critical role of model uncertainty in guiding exploration.

1 INTRODUCTION

Reinforcement Learning (RL) is recognized as a promising approach to solving complex sequential decision-making tasks in robotics (Levine et al., 2016). However, many popular RL algorithms require millions of interactions with the real world to train effective policies (Schulman et al., 2017; Mnih et al., 2015). This sample inefficiency is primarily due to the central challenge of balancing exploration, gathering information about the world, with exploitation, maximizing rewards given current knowledge (Sutton & Barto, 2018; Szepesvári, 2022). Sample inefficiency is especially prohibitive for RL applied to robotics due to the high cost and potential wear on physical systems.

Efficient RL requires a careful balance between exploration and exploitation, and some approaches are better suited for this than others. Our work centers on model-based RL, a class of algorithms where agents build a predictive model of the environment dynamics from which they derive a policy. Many works have demonstrated impressive sample efficiency using model-based RL by leveraging the learned dynamics model to simulate future outcomes, thereby allowing agents to plan strategically and optimize the controller using fewer real-world trials (Ibarz et al., 2021; Chua et al., 2018; Janner et al., 2019; Yang et al., 2020; Sekar et al., 2020; Hansen et al., 2024). However, learning a model of the environment is a double-edged sword: a faithful model enhances efficiency by reducing the number of interactions needed in the real world while a biased model severely hampers performance by misguiding the policy (Gu et al., 2016). Planning with *uncertainty-aware* dynamics reduces the effects of model errors (Deisenroth & Rasmussen, 2011; Deisenroth et al., 2013; Gal et al., 2016) and allows model-based RL algorithms to reach state-of-the-art asymptotic performance on benchmark control tasks (Chua et al., 2018; Janner et al., 2019). Furthermore, by learning a probabilistic dynamics model, model-based RL enables directing exploration in a principled manner towards unseen parts of the state-action space.

In this work, we focus on *optimistic exploration* using model-based RL with an uncertainty-aware model. In a subtle departure from classical exploration, which prioritizes transitions with high uncertainty under the model, we define optimistic exploration as favoring transitions likely to yield high rewards. This approach aligns well with the RL objective of maximizing cumulative rewards (Sutton & Barto, 2018). Notable works unite optimistic exploration with uncertainty-aware model-based

RL to hallucinate plausible, optimistic training experiences for superior robustness and sample efficiency (Curi et al., 2020; Sessa et al., 2022), however these formulations ignore joint uncertainty between the reward and dynamics. They also assume knowledge of a reward function, which can be impractical in real-world applications due to the difficulty of specifying dynamic, variable, and subjective reward functions accurately a priori. While learning the reward function is a straightforward remedy, there is a dearth of efficient and principled approaches to integrating reward learning into optimistic exploration for model-based RL.

Contributions. Our primary contribution is the first practical model-based RL algorithm, called Hallucination-based Optimistic Thompson sampling with Gaussian Processes (HOT-GP), that can be used with state-of-the-art off-policy RL algorithms for *principled* optimistic exploration. Inspired by the premise that efficient exploration should be informed by joint uncertainty over state and reward distributions, we propose a GP model to maintain this joint belief in order to simulate transitions that are simultaneously plausible under the learned dynamics and optimistic with respect to the estimated reward. We evaluate this approach on a set of MuJoCo (Todorov et al., 2012) and VMAS (Bettini et al., 2022) continuous control tasks. The results reveal that HOT-GP matches or improves sample efficiency on standard benchmark tasks and substantially accelerates learning in challenging settings involving sparse rewards, action penalties, and difficult-to-explore regions. As these findings validate our hypothesis, our secondary contribution is an empirically-supported argument for maintaining a belief over *both* the reward and state distributions, as this is key for effective optimistic exploration. Moreover, we study the factors that influence the utility of optimism and underscore the important role of model uncertainties in shaping exploration.

2 RELATED WORK

Model-based RL methods are promising for complex real-world decision problems due to their potential for data efficiency (Kaelbling et al., 1996). Selecting the appropriate model is critical, as it must facilitate effective learning in both low-data regimes (during early stages of training) and high-data regimes (later in training). This requirement naturally aligns with the advantages of Bayesian models. Non-parametric models like Gaussian processes (GPs) provide excellent performance in settings with limited, low-dimensional data (Deisenroth et al., 2013; Deisenroth & Rasmussen, 2011; Kamthe & Deisenroth, 2018). Neural network predictive models have also been effectively used for tasks with non-smooth dynamics (Nagabandi et al., 2018) and high dimensions like the Atari suite (Oh et al., 2015; Kaiser et al., 2020). As deterministic neural networks tend to overfit to data in the early stages of training, uncertainty-aware neural network models have been shown to achieve better performance with algorithms such as PETS (Chua et al., 2018) and MBPO (Janner et al., 2019). However, designing scalable Bayesian neural networks remains an open problem (Roy et al., 2024; Guo et al., 2017; Osband, 2016) and popular approximations using dropout (Gal et al., 2017) and ensembles (Lakshminarayanan et al., 2017; Chua et al., 2018) fall short of the precise uncertainty quantification that GPs provide. In order to benefit from both the high-capacity function approximation of neural networks and probabilistic inference of GPs, in this work we model the joint dynamics and reward functions using a GP with a neural network mean function (Iwata & Ghahramani, 2017).

Thompson sampling is a provably efficient exploration algorithm in RL (Thompson, 1933). This approach implicitly balances exploration and exploitation by sampling statistically plausible outcomes from the posterior distribution over dynamics models and selecting actions based on the sampled outcomes (Russo & Van Roy, 2014; Russo et al., 2018). Thompson sampling has been studied on tabular MDPs (Osband et al., 2013) and extended in theory to continuous state-action spaces with sublinear regret under certain conditions (Chowdhury & Gopalan, 2019).

Optimism in the face of uncertainty is another theoretically grounded approach to guide exploration that selects actions that optimize for optimistic outcomes. The motivation is to promote exploration guided by beliefs about future value to facilitate efficient learning. While optimism has been extensively studied in tabular MDPs (Brafman & Tennenholtz, 2002; Jaksch et al., 2010) and linear systems (Jin et al., 2020; Abbasi-Yadkori & Szepesvári, 2011; Neu & Pike-Burke, 2020), optimism in continuous state-action MDPs with non-linear dynamics remains less explored. GP-UCRL (Chowdhury & Gopalan, 2019) is one such theoretical approach that targets dynamics models that lie in a Reproducing Kernel Hilbert Space. Another is LOVE (Seyde et al., 2021), which uses an ensemble of networks to direct exploration towards regions with high predicted potential for long-term improvement. TIP (Mehta et al., 2022) also employs optimism by selecting actions according to an optimistic acquisition function. Curi et al. (2020) proposes a reparameterization of the dynamics function space in order to reduce optimistic exploration to greedy exploitation with the H-UCRL algorithm. H-UCRL has sublinear regret bounds under certain conditions and demonstrates efficient exploration on deep RL benchmark tasks with action penalties. The H-MARL algorithm extends this to the multi-agent setting for general-sum Markov games and introduces a practical approximation for generating optimistic states (Sessa et al., 2022). While these methods all derive useful optimistic exploration strategies for RL, they ignore the relationship between uncertainty around dynamics and uncertainty around the reward. Moreover, TIP, H-UCRL, H-MARL use the ground-truth reward function throughout training. Our approach, inspired by H-UCRL (Curi et al., 2020), improves optimistic hallucination for model-based RL by learning and leveraging the joint uncertainty over state transitions and associated rewards.

Reward learning is commonly applied in RL for reward shaping (Hu et al., 2020). This is closely related to exploration, as intrinsic rewards learned through curiosity-driven methods help guide exploration by reducing uncertainty over the agent's knowledge of the environment (Pathak et al., 2017). Reward learning also plays an important role in model-based RL. Receding-horizon planning approaches like PETS (Chua et al., 2018), Dreamer (Hafner et al., 2020; 2019; 2023), and TD-MPC (Hansen et al., 2022; 2024) use estimated rewards to evaluate and optimize sequences of actions, while model-based policy learning approaches like MBPO (Janner et al., 2019) use estimated rewards to inform policy updates. Some of these approaches explicitly account for uncertainty in the learned reward function (Chua et al., 2018; Janner et al., 2019; Chowdhury & Gopalan, 2019), however they treat it as independent from the uncertainty in the dynamics. We argue that learning an uncertainty-aware reward-dynamics function and strategically leveraging the joint uncertainty leads to the most principled and effective optimistic exploration.

3 PROBLEM STATEMENT

We consider a stochastic environment with states $s \in \mathcal{S} \subseteq \mathbb{R}^p$ and actions $a \in \mathcal{A} \subseteq \mathbb{R}^q$. Given the current state s_t and action a_t, the agent transitions to the next state s_{t+1} and incurs reward r_t with

$$\begin{pmatrix} s_{t+1} \\ r_t \end{pmatrix} = f(s_t, a_t) = \begin{pmatrix} f_t(s_t, a_t) \\ f_r(s_t, a_t) \end{pmatrix} \tag{1}$$

for transition dynamics $f_t : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ and reward function $f_r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. Our objective is to learn optimal control for this system within episodes of finite time horizon T. To control the agent, we learn a deterministic or stochastic policy π that selects actions given the current state. For ease of notation, we write the policy deterministically as $\pi : \mathcal{S} \to \mathcal{A}$ so that $a_t = \pi(s_t)$. We consider a specific transition dynamic $\hat{f}_t$ and reward function $\hat{f}_r$, denoted with a hat to indicate realizations that are drawn from random functions. The performance of the policy is measured by the sum of accumulated rewards during an evaluation episode,

$$J(\pi, \hat{f}_t, \hat{f}_r) = \left[\sum_{t=0}^{T} \hat{f}_r(s_t, a_t) \right], \quad \text{s.t.} \quad a_t = \pi(s_t), s_{t+1} = \hat{f}_t(s_t, a_t). \tag{2}$$

The optimal policy π^* maximizes performance over the true dynamics and reward function with,

$$\pi^* = \arg\max_{\pi \in \Pi} J(\pi, f_t, f_r). \tag{3}$$

In model-based RL, the true dynamic f_t is unknown and must be learned by interactions with the environment. In our setting, the ground-truth reward function f_r is also unknown.

4 BACKGROUND

In this section, we introduce relevant background on model-based reinforcement learning and existing exploration strategies.

4.1 MODEL-BASED REINFORCEMENT LEARNING

When systems have unknown dynamics, we consider the dynamics f_t to initially be random. Therefore, learning a model of the environment dynamics entails fitting an approximation $p(f_t)$ over the space of dynamics functions, given observations from the real system. The same approach can be used to estimate the true reward function f_r. We consider model-based RL algorithms that, on each iteration of training, roll out the current policy π on the real system for an episode and update $p(f)$ using the newly observed transition data. Standard approaches use the model to simulate sequential transitions and use this data to update the policy (Sutton, 1990). We opt to update the policy based on short k-length model-generated rollouts branched from the state distribution of a previous policy under the true environment, as recommended by Janner et al. (2019) to reduce compounding model errors over extended rollouts. Algorithm 1 gives an overview of the process.

Algorithm 1 Model-based Policy Optimization

```
Require: max environment steps $N$, model rollouts $M$, steps per rollout $T$, steps per model rollout $K$, initial state distribution $d(s_0)$, policy-learning algorithm PolicySearch
Initialize: policy $\pi$, reward-dynamics model $p(f)$, environment dataset $\mathcal{D}_{\text{env}}$
while $|\mathcal{D}_{\text{env}}| < N$ do
    /*Simulate Data*/
    Initialize model dataset $\mathcal{D}_{\text{model}} = \emptyset$
    for $m = 1, 2, \ldots, M$ do
        Sample $\hat{s}_0$ uniformly from $\mathcal{D}_{\text{env}}$
        for $k = 1, \ldots, K$ do
            Compute action $\hat{a}_k$ from $\pi(\hat{s}_k)$
            Select next state $\hat{s}_{k+1}$ and reward $\hat{r}_k$ using $p(f \mid \mathcal{D}_{\text{env}})$    ▷ This is algorithm-specific
            Append transition to buffer $(\hat{s}_k, \hat{a}_k, \hat{s}_{k+1}, \hat{r}_k) \rightarrow \mathcal{D}_{\text{model}}$
    /*Optimize Policy*/
    $\pi \leftarrow$ PolicySearch$(\pi, \mathcal{D}_{\text{model}})$
    /*Optimize Dynamics Model*/
    Start from initial state $s_0 \sim d(s_0)$
    for $t = 1, 2, \ldots, T$ do
        Compute action $a_t$ from $\pi(s_t)$
        Observe next state and reward $s_{t+1}, r_t = f(s_t, a_t)$
        Append transition to buffer $(s_t, a_t, s_{t+1}, r_t) \rightarrow \mathcal{D}_{\text{env}}$
    Retrain model $p(f)$ using $\mathcal{D}_{\text{env}}$
```

Model Design. We use an uncertainty-aware model to represent the prior distribution of dynamics models $p(f_t)$ aligned with the 1-step transition data in $\mathcal{D}_{\text{env}}$. Probabilistic models allow for exploiting correlations in observed data by generalizing to unseen states and actions. Popular approaches include probabilistic ensembles of neural networks, which provide mean $\mu(s, a)$ and confidence estimates $\Sigma(s, a)$ by averaging outputs over multiple trained models (Lakshminarayanan et al., 2017; Chua et al., 2018). Bayesian inference with neural networks is computationally expensive and it is an open question if parameter uncertainties lead to relevant posterior function estimates (Sharma et al., 2023; Roy et al., 2024). Alternatively, GPs provide distributions directly over the function space, enabling direct estimation of the posterior distribution $p(f_t \mid \mathcal{D}_{\text{env}})$ over dynamics models. Since GPs incorporate strong and interpretable prior beliefs, they naturally capture uncertainty due to limited data (epistemic uncertainty) and variability in the data (aleatoric uncertainty) (Rasmussen & Williams, 2006), which contributes to GPs' effectiveness in low-data regimes. GP priors, defined by a mean and covariance function, describe how the function varies around the mean. With flexible mean functions, GPs offer principled uncertainty estimates for complex functions (Iwata & Ghahramani, 2017; Fortuin et al., 2019; Fortuin, 2022).

4.2 EXPLORATION STRATEGIES

It is challenging to learn the reward-dynamics model while solving for optimal control. Performance depends on the utility of model-generated data for updating the policy and likewise the value of real experiences gathered by the policy for updating the model. Solving this problem efficiently requires balancing exploitation of current knowledge with exploration of the state-action space.

The mechanism for selecting the next model-generated state plays an important role, with different strategies incorporating varying degrees of uncertainty to encourage exploration. In this section, we provide an overview of the relevant exploration strategies, as categorized in Curi et al. (2020).

Greedy Exploitation. Most model-based RL approaches approximate $p(f_t \mid \mathcal{D}_{\text{env}})$ and maximize the policy greedily over the uncertainty in the dynamics model. They use this greedy policy:

$$\pi^{\text{Greedy}} = \arg\max_{\pi \in \Pi} \mathbb{E}_{p(f_t \mid \mathcal{D}_{\text{env}})} J(\pi, \hat{f}_t, \hat{f}_r). \tag{4}$$

For instance, GP-MPC (Kamthe & Deisenroth, 2018) and PETS (Chua et al., 2018) are MPC-based methods that use greedy exploitation and represent $p(f_t \mid \mathcal{D}_{\text{env}})$ with a GP and ensemble of neural networks respectively. Similarly, PILCO (Deisenroth & Rasmussen, 2011) uses a GP with moment matching to estimate $p(f_t \mid \mathcal{D}_{\text{env}})$ and greedy exploitation to optimize the policy. While such algorithms can converge to optimal policies under certain reward and dynamics structures (Mania et al., 2019), greedy exploitation is generally inefficient for exploration.

Thompson Sampling. This is a theoretically-grounded approach that offers a principled balance between exploration and exploitation (Russo et al., 2018; Chapelle & Li, 2011). Under Thompson sampling (Thompson, 1933), the policy is optimized with respect to a single model sample $\hat{f}$ on each episode,

$$\hat{f} \sim p(f \mid \mathcal{D}_{\text{env}}), \quad \pi^{\text{TS}} = \arg\max_{\pi \in \Pi} J(\pi, \hat{f}_t, \hat{f}_r). \tag{5}$$

This optimization problem is equivalent to greedy exploitation (equation 4) after sampling $\hat{f} \sim p(f \mid \mathcal{D}_{\text{env}})$. Under certain assumptions, Thompson sampling satisfies strong Bayesian regret bounds (Osband & Van Roy, 2016; 2017), theoretically positioning it as the optimal strategy.

Hallucinated Upper-Confidence Reinforcement Learning (H-UCRL). This is a tractable version of the UCRL algorithm (Jaksch et al., 2010), which optimizes an optimistic policy over the set of statistically plausible dynamics models. Curi et al. (2020) propose reparameterizing plausible dynamics models in order to optimize over a smaller class of variables. Specifically, they write dynamics functions as $\hat{f}_t(s, a) = \mu(s, a) + \beta\, \Sigma(s, a)\eta(s)$ for some function $\eta : \mathbb{R}^p \to [-1, 1]^p$ and optimize over $\eta(\cdot)$ instead. As this problem is still challenging to optimize, Sessa et al. (2022) propose sampling $n^j \sim \text{Uniform}([-1, 1]^p)$ for $j = 1, \ldots, Z$ and updating the policy with

$$\pi^{\text{H-UCRL}} = \arg\max_{\pi \in \Pi} \max_{\eta^j, j \in 1, \ldots, Z} J(\pi, \hat{f}_t, \hat{f}_r), \text{ s.t. } \hat{f}_t(s, a) = \mu(s, a) + \beta\, \Sigma(s, a)\eta^j. \tag{6}$$

Unlike greedy exploitation, this approach effectively optimizes an optimistic policy over the modified dynamics $\hat{f}_t$ in equation 6, although the reward function is assumed to be known. H-UCRL also does not model or utilize the joint uncertainty over transitions and rewards.

5 THE HOT-GP ALGORITHM

We introduce a new exploration strategy based on the insight that principled optimism should reason about uncertainty over transitions and rewards jointly. Importantly, this requires a joint model of the transition dynamics and reward function so that optimism about the reward can be connected to a distribution over states. Given such a model, there are many possible acquisition strategies. A natural approach uses Thompson sampling, which we adapt for optimistic exploration with our method, Hallucination-based Optimistic Thompson sampling with Gaussian Processes (HOT-GP).

5.1 MODEL STRUCTURE

Optimistic exploration implies biased exploration towards regions of the state-action space believed to yield high rewards. Thus, optimistic learning requires a reward-dynamics model that describes the correlation of state and reward. In this work, we use a Gaussian process (GP) prior to describe our belief over the product space of reward and state,

$$\hat{f} \sim \mathcal{GP}\left(\begin{pmatrix} \mu_s(\cdot) \\ \mu_r(\cdot) \end{pmatrix}, \begin{pmatrix} K_{ss}(\cdot, \cdot) & K_{sr}(\cdot, \cdot) \\ K_{sr}^{\mathrm{T}}(\cdot, \cdot) & K_{rr}(\cdot, \cdot) \end{pmatrix} \right), \tag{7}$$

where $\mu_s : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ and $\mu_r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ are the dynamics and reward mean functions and K_{ss}, K_{sr}, K_{rr} represent covariance functions $(\mathcal{S} \times \mathcal{A}) \times (\mathcal{S} \times \mathcal{A}) \to \mathbb{R}$. Traditional GPs describe outputs as conditionally independent given the input and hence factorize across the state dimensions and the reward. However, we hypothesize that this assumption is unfounded in most RL tasks and furthermore antithetical to principled optimistic exploration. Therefore, we want to explicitly model the correlation structure between the dimensions of the predicted state, and more importantly, between the predicted state and its associated reward. While this correlation can naturally be included in the GP by parameterizing the full covariance structure across examples and output dimensions, this is computationally intractable due to cubic scaling with the size of the covariance matrix (Rasmussen & Williams, 2006). Instead, we adopt a linear model of coregionalization (Grzebyk & Wackernagel, 1994; Wackernagel, 2003) where the covariance between output dimensions is restricted to be linear combinations of the input covariance. This induces a Kronecker structured covariance function that allows for efficient factorization while still meaningfully directing optimistic exploration.

In most GP regression use cases, the mean function is a constant function. While this is a valid modeling assumption for simple data, the dynamics and reward structures we encounter in RL tasks vary in a non-constant manner across different regions of the input space. To better predict these complex transition-reward relationships, we use a GP model with a multi-layer perceptron (MLP) mean function, as in Iwata & Ghahramani (2017); Fortuin et al. (2019), and use the covariance function to learn the uncertainty, representing the residual around the MLP. This flexible model can capture intricate transitions while providing useful uncertainty estimates that can be leveraged in an optimistic fashion. To learn the mean and covariance functions, we perform approximate inference by optimizing a variational lower bound on the marginal likelihood, as done by Titsias (2009). This updates the posterior distribution $p(\hat{f} \mid \mathcal{D}_{\text{env}})$, which we use to obtain the posterior predictive distribution $p(s_{t+1}, r_t \mid s_t, a_t)$. Given the current state s_t and action a_t, this multivariate Gaussian distribution reflects the model's predictions and uncertainty over the next state and reward.

5.2 USING THOMPSON SAMPLING IN THE GP ACQUISITION FUNCTION

Thompson sampling proscribes optimizing the policy using a single reward-dynamics model from the distribution of predictive models. A naive approach would sample from $p(s_{t+1}, r_t \mid s_t, a_t)$ until discovering a good model as a function of the predicted reward r_t. Instead, we observe that we can condition the reward value r_t to be high, specifically greater than some minimum percentile threshold r_{min} over the reward distribution, and sample once from this distribution. Crucially, this induces optimism into the sampling procedure because we only consider transitions where the next state is plausible under the dynamics model and whose associated reward we believe to be large.

To do this, we sample independent realizations from one non-parametric reward-dynamics model for each episode of length T, denoted as

$$\hat{f}^{(t:0\to T)} \sim \prod_{t=0}^{T} p(s_{t+1}, r_t \mid s_t, a_t, r_t > r_{\text{min}}) \tag{8}$$

for s_t sampled uniformly from $\mathcal{D}_{\text{env}}$ and $a_t = \pi(s_t)$ under the current policy. Then the optimistic Thompson sampling algorithm is

$$\hat{f}^{(0\to T)} \sim p(s_{t+1}, r_t \mid s_t, a_t, r_t > r_{\text{min}}), \quad \pi^{\text{HOT-GP}} = \arg\max_{\pi \in \Pi} J(\pi, \hat{f}_t^{(0\to T)}, \hat{f}_r^{(0\to T)}). \tag{9}$$

Performing Thompson sampling following the joint distribution in equation 9 predicts both the next state and its associated reward in an optimistic manner. We argue that this would lead to a suboptimal exploration strategy as we are only interested in optimism over the state to the extent that it relates to optimism over the reward. To address this, we instead sample reward-dynamics model realizations using a two-step approach where only the reward is explored in an optimistic manner:

1. Sample an optimistic reward $\hat{r}_t$ from $p(r_t \mid s_t, a_t, r_t > r_{\text{min}})$, a truncated normal distribution, using inverse transform sampling or another sampling technique.
2. Compute the expected transition $\hat{s}_{t+1} = \mathbb{E}_{p(s_{t+1} \mid s_t, a_t, r_t = \hat{r}_t)}[s_{t+1}]$ conditioned on the sampled optimistic reward $\hat{r}_t$.

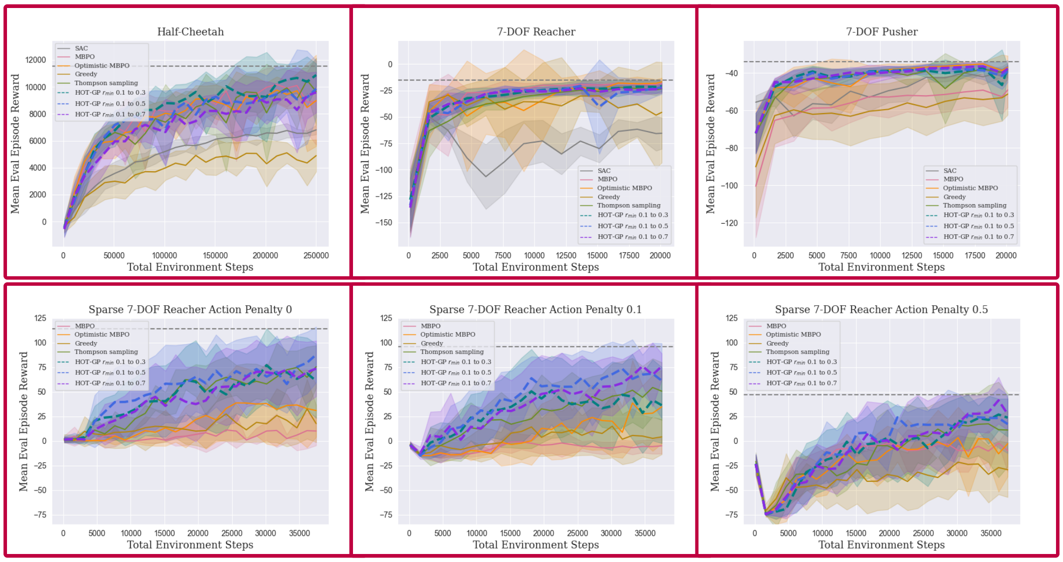

Figure 1: Learning curves on MuJoCo tasks averaged over 10 seeds. HOT-GP demonstrates equivalent or superior sample efficiency and performance for all tasks considered. The dashed line denotes SAC performance at convergence within 1,000,000 environment steps (3,000,000 for Half-Cheetah).

This approach deviates from traditional Thompson sampling, and hence allows us to disentangle optimism over the state from optimism over the reward so that we can concentrate solely on optimism over the reward from the states. This aligns well with our goal of sample-efficient reinforcement learning as the objective is to maximize the reward in as few environment steps as possible.

6 EXPERIMENTS

We evaluate the performance of our proposed algorithm, HOT-GP, on continuous state-action control tasks by measuring the mean total rewards across evaluation episodes. Since inaccurate reward estimates are likely initially, we increase r_{min} linearly throughout training to become more optimistic as the model becomes more reliable. In our experiments, we consider three variations of HOT-GP where r_{min} begins at 0.1 and increases to either 0.3, 0.5, or 0.7. We also consider the alternative Thompson sampling and greedy exploration strategies. Our Thompson sampling implementation follows the HOT-GP procedure but draws samples from the reward-conditioned distribution $p(s_{t+1} \mid s_t, a_t, r_t = \hat{r}_t)$ rather than taking the expectation. For greedy exploitation, we predict the next state and reward as the mean from the model.

First, we compare these approaches on standard MuJoCo benchmark tasks (Todorov et al., 2012) and extended MuJoCo sparse maze tasks from the D4RL suite (Fu et al., 2020) against additional model-free and model-based methods. Then we evaluate these approaches on a practical robotics task using the VMAS simulator (Bettini et al., 2022) and provide an analysis of the individual components. In-depth algorithmic descriptions of HOT-GP, along with the model-based methods we use for comparison, are given in Appendix A. Details about training are discussed in Appendix B, specific tasks are described in Appendix C, and code for reproducing the experiments can be found in the supplementary materials.

6.1 MUJOCO BENCHMARKS

Comparison Methods. In these experiments, we consider HOT-GP with varying levels of optimism using SAC (Haarnoja et al., 2018) as the underlying `PolicySearch` algorithm, as done in Janner et al. (2019) with MBPO. We compare performance against SAC, Thompson sampling, greedy exploitation, and MBPO. SAC is a model-free actor-critic algorithm that is known to achieve better sample efficiency than DDPG (Lillicrap et al., 2015) on MuJoCo tasks (Chua et al., 2018). MBPO is a model-based approach that uses an ensemble of probabilistic neural networks to model uncertainty as done in Chua et al. (2018). Predictions are drawn from a Gaussian distribution with diagonal covariance over the reward and dynamics: $p(s_{t+1}, r_t \mid s_t, a_t) = \mathcal{N}(\mu(s_t, a_t), \Sigma(s_t, a_t))$, where $\mu(s_t, a_t)$ and $\Sigma(s_t, a_t)$ are bootstrapped from the ensemble. For an optimistic baseline, we introduce an optimistic version of MBPO where the reward is sampled from $p(r_t \mid s_t, a_t, r_t > r_{\text{min}})$ and expected next state $\mathbb{E}_{p(s_{t+1}|s_t,a_t)}[s_{t+1}]$ is selected. This is the same procedure as HOT-GP, however, due to MBPO lacking a joint model over the state and reward distributions, this implies a naive optimistic exploration as there is no explicit relationship between the predicted next state and reward.

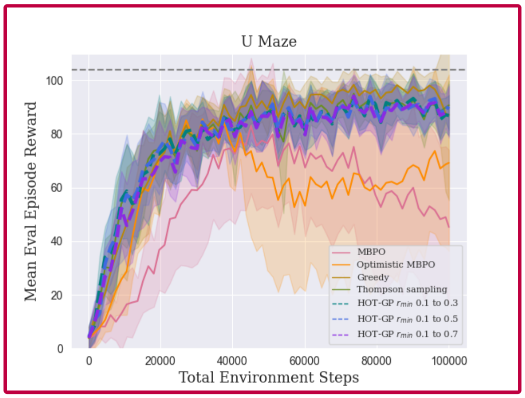

Figure 2: Learning curves on sparse maze tasks averaged over 10 seeds. HOT-GP achieves equivalent sample efficiency and performance. The dashed line denotes SAC performance at convergence within 1,000,000 environment steps.

Standard Benchmarks. The learning curves for all methods on Half-Cheetah, Reacher, and Pusher are given in the top row of Figure 1. In each case, the HOT-GP variants attain SAC-best performance with comparable or superior sample efficiency to other approaches. For instance, on Half-Cheetah, HOT-GP with r_{min} from 0.1 to 0.3 achieves the same final performance as SAC in 250,000 environment steps rather than 3 million. We observe that the deterministic approach used in greedy exploitation leads to slower learning on Half-Cheetah compared to uncertainty-aware methods, while showing no impact on performance on Pusher. This indicates that model uncertainty is key for efficient learning on Half-Cheetah, likely because the dynamics are more challenging to model, which makes model biases more detrimental to learning (Chua et al., 2018). Interestingly, uncertainty-aware learning alone is insufficient on Pusher as optimistic exploration methods enhance sample efficiency compared to MBPO and greedy exploitation. This is likely due to the task dynamics as the agent must navigate a landscape of negative rewards and sparse feedback. Optimistic exploration allows the agent to discover valuable actions more effectively by pursuing riskier options with potential to help reach the goal state. Next, we examine the impact of optimistic exploration in sparse reward settings more closely.

Sparse Reacher Task. The Sparse Reacher task provides reward $r_{\text{dist}}(s)$ that is nonzero only when the agent is within a small threshold of the target location, thus demanding clever exploration strategies to reach the target as there is no dense feedback. As in Curi et al. (2020), we introduce an action penalty of the form $r(s, a) = r_{\text{dist}}(s) - \rho \cdot \text{cost}(a)$ with the aim of penalizing random and encouraging local exploration. This is described in greater detail in Appendix C. We evaluate HOT-GP and the other comparison methods on Sparse Reacher using four different action penalty weights: $\rho = 0$, $\rho = 0.1$, $\rho = 0.3$, and $\rho = 0.5$, and present the results in the bottom row of Figure 1 as well as Figure 6 in Appendix D. HOT-GP demonstrates significantly superior sample efficiency over MBPO and moderately improved sample efficiency over optimistic MBPO in each case. The findings on Sparse Reacher with $\rho = 0$ indicate that all the HOT-GP variants perform robust optimistic exploration and successfully uncover the signal despite sparse rewards. When using the action penalty $\rho = 0.1$, HOT-GP with r_{min} from 0.1 to 0.3 performs poorly relative to higher levels of optimism. This suggests that higher levels of optimism are necessary to overcome the prior against taking large actions and explore the reward signal r_{dist}. HOT-GP is able to do this by relating the action-based penalty to the state to the overall reward through the joint model. The less efficient learning of optimistic MBPO highlights the critical importance of having a joint model of the reward and state distributions to facilitate efficient exploration. For $\rho = 0.3$ and $\rho = 0.5$, we observe that the cost term accounts for up to half of the overall reward. As a result, the HOT-GP variants explore much of the cost signal rather than r_{dist}, leading to their similar performance in these settings.

Sparse Maze Tasks. The Sparse Maze tasks require an agent to navigate through a maze to a target location, providing a reward of 1 when the agent is within a small threshold of the target location and zero in other cases. Like the Sparse Reacher task, efficiently solving such tasks demands an effective exploration strategy as there is no dense feedback to guide the agent. We consider the simpler U

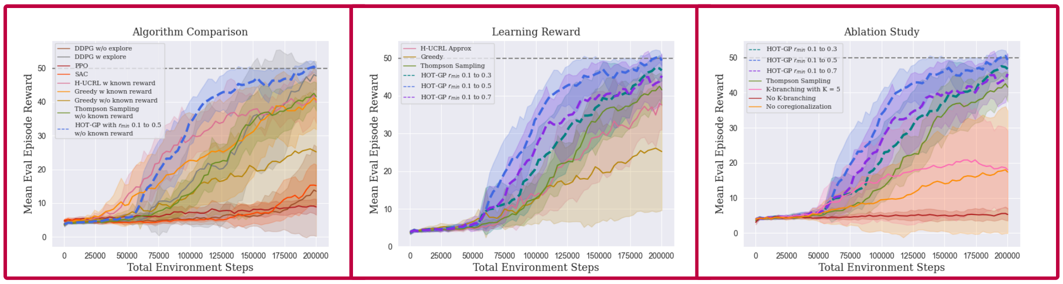

Figure 3: Learning curves in the coverage environment averaged over 10 seeds. HOT-GP achieves strong performance within 200,000 environment steps while other methods exhibit poorer sample efficiency or asymptotic performance. The dashed line denotes DDPG performance at convergence within 500,000 environment steps.

Maze and challenging Medium Maze environments with further details in Appendix C. The results of HOT-GP and the comparison methods on the U Maze are given in Figure 2 (Medium Maze is in progress). The results show that HOT-GP attains equivalent sample efficiency performance with the greedy and Thompson sampling strategies, while MBPO and Optimistic MBPO are less stable. It is likely that greedy exploitation does well on this task because of the simple dynamics.

6.2 ROBOT COVERAGE

Coverage Description. We consider a practically motivated continuous control robotics problem where an agent must visit as many unique, high valued cells as possible within a finite episode. This mirrors real-world target-tracking tasks where an agent monitors multiple regions of interest under time constraints (Robin & Lacroix, 2016; Xin et al., 2024). The reward for visiting a new 2D cell is distributed as the probability density function of a mixture of three randomly placed Gaussians over the x and y axes, so there are cluster rewards that the agent must find and exploit. To incentivize thorough coverage, the agent receives no reward for revisiting cells in an episode.

Comparison Methods. As in the MuJoCo tasks, we evaluate HOT-GP with three levels of optimism and compare against Thompson sampling and greedy exploitation. A distinction is that, since DDPG outperforms SAC in this task, we use DDPG with a probabilistic actor for the `PolicySearch` algorithm. Learning a stochastic policy allows for a baseline level of exploration, which we found to be helpful when also incorporating uncertainty induced by the reward-dynamics model. For model-free baselines, we use DDPG, SAC, and the on-policy PPO algorithm (Schulman et al., 2017). For an optimistic baseline, we use two variations of H-UCRL: one has privileged access to the ground truth reward function f_r while the other adapts to our unknown reward setting by learning the reward function $\hat{f}_r$ and applying it as in equation 6. In both cases, we use the same dynamics-reward model with H-UCRL as in HOT-GP for a fair comparison of the optimistic exploration strategies.

Performance Analysis. The first two subfigures in Figure 3 show the training curves for these approaches with and without access to the ground-truth reward function. Our method outperforms the model-based approaches in both settings and achieves DDPG-best performance marginally ahead of DDPG. The poorer performance of greedy exploitation without access to the true reward function suggests that strategic exploration is key to efficient learning for this task. Indeed, the agent must explore the environment sufficiently well enough to learn the relationship between its observations and the locations of the Gaussian centers, make informed decisions about which Gaussian center to prioritize, and realize the importance of avoiding previously seen cells. HOT-GP also demonstrates performance superior to H-UCRL, which does not converge to the optimal performance level within 200,000 environment interactions, with and without access to the ground truth reward. This may be due to infeasible optimism where the rewards associated with perturbed states $\mu(s, a) + \beta\, \Sigma(s, a)\eta$ are high but the states are not reachable in practice. HOT-GP avoids this problem by selecting states conditioned on optimistic rewards from the joint distribution, thus ensuring the states are plausible under the learned model and reward value. These findings reinforce the benefits of principled optimistic exploration for improving sample efficiency in tasks where exploration is key.

Ablation Study. Finally, we conduct an ablation study over the main components of HOT-GP to investigate their relative importance. The ablation study in Figure 3 presents the findings. We observe that scaling r_{min} from 0.1 to 0.5 yields the best results, indicating that this task benefits from a medium level of optimistic exploration. Since HOT-GP using other ranges of r_{min} values still performs fairly well, our approach does not appear to be overly sensitive to r_{min} settings. Indeed, r_{min} is an interpretable parameter that can be informed by the hypothesized level of exploration required. The ablation study in Figure 3 shows that HOT-GP without k-branching results in consistently poor performance. We also observe that k-branching with $k = 5$ leads to less optimal performance than with $k = 1$. This aligns with prior findings that accumulating errors in the dynamics model can be detrimental to learning (Janner et al., 2019; Gu et al., 2016). Furthermore, this highlights the importance of learning a faithful model to benefit from optimistic exploration, as optimism guided by an inaccurate belief harms learning. The ablation study also reveals the importance of modeling the correlation between the predicted state and reward, as this is also essential to be able to harness optimistic exploration. Lastly, sampling the next state from an optimistic distribution with Thompson sampling leads to a larger variance in performance than selecting the expected next state, thus validating our two-step sampling approach. Overall, this ablation study confirms that each component of HOT-GP is necessary to learn effectively.

7 CONCLUSION

We have introduced a principled approach to optimistic exploration based on leveraging a joint belief over the reward and state distribution. Our proposed algorithm, HOT-GP, uses a Gaussian Process to approximate the reward-dynamics and allows us to simulate plausible transitions associated with optimistic rewards, in a practical adaptation of Thompson sampling. Our experiments showed that HOT-GP achieves comparable or superior sample efficiency relative to other model-based methods and exploration strategies. Notably, we found that joint model uncertainty over outputs is crucial for effective exploration, particularly in challenging settings with sparse rewards, action penalties, and difficult-to-explore regions. Ultimately, this work establishes the importance of joint uncertainty modeling for optimistic exploration and makes a meaningful advancement towards more sample-efficient reinforcement learning.

Future Work. Future work should study how the schedule on r_{min} affects optimism and principled ways for setting this value throughout training, as we only considered a linear schedule. Another promising avenue would extend the planning horizon. We concentrated on learning 1-step lookahead reward-dynamics as HOT-GP can implicitly propagate uncertainty across the horizon by using pointwise uncertainty estimates from the model. Nevertheless, there may be benefits in explicitly propagating uncertainty to facilitate optimism over longer-term reward predictions (Seyde et al., 2021; Mehta et al., 2022; Hansen et al., 2024). Furthermore, our investigation indicates that the utility of optimistic exploration depends on the complexity of the task, although our experiments were limited to six tasks. Future research should explore the role of optimism across a broader range of problems, including in real-world scenarios with on-robot learning.

REFERENCES

Yasin Abbasi-Yadkori and Csaba Szepesvári. Regret bounds for the adaptive control of linear quadratic systems. In *Proceedings of the 24th Annual Conference on Learning Theory*, pp. 1–26. JMLR Workshop and Conference Proceedings, 2011.

Matteo Bettini, Ryan Kortvelesy, Jan Blumenkamp, and Amanda Prorok. VMAS: A vectorized multi-agent simulator for collective robot learning. *The 16th International Symposium on Distributed Autonomous Robotic Systems*, 2022.

Albert Bou, Matteo Bettini, Sebastian Dittert, Vikash Kumar, Shagun Sodhani, Xiaomeng Yang, Gianni De Fabritiis, and Vincent Moens. Torchrl: A data-driven decision-making library for pytorch. In *International Conference on Learning Representations*, 2024.

Ronen I Brafman and Moshe Tennenholtz. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct):213–231, 2002.

Paul Edmund Chang, Prakhar Verma, ST John, Arno Solin, and Mohammad Emtiyaz Khan. Memory-based dual gaussian processes for sequential learning. In *International Conference on Machine Learning*, pp. 4035–4054. PMLR, 2023.

Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. *Advances in Neural Information Processing Systems*, 24, 2011.

Sayak Ray Chowdhury and Aditya Gopalan. On kernelized multi-armed bandits. In *International Conference on Machine Learning*, pp. 844–853. PMLR, 2017.

Sayak Ray Chowdhury and Aditya Gopalan. Online learning in kernelized markov decision processes. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 3197–3205. PMLR, 2019.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Sebastian Curi, Felix Berkenkamp, and Andreas Krause. Efficient model-based reinforcement learning through optimistic policy search and planning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 14156–14170, 2020.

Marc Peter Deisenroth and Carl Edward Rasmussen. PILCO: a model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on Machine Learning*, pp. 465–472. Omnipress, 2011.

Marc Peter Deisenroth, Dieter Fox, and Carl Edward Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):408–423, 2013.

Vincent Fortuin. Priors in bayesian deep learning: A review. *International Statistical Review*, 90(3): 563–591, 2022.

Vincent Fortuin, Heiko Strathmann, and Gunnar Rätsch. Meta-learning mean functions for gaussian processes. *arXiv preprint arXiv:1901.08098*, 2019.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Yarin Gal, Rowan McAllister, and Carl Edward Rasmussen. Improving pilco with bayesian neural network dynamics models. In *ICML Workshop on Data-efficient Machine Learning*, volume 4, pp. 25, 2016.

Yarin Gal, Jiri Hron, and Alex Kendall. Concrete dropout. *Advances in Neural Information Processing Systems*, 30, 2017.

Jacob Gardner, Geoff Pleiss, Kilian Q Weinberger, David Bindel, and Andrew G Wilson. GPyTorch: Blackbox matrix-matrix gaussian process inference with GPU acceleration. *Advances in Neural Information Processing Systems*, 31, 2018.

Michel Grzebyk and Hans Wackernagel. Multivariate analysis and spatial/temporal scales: real and complex models. In *International Biometrics Conference*, volume 1, pp. 19–33, 1994.

Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep Q-learning with model-based acceleration. In *International Conference on Machine Learning*, pp. 2829–2838. PMLR, 2016.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pp. 1321–1330. PMLR, 2017.

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, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Representation Learning*, 2020.

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

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. In *International Conference on Machine Learning*, 2022.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control, 2024.

James Hensman, Alexander Matthews, and Zoubin Ghahramani. Scalable variational gaussian process classification. In *Artificial Intelligence and Statistics*, pp. 351–360. PMLR, 2015.

Yujing Hu, Weixun Wang, Hangtian Jia, Yixiang Wang, Yingfeng Chen, Jianye Hao, Feng Wu, and Changjie Fan. Learning to utilize shaping rewards: A new approach of reward shaping. In *Advances in Neural Information Processing Systems*, volume 33, pp. 15931–15941, 2020.

Julian Ibarz, Jie Tan, Chelsea Finn, Mrinal Kalakrishnan, Peter Pastor, and Sergey Levine. How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research*, 40(4-5):698–721, 2021.

Tomoharu Iwata and Zoubin Ghahramani. Improving output uncertainty estimation and generalization in deep learning via neural network gaussian processes. *arXiv preprint arXiv:1707.05922*, 2017.

Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(51):1563–1600, 2010.

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.

Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan. Provably efficient reinforcement learning with linear function approximation. In *Conference on Learning Theory*, pp. 2137–2143. PMLR, 2020.

Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model-based reinforcement learning for atari. *International Conference on Learning Representations*, 2020.

Sanket Kamthe and Marc Deisenroth. Data-efficient reinforcement learning with probabilistic model predictive control. In *International Conference on Artificial Intelligence and statistics*, pp. 1701–1710. PMLR, 2018.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in Neural Information Processing Systems*, 30, 2017.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 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.

Horia Mania, Stephen Tu, and Benjamin Recht. Certainty equivalence is efficient for linear quadratic control. *Advances in Neural Information Processing Systems*, 32, 2019.

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.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

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. IEEE, 2018.

Gergely Neu and Ciara Pike-Burke. A unifying view of optimism in episodic reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1392–1403, 2020.

Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. *Advances in Neural Information Processing Systems*, 28, 2015.

Ian Osband. Risk versus uncertainty in deep learning: Bayes, bootstrap and the dangers of dropout. In *NeurIPS Workshop on Bayesian Deep Learning*, volume 192, 2016.

Ian Osband and Benjamin Van Roy. On lower bounds for regret in reinforcement learning. *arXiv preprint arXiv:1608.02732*, 2016.

Ian Osband and Benjamin Van Roy. Why is posterior sampling better than optimism for reinforcement learning? In *International Conference on Machine Learning*, pp. 2701–2710. PMLR, 2017.

Ian Osband, Daniel Russo, and Benjamin Van Roy. (more) efficient reinforcement learning via posterior sampling. *Advances in Neural Information Processing Systems*, 26, 2013.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017.

Luis Pineda, Brandon Amos, Amy Zhang, Nathan O Lambert, and Roberto Calandra. Mbrl-lib: A modular library for model-based reinforcement learning. *arXiv preprint arXiv:2104.10159*, 2021.

Carl Edward Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*, volume 2. MIT Press Cambridge, MA, 2006.

Cyril Robin and Simon Lacroix. Multi-robot target detection and tracking: taxonomy and survey. *Autonomous Robots*, 40:729–760, 2016.

Hrittik Roy, Marco Miani, Carl Henrik Ek, Philipp Hennig, Marvin Pförtner, Lukas Tatzel, and Søren Hauberg. Reparameterization invariance in approximate bayesian inference. In *Advances in Neural Information Processing Systems*, 2024.

Daniel Russo and Benjamin Van Roy. Learning to optimize via information-directed sampling. *Advances in neural information processing systems*, 27, 2014.

Daniel J Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, Zheng Wen, et al. A tutorial on thompson sampling. *Foundations and Trends® in Machine Learning*, 11(1):1–96, 2018.

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.

Pier Giuseppe Sessa, Maryam Kamgarpour, and Andreas Krause. Efficient model-based multi-agent reinforcement learning via optimistic equilibrium computation. In *International Conference on Machine Learning*, pp. 19580–19597. PMLR, 2022.

Tim Seyde, Wilko Schwarting, Sertac Karaman, and Daniela Rus. Learning to plan optimistically: Uncertainty-guided deep exploration via latent model ensembles. In *Conference on Robot Learning*, 2021.

Mrinank Sharma, Sebastian Farquhar, Eric Nalisnick, and Tom Rainforth. Do bayesian neural networks need to be fully stochastic? In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pp. 7694–7722. PMLR, 2023.

Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine Learning Proceedings 1990*, pp. 216–224. Elsevier, 1990.

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

Csaba Szepesvári. *Algorithms for reinforcement learning*. Springer nature, 2022.

William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294, 1933.

Michalis Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *International Conference on Airtificial Inteligence and Statistical Learning*, pp. 567–574, 2009.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

Hans Wackernagel. *Multivariate geostatistics: an introduction with applications*. Springer Science & Business Media, 2003.

Jonathan Wenger, Kaiwen Wu, Philipp Hennig, Jacob R Gardner, Geoff Pleiss, and John P Cunningham. Computation-aware gaussian processes: Model selection and linear-time inference. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Pujie Xin, Zhanteng Xie, and Philip Dames. Towards predicting collective performance in multi-robot teams. *arXiv preprint arXiv:2405.01771*, 2024.

Yuxiang Yang, Ken Caluwaerts, Atil Iscen, Tingnan Zhang, Jie Tan, and Vikas Sindhwani. Data efficient reinforcement learning for legged robots. In *Conference on Robot Learning*, pp. 1–10. PMLR, 2020.

A ALGORITHM DETAILS

In this section, we provide more detailed information about the algorithms we use for evaluation. Note that the code we used to implement these algorithms are available in the code that can be found in the supplementary materials.

A.1 MODEL-FREE BASELINES

We compare model-based RL approaches with the following popular model-free algorithms:

SAC. We use a standard version of SAC as introduced in Haarnoja et al. (2018). Our implementation utilizes a Gaussian policy and automatic entropy tuning.

PPO. We also use a standard version of PPO as introduced in Schulman et al. (2017).

DDPG. We use a variant of DDPG (Lillicrap et al., 2015) with a probabilistic actor network. In standard DDPG, the deterministic nature of the policy can lead to insufficient exploration of the action space. We mitigate this tendency by using a probabilistic actor, which incorporates exploration into the policy by sampling actions from a probability distribution. Specifically, the probabilistic actor network outputs a mean $\mu_\theta(s)$ and standard deviation $\sigma_\theta(s)$ for the current state s and we select the action by sampling from the Gaussian distribution $a \sim \mathcal{N}(\mu_\theta(s), \sigma_\theta(s)^2)$. In our experiments, we compare with this version of DDPG using natural exploration under the label "DDPG w/o explore". With a deterministic policy, it is also common to facilitate exploration by adding noise to the actions. Therefore, we additionally evaluate our DDPG baseline with explicit exploration under the label "DDPG w explore". Specifically, we add Gaussian noise to the sampled action with $a = a + \zeta$ for $\zeta \sim \mathcal{N}(0, \sigma^2_{\text{explore}})$.

A.2 GREEDY EXPLOITATION

This exploration strategy follows the optimization rule given in equation 4. In particular, we hallucinate the next state as the mean output from the model and disregard the uncertainty around the prediction. In one variation, we assume access to the ground-truth reward function f_r (referred to as "Greedy w known reward") for a comparison with H-UCRL. In another variation, we also learn the reward function (referred to as "Greedy w/o known reward" in the coverage experiment and "Greedy" in the MuJoCo experiments) and similarly hallucinate rewards as the mean output from the model. In pseudocode, this approach follows Algorithm 1 exactly with the following algorithm-specific routine replacing line 10 to select the next state and reward:

- Compute the next expected state as $\hat{s}_{k+1} = \mu_s(\hat{s}_k, \hat{a}_k)$
- Observe the expected reward $\hat{r}_k = \mu_r(\hat{s}_k, \hat{a}_k)$ or true reward $\hat{r}_k = f_r(\hat{s}_k, \hat{a}_k)$

A.3 MBPO

We implement MBPO as described in Janner et al. (2019) and summarized in Section 6.1. Specifically, the model is an ensemble of probabilistic neural networks that each predict a mean output and variance (exponentiated log variance in practice) $\mu^i(s_k, a_k)$ and $\Sigma^i(s_k, a_k)$ and parameterize a multivariate Gaussian distribution with diagonal covariance $p^i(s_{k+1}, r_k \mid s_k, a_k) = \mathcal{N}(\mu^i(s_k, a_k), \Sigma^i(s_k, a_k))$. To predict an output, the parameters are bootstrapped using expectation or randomly selecting a member and the outcome is drawn from the Gaussian distribution. This is described with the following steps replacing line 10 in Algorithm 1:

- Compute the predictions for each ensemble member $\mu^i(\hat{s}_k, \hat{a}_k), \Sigma^i(\hat{s}_k, \hat{a}_k) \leftarrow \hat{f}^i(\hat{s}_k, \hat{a}_k)$
- Bootstrap predictions to get $\bar{\mu}(\hat{s}_k, \hat{a}_k), \bar{\Sigma}(\hat{s}_k, \hat{a}_k)$
- Sample outcome from the Gaussian distribution $\hat{s}_{k+1}, \hat{r}_k \sim \mathcal{N}(\bar{\mu}(\hat{s}_k, \hat{a}_k), \bar{\Sigma}(\hat{s}_k, \hat{a}_k))$

For the optimistic MBPO baseline, we maintain the ensemble model structure but perform a procedure similar to HOT-GP, following these steps to generate next states and rewards:

- Compute the predictions for each ensemble member $\mu^i(\hat{s}_k, \hat{a}_k), \Sigma^i(\hat{s}_k, \hat{a}_k) \leftarrow \hat{f}^i(\hat{s}_k, \hat{a}_k)$
- Bootstrap predictions to get $\bar{\mu}(\hat{s}_k, \hat{a}_k), \bar{\Sigma}(\hat{s}_k, \hat{a}_k)$
- Separate the reward distribution by taking its component
  $p(s_{k+1}, r_k \mid \hat{s}_k, \hat{a}_k)_{[r]} = \mathcal{N}(\bar{\mu}(\hat{s}_k, \hat{a}_k), \bar{\Sigma}(\hat{s}_k, \hat{a}_k))_{[r]}$
- Sample reward from an optimistic reward distribution
  $\hat{r}_k \sim p(s_{k+1}, r_k \mid \hat{s}_k, \hat{a}_k, \hat{r}_k > r_{\text{min}})_{[r]}$
- Select expected next state from state distribution $\hat{s}_{k+1} = \mathbb{E}_{p(s_{k+1} \mid \hat{s}_k, \hat{a}_k)_{[:r]}}[s_{k+1}]$

A.4 H-UCRL

We implement H-UCRL (Curi et al., 2020) using the sampling procedure introduced in (Sessa et al., 2022). On each model-generated rollout step, this approach hallucinates Z candidate next states that are plausible under the learned dynamics model and selects the one leading to the highest reward under f_r. Therefore, we assume access to the ground-truth reward function f_r and only learn the dynamics model f_t in our implementation of H-UCRL. In pseudocode, this approach follows Algorithm 1 exactly with the following steps replacing line 10 where we refer to $\sigma_s(s, a)$ as the standard deviation of an output from the GP dynamics model:

- Draw random perturbations $\eta^j \sim \text{Uniform}([-1, 1]^p)$ for $j = 1, \ldots, Z$

Table 1: Task-specific hyperparameter values

Hyperparameter	Half-Cheetah	Reacher	Pusher	Sparse Reacher	Coverage
Environment steps N	250,000	20,000	20,000	37,500	200,000
Steps per rollout T	1000	150	150	150	150
Model rollouts M	adaptive	adaptive	adaptive	adaptive	150
Model rollout steps K	1	1	1	1	1
Batch size B	256	256	256	256	150
Discount factor γ	0.99	0.99	0.99	0.99	0.9
Learning rate α	0.001	0.001	0.001	0.001	0.00005
Mixing factor τ	0.005	0.005	0.005	0.005	0.005
Replay buffer size	unlimited	unlimited	unlimited	unlimited	20,000

- Compute plausible next states $\hat{s}^j_{k+1} = \mu_s(\hat{s}_k, \hat{a}_k) + \beta\, \sigma_s(\hat{s}_k, \hat{a}_k)\eta^j$
- Select next state $\hat{s}_{k+1} = \arg\max_{\hat{s}^j_{k+1}} f_r(\hat{s}^j_{k+1}, \hat{a}^j_{k+1})$ for associated actions $\hat{a}^j_{k+1}$
- Observe the true reward $\hat{r}_k = f_r(\hat{s}_k, \hat{a}_k)$

We also adapt H-UCRL to our setting where the reward is unknown with a variation we call "H-UCRL Approx". We do this by learning both the dynamics and reward models and selecting the candiate state leading to the highest *predicted* reward, i.e., under $\hat{f}_r$ rather than f_r.

A.5 THOMPSON SAMPLING

This approach follows the optimization rule given in equation 5. Our Thompson Sampling implementation only differs from HOT-GP in that we sample both the reward and next state from the optimistic reward-dynamics distribution. In practice, we implement this as described in Algorithm 1 with the following algorithm-specific routine replacing line 10:

- Compute predictive posterior distribution $p(s_{k+1}, r_k \mid \hat{s}_k, \hat{a}_k)$ using $p(f \mid \mathcal{D}_{\text{env}})$
- Sample an optimistic reward $\hat{r}_k \sim p(r_k \mid \hat{s}_k, \hat{a}_k, r_k > r_{\text{min}})$
- Sample a plausible next state $\hat{s}_{k+1} \sim p(s_{k+1} \mid \hat{s}_k, \hat{a}_k, r_k = \hat{r}_k)$

A.6 HOT-GP

In HOT-GP, we sample an optimistic reward and select the most likely next state associated with the sampled reward on each step of the model-generated rollout, as described in equation 9. The steps are detailed in Algorithm 1 with the following routine replacing line 10:

- Compute predictive posterior distribution $p(s_{k+1}, r_k \mid \hat{s}_k, \hat{a}_k)$ using $p(f \mid \mathcal{D}_{\text{env}})$
- Sample an optimistic reward $\hat{r}_k \sim p(r_k \mid \hat{s}_k, \hat{a}_k, r_k > r_{\text{min}})$
- Select expected next state $\hat{s}_{k+1} = \mathbb{E}_{p(s_{k+1} \mid \hat{s}_k, \hat{a}_k, r_k = \hat{r}_k)}[s_{k+1}]$

B TRAINING

Learning Framework. We implement our learning framework with MBRL-Lib (Pineda et al., 2021) for the MuJoCo tasks and TorchRL (Bou et al., 2024) for the coverage task. Additionally, we use GPyTorch (Gardner et al., 2018) to build the GP reward-dynamics model.

Gaussian Process with Neural Network Mean Function. A standard Gaussian Process is defined by a mean function $m(x)$ and a covariance function $k(x, x')$, where x and x' are input vectors, and $m(x)$ is typically assumed to be a zero mean. In this work, we replace the traditional zero mean function with a neural network so that $m(x) = f_\theta(\mathbf{x})$, where θ are the parameters of the network. Given inputs $X = [x_1, \ldots, x_n]$ and outputs $y = [y_1, \ldots, y_n]$, the posterior distribution of this GP is

$$p(y|X, y) = \mathcal{N}(m, K + \sigma^2 I),$$

where $m = [f_\theta(x_1), \ldots, f_\theta(x_n)]$ is the mean vector of neural network outputs for each input, K is the kernel matrix $K_{ij} = k(x_i, x_j)$, σ^2 is the noise variance, and I is the identity matrix. To optimize this model, we follow the procedure in Iwata & Ghahramani (2017). This entails optimizing θ using the Mean Squared Error loss and then optimizing the kernel hyperparameters and noise variance σ^2 on the train dataset with transformed targets $y - m$, as described in the following section, on each iteration. For predicting on a new input x_*, the predictive posterior distribution is $p(y_* \mid x_*, X, y) = \mathcal{N}(y_* \mid u_*, \sigma_*)$ with:

$$\mu_* = m_* + K_*^T \left(K + \sigma^2 I\right)^{-1} (y - m)$$

$$\sigma_*^2 = K_{**} - K_*^T \left(K + \sigma^2 I\right)^{-1} K_*,$$

where $m_* = f_\theta(x_*)$, $\mathbf{K}_* = [k(x_*, x_1), \ldots, k(x_*, x_n)]^T$, $K_{**} = k(x_*, x_*)$. Thus, the neural network learns the non-zero mean function while the GP covariance function is modeled by the kernel as usual, thus allowing the model to quantify uncertainty around complex functions.

Reward-Dynamics Model Training. Our approach requires access to the full posterior distribution of the Gaussian process. To facilitate this in an efficient manner, we train the GP using a variational bound based on inducing points (Hensman et al., 2015) following the approach given in Iwata & Ghahramani (2017). This allows for efficient training using stochastic inference. To further reduce the computational cost of learning the model, we do not train on the whole dataset on each iteration and instead draw $1,000$ randomly sampled transitions from $\mathcal{D}_{\text{env}}$. This approach reduces the training cost significantly and provides an additional source of stochasticity that helps training. Using a Gaussian likelihood allows us to compute the predictive posterior in closed form, allowing us direct access to its mean and variance.

Hyperparameters. We train all algorithms using a set of hyperparameters that we found to be optimal through trial-and-error or were recommended. Table 1 describes the hyperparameter values specific to the MuJoCo and VMAS tasks we considered. We also report additional algorithm-specific hyperparameters used. In H-UCRL, we use the exploration-exploitation coefficient $\beta = 0.01$ and number of samples $Z = 5$. In MBPO and optimistic MBPO, we use an ensemble of size 7. For DDPG, we use an exploration noise σ_{explore} from 1 to 0.1. For GP-based approaches, we use the Matern kernel as the covariance function and learn from 100 inducing points. For the MuJoCo tasks, we use a 4-layer MLP with 200 hidden units per layer and SiLU activation function. For the VMAS coverage task we use a 2-layer MLP with 200 hidden units per layer and Mish activation function. We use the Adam optimizer in all cases. For further information on our implementation, please see our code under the supplementary materials.

Computation Time Comparison. We report the mean wall clock runtime and standard deviations for a single run of each algorithm on the MuJoCo tasks considered in Table 2. These results are averaged over 10 seeds. Our HOT-GP implementation incurs longer wall clock runtimes due to computational cost of GP training. However, recent research proposes promising techniques that can mitigate this overhead (Wenger et al., 2024; Chang et al., 2023). Also, note that our approach is not tied to GP models and can seamlessly generalize to alternative models such as Bayesian Neural Networks (BNNs), as the model is only needed to approximate the joint posterior distribution over next states and rewards.

C EVALUATION TASKS

In this work, we evaluate HOT-GP on five continuous control tasks: four using the MuJoCo physics engine and one using the VMAS simulator. In this section, we outline the setup for each task.

Half-Cheetah. The Half-Cheetah is a MuJoCo continuous control task where a 2D bipedal agent, resembling a simplified cheetah, must learn to run as quickly as possible. The observations are 17-dimensional vectors that include the position and velocity of the agent's torso and 6 joints. Actions are 6-dimensional vectors, where each component corresponds to the torque applied to one of the agent's joints. The reward is primarily based on the forward velocity of the agent's torso while

Table 2: Computation times in hours across tasks and algorithms

Wall Clock Runtime	**MBPO**	**Optimistic MBPO**	**Greedy**	**HOT-GP**
Half-Cheetah	24.5 ± 3.5	26.7 ± 4.88	18.5 ± 3	28.89 ± 4.1
Reacher	1.89 ± 0.1	2.04 ± 0.2	2 ± 0.1	5.18 ± 0.2
Pusher	1.87 ± 0.2	2.05 ± 0.2	1.92 ± 0.4	6.53 ± 0.1
Sparse Reacher	3.76 ± 0.3	4.1 ± 0.4	3.94 ± 0.3	8.55 ± 1.5
U Maze	12.45 ± 1.1	12.43 ± 1.7	11.75 ± 1.6	21.08 ± 3.8

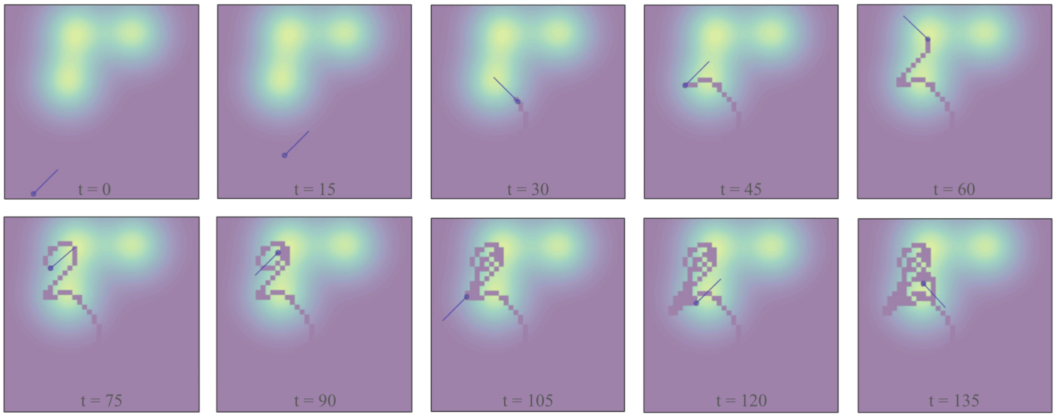

Figure 4: Frames from a rollout of a HOT-GP trained policy in the coverage environment. The agent (purple circle) drives around to cover target areas which are brightly colored. The target locations change on each episode.

also penalizing excessive torques on the joints. Let x_0 be the agent's original position, x_1 be the position after a simulation step, and dt be the time between. Then the reward is given by $r(s, a) = \frac{x_1 - x_0}{dt} - 0.1 \sum_{i=1}^{6} a_i^2$.

Pusher. The Pusher task in MuJoCo is a continuous control task where a robotic arm must learn to push an object to a target location on a 2D plane. The agent controls a three-link planar arm and must manipulate an object to move it towards a fixed goal position. The observation space is a 23-dimensional vector, which includes the position and velocity of the robot arm's joints, position and velocity of the object being pushed, and the position of the target. The action space is a 7-dimensional vector, where each component represents the torque applied to one of the joints in the robotic arm. The reward function is based on the Euclidean distance between the object and the target location, incentivizing the agent to move the object closer to the target while also penalizing large torques. Let p_{obj} be the position of the object, p_{goal} be the position of the goal, p_{arm} be the position of the arm's tip. Then the reward is given by $r(s, a) = -\|p_{\text{obj}} - p_{\text{goal}}\|_2 + 0.1 \cdot \left(-\sum_{i=1}^{7} a_i^2\right) + 0.5 \cdot (-\|p_{\text{obj}} - p_{\text{arm}}\|_2)$.

Reacher. The Reacher task in MuJoCo is a continuous control environment where a robotic arm must learn to position its end effector at a goal location in 3D space. The observations are 19-dimensional vectors that include the joint angles and velocities of the robotic arm. Actions are 6-dimensional vectors, where each component corresponds to the torque applied at one of the arm's joints. The reward incentivizes the agent to reach the goal while also penalizing excessive control actions. Letting $EE_{\text{pos}}(s)$ represent the position of the end effector and ρ be the action penalty, the reward function is $r(s, a) = -\sum_{i=1}^{3} (EE_{\text{pos}}(s)_i - \text{goal}_i)^2 - \rho \cdot \|a\|^2$.

Sparse Reacher. The Sparse Reacher is the same as the Reacher task, except with a modified reward structure. Let $EE(s)$ represent the position of the end effector, ϵ be a small threshold distance, and

ρ be the action penalty. The reward function is

$$r(s,a) = -\rho\left(\exp\left(-\sum_{i=1}^{6} a_i^2\right) - 1\right) + \begin{cases} \exp\left(-\|\mathrm{EE}(s) - \mathrm{goal}\|^2\right) & \text{if } \|\mathrm{EE}(s) - \mathrm{goal}\| < \epsilon \\ 0 & \text{otherwise.} \end{cases}$$

The first term rewards the agent based on the negative squared distance between the end effector's position and the target goal if the distance is less than ϵ and gives no reward otherwise. The second term penalizes the agent for applying excessive torques to the joints. This design creates a challenge for exploration as the agent receives minimal feedback about its performance until it is sufficiently close to the target. We take inspiration for this setup from Curi et al. (2020) although our rewards are more sparse and we use $\epsilon = 0.2$ in our experiments.

U Maze. In this task, the agent must navigate through a U-shaped maze to reach a designated target location. The agent receives a sparse reward of 1 when it comes within a small threshold distance ϵ of the target and 0 otherwise. Specifically, the reward function is

$$r(s,a) = \begin{cases} 1 & \text{if } ||\mathrm{Pos}(s) - \mathrm{goal}|| < \epsilon \\ 0 & \text{otherwise.} \end{cases}$$

The agent observes its position (x, y) and 2-dimensional goal position. Actions are 2-dimensional vectors representing velocity in the x and y dimensions.

Coverage. In this VMAS task, the agent must discover and visit unique, discretized cells with high value in a continuous state-action space. The reward for visiting a new cell at coordinates (x, y) is distributed as the probability density function of a mixture of three Gaussians over the x and y axes,

$$p(x,y) = \frac{1}{3}\sum_{i=1}^{3} \mathcal{N}((x,y) \mid (\mu_i, \Sigma)). \tag{10}$$

To incentivize thorough coverage of the regions around each Gaussian center, the agent receives no reward for revisiting cells in an episode. The agent observes its position (x, y), velocity, current value $p(x, y)$, values of neighboring cells $p(x_n, y_n)$ for $(x_n, y_n) \in \mathcal{N}(x, y)$, and the target locations $\{\mu_i\}_{i=1}^3$. The target locations are randomly generated in each episode so that the agent must learn the high-level pattern of reward distribution as a function of (x, y) and $\{\mu_i\}_{i=1}^3$, and Σ. Throughout our experiments, we use $\Sigma = 0.05$. We provide snapshots from sample rollouts on the coverage task in Figure 4.

D EXTENDED EXPERIMENTS

In this section, we extend our experiments by evaluating H-UCRL approx and MBPO without uncertainty clamping on the Half-Cheetah task, with the results given in Figure 5. We also provide learning curves for all comparison methods on the Sparse Reacher task with action penalty 0.3.

Half-Cheetah. As described in Section A, H-UCRL approx differs from the algorithm presented in Curi et al. (2020) in a few ways: (1) we use our GP model rather than a probabilistic ensemble, (2) we use the sampling-based optimistic hallucination method in equation 6 to select states, and (3) we use predicted rewards to guide optimism rather than the ground truth rewards. This version of H-UCRL does not perform well on Half-Cheetah relative to MBPO and HOT-GP, with the best seeds achieving mean evaluation episode rewards as high as 6000 after training with 250,000 environment steps. As we observed when discussing H-UCRL performance on the coverage task, this performance gap may stem from infeasible optimism, where the rewards associated with perturbed states $\mu(s, a) + \beta\, \Sigma(s, a)\eta$ are high but unreachable in practice. Additionally, a suboptimal choice of β could be a contributing factor, as we did not fine tune this parameter on Half-Cheetah due to its extended runtime and instead used the optimal β finetuned on the coverage task. HOT-GP also has a hyperparameter r_{min}, which controls the level of optimism during training. While the domain dependence of r_{min} means some adaptation is required for new tasks, we found that setting r_{min} is generally intuitive and, moreover, provides interesting insights into the nature of the task itself. For instance, if the dynamics in the domain are simple, we can be more optimistic about exploration (i.e., use a higher value of r_{min}), whereas if they are complex, we need to spend more time learning the dynamics before becoming more optimistic.

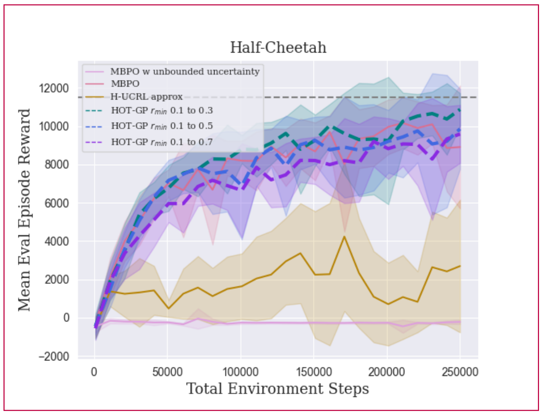

Figure 5: Learning curves for select methods on Half-Cheetah averaged over 5 seeds. The dashed line denotes SAC performance at convergence within 3,000,000 environment steps for Half-Cheetah.

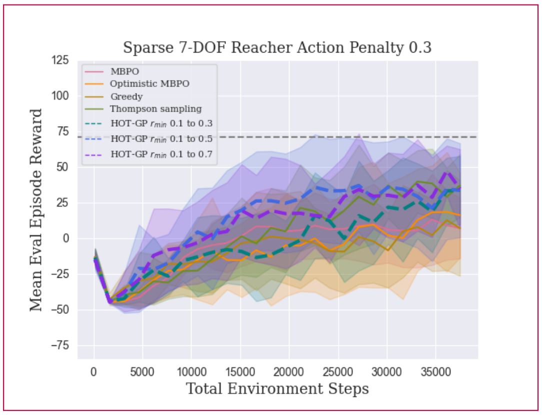

Figure 6: Learning curves on the Sparse Reacher with action penalty of 0.3. HOT-GP achieves superior sample efficiency and performance. The dashed line denotes SAC performance at convergence within 1,000,000 environment steps.

We also experimented with removing uncertainty clamping from MBPO, which resulted in a complete failure to learn as shown in Figure 5. Standard MBPO uses an ensemble of probabilistic neural networks that predicts both the mean and log variance for a given input, with log variances clamped to prevent excessive growth. Removing this clamping significantly degrades performance, indicating that the model uncertainties are poorly calibrated and heavily dependent on clamping to maintain stable learning. This finding reinforces our choice of model, as GPs provide well-calibrated uncertainties (Chowdhury & Gopalan, 2017).

Sparse Reacher. To supplement the Sparse Reacher results presented in Figure 1, we provide additional learning curves on the Sparse Reacher using action penalty of 0.3. These results are consistent with the trends observed using other action penalties and HOT-GP variants perform the best.