AN EXPERIMENTAL DESIGN PERSPECTIVE ON MODEL-BASED REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

In many practical applications of RL, it is expensive to observe state transitions from the environment. For example, in the problem of plasma control for nuclear fusion, computing the next state for a given state-action pair requires querying an expensive transition function which can lead to many hours of computer simulation or dollars of scientific research. Such expensive data collection prohibits application of standard RL algorithms which usually require a large number of observations to learn. In this work, we address the problem of efficiently learning a policy while making a minimal number of state-action queries to the transition function. In particular, we leverage ideas from Bayesian optimal experimental design to guide the selection of state-action queries for efficient learning. We propose an *acquisition function* that quantifies how much information a state-action pair would provide about the optimal solution to a Markov decision process. At each iteration, our algorithm maximizes this acquisition function, to choose the most informative state-action pair to be queried, thus yielding a data-efficient RL approach. We experiment with a variety of simulated continuous control problems and show that our approach learns an optimal policy with up to $5 - 1,000\times$ less data than model-based RL baselines and $10^3 - 10^5\times$ less data than model-free RL baselines. We also provide several ablated comparisons which point to substantial improvements arising from the principled method of obtaining data.

1 INTRODUCTION

Reinforcement learning (RL) has suffered for years from a curse of poor sample complexity. State-of-the-art model-free reinforcement learning algorithms routinely take tens of thousands of sampled transitions to solve very simple tasks and millions to solve moderately complex ones (Haarnoja et al., 2018; Lillicrap et al., 2015). The current best model-based reinforcement learning (MBRL) algorithms are better, requiring thousands of samples for simple problems and hundreds of thousands of samples for harder ones (Chua et al., 2018). In settings where each sample is expensive, even this smaller cost can be prohibitive for the practical application of RL. For example, in the physical sciences many simulators require the solution of computationally demanding spatial PDEs in plasma control (Breslau et al., 2018) or aerodynamics applications (Jameson & Fatica, 2006). In robotics, due to the difficulty of simulating more complicated objects (Heiden et al., 2021), typical RL methods operate on fast but limited rigid-body simulators (Todorov et al., 2012). These costly transition functions prompt the question: *"If we were to collect one additional datapoint from anywhere in the state-action space to best improve our solution to the task, which one would it be?"* An answer to this question can be used to guide data collection in RL.

Across the fields of black-box optimization and experimental design, techniques have been developed which choose data to collect that are particularly useful in improving the value of the objective function of the problem. For example, Bayesian optimization (BO) focuses on maximizing a function for which only a black-box model is available and queries are expensive (Frazier, 2018; Shahriari et al., 2015). More generally, Bayesian optimal experimental design (BOED) aims to choose data to collect which are maximally informative about the value of some derived quantity (Chaloner & Verdinelli, 1995). We aim to leverage these ideas for data-efficiency in reinforcement learning. Along these lines, several works in the realm of Bayesian RL address this problem in the sequential setting. Bayes-adaptive MDP (Ross et al., 2007) constructs a modified MDP by augmenting it using the posterior of the MDP, leading to a policy that can optimally trade off between acquiring more

information and exploiting the knowledge it already has. However such an MDP is intractable to exactly solve in large spaces so approximations and heuristics have been developed for the solution (Smith, 2007; Guez et al., 2012). A particularly relevant heuristic is the value of perfect information (VPI) from Dearden et al. (1998), which attempts to capture the potential change in value of a state if the value of a particular action at that state was perfectly known, specifically in a tabular setting. However, VPI doesn't attempt to distinguish between states that are visited during the execution of the optimal policy and those that aren't. This is a critical distinction when evaluating data to collect in continuous spaces as otherwise queries would be wasted learning the optimal policy in irrelevant parts of the state space.

Motivated by our opening question, in this paper we study the setting where the agent collects data by sequentially making queries to the transition function with free choice of both the initial state and the action. We refer to this setting as transition-query reinforcement learning (TQRL) and formally define it in Section 3.1. An effective algorithm for this setting would need to choose informative initial states as well as actions, which to the best of our knowledge has not been studied in the continuous MDP literature.

In this work, we draw a connection between MBRL and the world of BOED by deriving an *acquisition function* that quantifies how much information a state-action pair would provide about the optimal solution to a MDP. Like the techniques in Bayesian RL, our acquisition function is able to determine which state-action pairs are worth acquiring in a way which takes into account the reward function and the uncertainty in the dynamics. However, like the full Bayes-Adaptive MDP and unlike the VPI heuristic, this function takes into account the current estimates of which states the optimal policy will actually visit and values potential queries accordingly. It is also scalable enough to apply to multidimensional continuous control problems. In particular, our acquisition function is the expected information gain (EIG) about the trajectory taken by an optimal policy in the MDP that would be achieved if we were to query the transition function at a given state-action pair.

Finally, we assess the performance of our acquisition function as a data selection strategy in the TQRL setting. Using this method we are able to solve several continuous reinforcement learning tasks (including a nuclear fusion example) using orders of magnitude less data than a variety of competitor methods. In summary, the contributions of our paper are:

- We construct a novel acquisition function that quantifies how much information a state-action pair would provide about the optimal solution to a continuous MDP if the next state were observed from the ground truth transition function. Our function is able to select relevant datapoints for control purposes leading to much-improved data efficiency.
- We propose a practical algorithm for computing this acquisition function and use it to solve continuous MDPs in the TQRL setting.
- We evaluate the algorithm on five widely varying control tasks, where it is often orders of magnitude more sample-efficient than competitor methods and reaches similar asymptotic performance.

2 Related Work

Transition Query Reinforcement Learning

In many RL algorithms, data is collected by initializing a policy at a start state and executing actions in the environment in an episodic manner. Kearns et al. (2002) introduced the setting where the agent collects data by sampling transitions in a sequential manner from the ground truth transition model by querying at a state and action of its choice, which they refer to as *RL with access to a generative model*. We refer to this setting for brevity as TQRL. This setting is relevant in a variety of real-world applications where there is a simulator of the transition model available. We in particular see the setting in nuclear fusion research, where plasma dynamics are modeled by solving large partial differential equations where 200ms of plasma time can take up to an hour in simulation (Breslau et al., 2018).

There is substantial theoretical work on TQRL for finite MDPs. In particular, Azar et al. (2013) give matching log-linear upper and lower PAC sample complexity bounds, a substantial speedup to the upper bound for the standard problem which is quadratic in state size (Kakade, 2003). This is

achieved simply by the naive algorithm of learning a transition model by uniformly sampling the space and then performing value iteration on the estimate of the MDP for an optimal policy. More recently, the bound for this setting was tightened to hold for smaller numbers of samples by Li et al. (2020), meaning that for any dataset size in a continuous problem, the PAC performance can be quantified. Finally, Agarwal et al. (2020) show that the naive 'plug-in' estimator used in the previous works is minimax optimal for this setting. In summary, this setting is thoroughly understood for finite MDPs and it gives a sample complexity reduction from quadratic to linear in the state space size.

To our knowledge there do not exist works specifically solving the TQRL setting, though every standard RL algorithm can be applied in this setting by continuing to collect data in an online manner. In this work, we give an algorithm specifically designed for this setting in continuous MDPs. Our work empirically shows sample complexity benefits reminiscent of those theoretically shown in the tabular setting.

Exploration in Reinforcement Learning To encourage exploration in RL, agents often use an ϵ-greedy approach (Mnih et al., 2013), upper confidence bounds (UCB) (Chen et al., 2017), Thompson sampling (Osband et al., 2016), added Ornstein-Uhlenbeck action noise (Lillicrap et al., 2015), or entropy bonuses (Haarnoja et al., 2018) to add noise to a policy which is otherwise optimizing the RL objective. Though UCB, Thompson Sampling, and an entropy bonus all try to explore in a way which can adapt to the problem, they all tackle which action to take from a predetermined state and don't explicitly consider which states would be good to acquire data from.

As we mentioned, an ideal method of exploration would be to solve the intractable Bayes-adaptive MDP (Ross et al., 2007), giving an optimal tradeoff between exploration and exploitation. Kolter & Ng (2009); Guez et al. (2012) show that even approximating these techniques in the sequential setting can result in substantial theoretical reductions in sample complexity compared to frequentist PAC-MDP bounds as in Kakade (2003). Other methods stemming from Dearden et al. (1998; 1999) address this by using the myopic Value of Perfect Information as a heuristic for similar Bayesian exploration. These methods don't scale to continuous problems and don't give any method of choosing at which states to query, only actions given a current state. These methods were further extended with the development of Knowledge Gradient policies (Ryzhov et al., 2019; Ryzhov & Powell, 2011), which approximate the value function of the Bayes-adaptive MDP, and Information-Directed Sampling (Russo & Van Roy, 2014), which takes actions based on minimizing the ratio between squared regret and information gain over dynamics. This was extended to continuous-state finite-action settings in Nikolov et al. (2019). However, this work doesn't solve fully continuous problems, operates in the rollout setting rather than TQRL, and computes the information gain with respect to the dynamics rather than some notion of the optimal policy. A tutorial on Bayesian methods can be found in Ghavamzadeh et al. (2016) for further reference.

Separate from the techniques used in RL for a particular task, several methods tackle the problem of *unsupervised exploration* (Schmidhuber, 1991), where the goal is to learn as much as possible about the transition model without a task or reward function. One approach synthesizes a reward from modeling errors (Pathak et al., 2017). Another estimates learning progress by estimating model accuracy (Lopes et al., 2012). Others use an information gain-motivated formulation of model disagreement (Pathak et al., 2019; Shyam et al., 2019) as a reward. Other methods incentivize the policy to explore regions it hasn't been before using hash-based counts (Tang et al., 2017), predictions mimicking a randomly initialized network (Burda et al., 2019), or a density estimate (Bellemare et al., 2016). However, these methods all assume that there is no reward function and would be inefficient if used in that setting as they spend time exploring areas of state space which can be quickly determined to be bad for maximizing reward on a task.

Bayesian Algorithm Execution and BOED

Recently, a flexible framework known as Bayesian algorithm execution (BAX) (Neiswanger et al., 2021) has been proposed for efficiently estimating properties of expensive black-box functions, which builds off of a large literature from Bayesian Optimal Experiment Design (Chaloner & Verdinelli, 1995). The BAX framework gives a general procedure for sampling points which are informative about the future execution of an algorithm. In this paper, we extend this framework to the setting of model-predictive control, when we have expensive dynamics (i.e. transition function) which we treat as a black-box function in the BAX framework. Via this strategy, we are able to use similar techniques to develop acquisition functions for data collection in reinforcement learning.

Gaussian Processes (GPs) in Reinforcement Learning There has been substantial prior work using GPs in reinforcement learning. Most well-known is PILCO (Deisenroth & Rasmussen, 2011), which computes approximate analytic gradients of policy parameters through the GP dynamics model while accounting for uncertainty. Most related to our eventual MPC method is (Kamthe & Deisenroth, 2018), which gives a principled probabilistic model-predictive control algorithm for GPs.

3 PRELIMINARIES

In this work we deal with finite-horizon discrete time *Markov decision processes* (MDPs) which consist of a tuple $\langle \mathcal{S}, \mathcal{A}, T, r, p_0, H \rangle$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, T is the transition function mapping $T : \mathcal{S} \times \mathcal{A} \to P(\mathcal{S})$ (using the convention that $P(\mathcal{X})$ is the set of probability measures over $\mathcal{X}$), $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ a reward function, $p_0(s)$ a distribution over $\mathcal{S}$ of start states, and $H \in \mathbb{N}$ a horizon. We always assume $\mathcal{S}, \mathcal{A}, p_0, H$ are known. We also assume the reward r is known, though our development of the method can easily be generalized to the case where r is unknown. Our primary function of interest is the transition function T, which we learn from data. Our aim is to find a policy $\pi : \mathcal{S} \to P(\mathcal{A})$ that maximizes the rewards obtained by executing the policy. We can describe the execution of π in the MDP as a finite collection of random variables generated by $s_0 \sim p_0$ and $a_i \sim \pi(s_i), s_{i+1} \sim T(s_i, a_i)$ for $i \in 0, \ldots, H-1$. Then this objective can be written

$$J_T(\pi) = \mathbb{E}_{p(s_{0:H}, a_{0:H-1})} \left[\sum_{i=0}^{H-1} r(s_i, a_i, s_{i+1}) \right]. \tag{1}$$

We aim to maximize this objective while minimizing the number of samples from the ground truth transition function T that are required to reach good performance. We denote $\pi^* = \operatorname{argmax}_\pi J_T(\pi)$ as an optimal policy, which we can assume to be deterministic (Sutton & Barto, 1998) but not necessarily unique. Finally, we assume that there is some prior $P(T)$ for which the posterior $P(T \mid D)$ is available for sampling given a dataset D.

3.1 TRANSITION QUERY REINFORCEMENT LEARNING

In the online RL setting, one assumes data $D = \{(s_i, a_i, s_i')\}_{i \in [n]}$ must be collected in length-H trajectories where the initial state s_0 is sampled from p_0 and after an action a_i is chosen the next state $s_i' = s_{i+1}$ is sampled from $T(s_i, a_i)$ up to $i = H$ at which point a new initial state $s_{H+1} \sim p_0$ is sampled.

We pay special attention to the TQRL setting, where the agent sequentially acquires datapoints $x_i = (s_i, a_i, s_i')$ in arbitrary order by choosing (s_i, a_i) from $\mathcal{S} \times \mathcal{A}$ and sampling $s_i' \sim T(s, a)$ (Kearns et al., 2002; Kakade, 2003; Azar et al., 2013). Performance in this setting is still evaluated by Objective 1.

It has been shown for finite MDPs in Azar et al. (2013) that the PAC sample complexity, which is the number of samples required to identify with high probability a policy that achieves almost optimal value, of this setting is $\tilde{O}(|\mathcal{S}||\mathcal{A}|)$, ignoring the PAC factors. This is notably better than the bound of $\tilde{O}(|\mathcal{S}|^2|\mathcal{A}|)$ in the online RL setting given in Section 8.3 of Kakade (2003). The improvement shown in finite cases suggests that there could be similar reductions available in a continuous setting.

3.2 MODEL-PREDICTIVE CONTROL

Though our acquisition function is derived for any policy search procedure including planning or model-free reinforcement learning algorithms, we focus on model-predictive control (MPC) as it is simple and requires minimal components to function. MPC is a standard technique in model-based reinforcement learning (Chua et al., 2018; Wang & Ba, 2020). Using a probabilistic dynamics model $\hat{T} = \mathbb{E}[T \mid D]$ and an optimization algorithm, an MPC strategy with a planning horizon $h \in \mathbb{N}$ choosing an action at a state s_0 solves the planning problem

$$\max_{a_0, \ldots, a_h \in \mathcal{A}} \mathbb{E}_{s_{i+1} \sim \hat{T}(s_i, a_i)} \left[\sum_{i=0}^{h} r(s_i, a_i, s_{i+1}) \right]. \tag{2}$$

with a stochastic optimization technique.

Planning is typically redone periodically, often every timestep, as actions are executed in the real environment. The cross-entropy method (CEM) is most often used to solve the optimization problem. In this work, we use an improved variant of CEM described in Pinneri et al. (2020). Given sufficient computational resources and an accurate model, model-predictive control will typically find very good if not optimal actions in an MDP. In this paper we will refer to π_T as the stochastic policy obtained by running MPC over a dynamics function T. The randomness in the policy is due to any randomness in $\hat{T}$ and the randomness used by the optimizer. We give details on the hyperparameters involved in this optimization problem and their values in Section B.

4 An Acquisition Function for Model-Based RL

We draw inspiration from BO and BOED in constructing an acquisition function suitable for control applications. For our purposes an *acquisition function* is a computationally tractable function $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$ that describes the marginal contribution of collecting a transition sample at a state-action pair in improving the performance of the policy on the MDP conditioned on all previously observed data. As acquisition functions are greedy, they aren't necessarily optimal data-selection strategies given a budget in the sense that solving a Bayes-adaptive MDP is optimal. However, they are tractable and allow for good one-step choices to be made that account for all information currently available.

In this work we focus on acquisition functions written as an Expected Information Gain (EIG). These functions output the mutual information between a query of our transition model and a target random variable. The target variable should contain the minimal information required for an optimal solution of our MDP with as low a total entropy as possible. The typical framing of reinforcement learning is as a search for an optimal policy π^*. However, fully specifying such a π^* would require giving a map for states which are never visited in rollouts of π^* with start states sampled from p_0. Thus, optimizing the EIG about π^* in the TQRL setting would result in queries across areas of the state space which wouldn't ever be visited by a good policy well after they can be determined to be irrelevant.

Rather than the policy π^* defined over $\mathcal{S}$, we consider as our variable of interest the *optimal trajectory* $O = \{s_i\}_{i=1}^H$ given by a random vector of states generated by first sampling $s_0 \sim p_0$ then sampling $a_i = \pi^*(s_i), s_{i+1} \sim T(s_i, a_i)$ for H timesteps. Clearly, O is determined up to the randomness of the MDP by specifying the optimal policy π^*. O also contains the information needed about T to solve the MDP, since any state that could ever be visited by π^* is in the support of O. We don't claim O is the minimal-entropy choice of target random variable but it empirically seems to acquire the necessary data in an efficient manner.

There are three sources of uncertainty about O: the randomness in the start state distribution p_0, the randomness in the dynamics T, which is commonly called *aleatoric uncertainty*, and the uncertainty in our estimate of the model T due to our limited experience which is usually discussed as *epistemic uncertainty*. As it is impossible to reduce the first two components of uncertainty, any appropriate information gain quantity will reduce epistemic uncertainty about O as desired. In this paper, we use $\mathbb{H}$ to denote the Shannon entropy of a random variable. Finally, we can give

$$\mathrm{EIG}_O(s,a) = \mathbb{E}_{s' \sim T'(s,a|D)} \left[\mathbb{H}[O \mid D] - \mathbb{H}[O \mid D \cup \{(s,a,s')\}] \right] \tag{3}$$

$$= \mathbb{E}_{s_0 \sim p_0} \left[\mathbb{E}_{s' \sim T'(s,a|D)} \left[\mathbb{H}[O \mid D, s_0] - \mathbb{H}[O \mid D \cup \{(s,a,s')\}, s_0] \right] \right] \tag{4}$$

as our acquisition function. Here we assume a posterior model of the dynamics $T'(s, a \mid D)$ for a dataset D we have already observed. The second equality is true because $s_0 \perp s' \mid s, a$. In our work, we assume the MPC policy for the ground truth model $\pi_T \approx \pi^*$, though in principle π^* could be approximated using any method. Of course, our method never actually has access to π_T or π^*.

4.1 Estimating EIG_O via Posterior Function Sampling

For EIG_O to be of practical benefit in selecting data, we must be able to approximate it in a tractable manner. Here we show how to attain such an approximation. By the symmetry of mutual information, we can rewrite Equation 4 as

$$\mathrm{EIG}_O(s,a) = \mathbb{E}_{s_0 \sim p_0} \left[\mathbb{E}_{O \sim P(O|D)} \left[\mathbb{H}[s' \mid s, a, D, s_0] - \mathbb{H}[s' \mid s, a, O, D, s_0] \right] \right]. \tag{5}$$

Since $\mathbb{H}[s' \mid s, a, D, s_0] = \mathbb{H}[s' \mid s, a, D]$ doesn't depend on O or s_0, we can simply compute it as the entropy of the posterior predictive distribution $P(s' \mid s, a, D)$ given by our posterior over

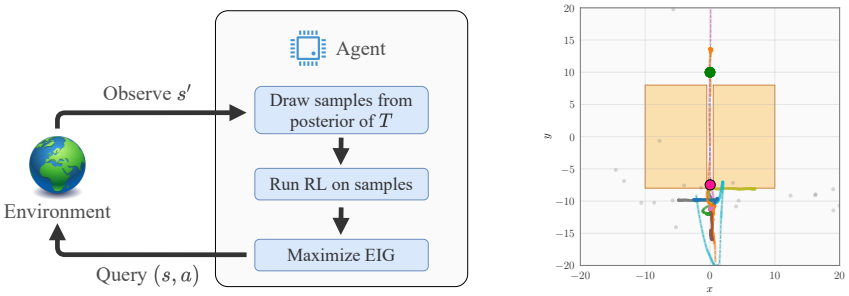

Figure 1: (a) is a schematic drawing of the BARL data-collection loop. (b) depicts EIG_O computation over several sample paths O_i in various colors sampled from $P(O \mid D)$ for a dataset consisting of past queries at the grey points. The optimizer (in pink) is a point that is maximally informative when learning a model for crossing the path between the lava pools (orange) to the goal (in green).

the transition function $P(T \mid D)$. In order to compute the other term, we must take samples $O_{ij} \sim P(O \mid D)$. To do this, we first sample m start states s_0^i from p_0 (we always set $m = 1$ in experiments but derive the procedure in general) and for each start state independently sample n posterior functions $T'_{ij} \sim P(T' \mid D)$ from our posterior over dynamics models. Then we can run the MPC procedure on each of the posterior functions from s_0^i using T'_{ij} for T and $\pi_{T'_{ij}}$ for π^* (using our assumption that $\pi^* \approx \pi_T$), giving our sampled O_{ij}. This is an expression of the generative process for O as described in the previous section that accounts for the uncertainty in T. Formally, we can approximate EIG_O via Monte-Carlo as

$$\text{EIG}_O(s, a) \approx \mathbb{H}[s' \mid s, a, D] - \frac{1}{mn} \sum_{i \in [m]} \sum_{j \in [n]} \mathbb{H}[s' | s, a, O_{ij}, D]. \tag{6}$$

Finally, we must calculate the entropy $\mathbb{H}[s'|s, a, O_i, D]$. For this, we follow a similar strategy as done in Neiswanger et al. (2021). In particular, since O_i is a set of states output from the transition model, we can treat them as additional noiseless datapoints for our dynamics model and condition on them. In the following section we describe our instantiation of this EIG estimate, and how we can use it in reinforcement learning procedures.

Though inspired by the work cited above, we modify the acquisition function to factor p_0 as an irreducible source of uncertainty. We also extend the function being queried to be vector-valued.

5 Bayesian Active Reinforcement Learning

In this work, we take the simplest possible approach for nonlinear control in continuous spaces and assume a Gaussian process prior T' to model the dynamics. Though computationally expensive, this choice ensures that we have a straightforward approximation for all necessary quantities. We assert however that the development of the acquisition function is general and any approximation of a Bayesian model could be used. In particular, we follow the approach of Wilson et al. (2020), based on sparse-GPs and random fourier approximations of kernels (Rahimi et al., 2007), allowing us to approximately but efficiently sample from the GP posterior conditioned on the observations.

We model the transition function $T : \mathcal{S} \times \mathcal{A} \to p(\mathcal{S})$ (the problem dynamics) using Gaussian processes due to their non-parametric nature and inherent ability to model uncertainties in the modeled function. The transition function takes a state action pair $(s, a) \in \mathbb{R}^{d+n}$ as input, and produces a d-dimensional output denoting the next state. We model each of the d dimensions of the output using individual GPs. More specifically we model the change in state $\Delta(s, a) = T(s, a) - s$ rather than modelling the final state $T(s, a)$ directly. This is helpful for continuous control problems since the state does not change by a huge magnitude. Given observations $D = \{(s_i, a_i, s'_i)\}$, our approach requires a posterior sample of the transition function conditioned on D. We follow the approach of Wilson et al. (2020) for fast approximate posterior sampling of GPs.

Assuming access to a generative model and an initial dataset D (which we in practice use one randomly sampled datapoint (s, a, s') for), we compute EIG_O for D by running MPC on posterior

function samples and approximate $\text{argmax}_{s\in\mathcal{S},a\in\mathcal{A}} \text{EIG}_O(s,a)$ by zeroth order approximation. Then we query $s' \sim T(s,a)$ and add the subsequent triple to the dataset D and repeat the process. To evaluate, we simply perform the MPC procedure in Equation 2 and execute $\pi_{\mathbb{E}[T'|D]}$ on the real environment. We refer to this procedure as Bayesian Active Reinforcement Learning (BARL). Details are given in Algorithm 1 (here, U denotes the uniform distribution) and a schematic diagram in Figure 1a. We discuss details of training hyperparameters and the GP model in Appendix A.

Algorithm 1 Bayesian active reinforcement learning (BARL) using EIG_O

Inputs: budget b, # of points for optimization k, # of posterior function samples n.
Initialize $(s_0, a_0) \sim U(\mathcal{S}\times\mathcal{A})$, $s_0' \sim T(s_0,a_0)$, and $D \leftarrow \{(s_0,a_0,s_0')\}$.
for $i \in [b]$ **do**
  Sample posterior functions $\{T'_\ell\}_{\ell=1}^n \sim P(T' \mid D)$.
  Sample start state $s_0 \sim p_0$.
  Compute $\{O_\ell\}_{\ell=1}^n$ by executing MPC policy $\pi_{T'_\ell}$ on the dynamics of T'_ℓ starting from s_0.
  $(x_1,y_1),\ldots,(x_k,y_k) \sim U(\mathcal{S}\times\mathcal{A})$
  $(x^*,y^*) \leftarrow \text{argmax}_{\{(x_i,y_i)\}_{i=1}^k} \text{EIG}_O(x_i,y_i)$.
  $D \leftarrow D \cup \{(x^*,y^*,x'^*)\}$ where $x'^* \sim T(x^*,y^*)$.
end for
return $\pi_{\hat{T}}$ where $\hat{T}(s,a) = \mathbb{E}_{T\sim P(T|D)}\left[T(s,a)\right]$

6 Experiments

As discussed in the introduction, the aim of our study of acquisition functions for RL is to reduce the sample complexity of learning good policies in continuous spaces. In this work, we demonstrate the effectiveness of EIG_O in allowing us to leverage transition queries by comparing the algorithm developed above to a variety of state-of-the-art RL algorithms. In particular, we compare the average return across five evaluation episodes across five runs with differing random seeds of each algorithm on five continuous control problems as data is collected. We also assess the amount of data taken by each algorithm to 'solve' the problem, which is taken to mean performing as well as our MPC procedure using the ground truth dynamics. Our method greatly outperforms other methods across the board. In particular we use $5-1,000\times$ less data to solve problems than state-of-the-art model-based algorithms and $10^3-10^5\times$ less than model-free ones. Though in this section we focus on the performance of the controller, we discuss the runtime of the algorithm in section A.1.

Comparison Methods We use as our model-based comparison methods in this work PETS (Chua et al., 2018) as implemented by Pineda et al. (2021), which does MPC using a probabilistic ensemble of neural networks and particle sampling for stochastic dynamics and a similar MPC method using the mean of the same GP model we use for BARL to execute $\pi_{\hat{T}}$ as in the standard RL setting. We also compare against PILCO (Deisenroth & Rasmussen, 2011), which also leverages a GP to directly optimize a policy that maximizes an uncertainty-aware long term reward. For model-free methods, we use Soft Actor-Critic (SAC) (Haarnoja et al., 2018), which is an actor-critic method that uses an entropy bonus for the policy to encourage evaluation, TD3 (Fujimoto et al., 2018) which addresses the stability questions of actor-critic methods by including twin networks for value and several other modifications, and Proximal Policy Optimization (PPO) (Schulman et al., 2017), which addresses stability by forcing the policy to change slowly in KL so that the critic remains accurate. As a baseline TQRL method and to better understand the GP performance, we use a method we denote EIG_T, which chooses points that maximize the predictive entropy of the transition model to collect data. We believe that when given access to transition queries many unsupervised exploration methods like Pathak et al. (2019); Shyam et al. (2019) or methods which value information gain over the transtion function (Nikolov et al., 2019) would default to this behavior.

Control Problems. We tackle five control problems: the standard underactuated **pendulum** swing-up problem (Pendulum-v0 from Brockman et al. (2016)), a **cartpole** swing-up problem, a 2D **lava path** navigation problem, a 2-DOF robot arm **reacher** problem with 8-dimensional state (Reacher-v2 from Brockman et al. (2016)), and a simplified **beta tracking** problem from plasma control where the controller must maintain a fixed normalized plasma pressure using as GT dynamics a model learned similarly to Abbate et al. (2021). Though this is very much a simplification of the problem, we intend to extend and apply BARL to more realistic settings in the immediate future. The lava

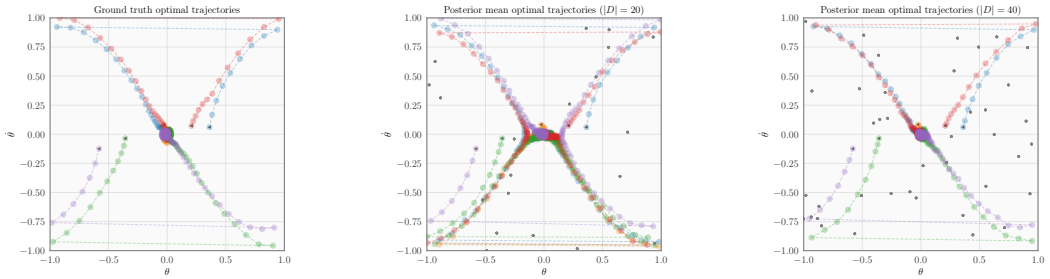

Figure 2: Progress and sampled points of BARL. These give trajectories through the state space of the pendulum problem from four fixed start points for the optimal controller and MPC with 20 and 40 datapoints, respectively. It is clear that even with very few points the controller is able to closely track the optimal paths. Here color is used only to disambiguate the trajectories.

Environment	BARL	MPC	EIG_T	PETS	SAC	TD3	PPO
Lava Path	**11**	41	41	600	n/a	n/a	n/a
Pendulum	**16**	46	46	5,200	6,000	57,000	13,000
Cartpole	**91**	161	121	1,625	31,000	18,000	n/a
Beta Tracking	96	**36**	n/a	300	9,000	6,000	16,000
Reacher	**251**	751	n/a	700	23,000	13,000	n/a

Table 1: **Sample Complexity:** Median number of samples across 5 seeds required to reach 'solved' performance averaged across 5 trials on our control environments. 'Solved' is determined by operating an MPC policy similar to the one used for evaluation in many methods using the ground truth dynamics to predict actions. 'n/a' is recorded when the median run didn't solve the problem by the end of our training.

path is intended to test stability and exploration of algorithms. The goal is to reach a fixed goal state from a relatively narrow uniform distribution over start states. As shown in Figure 1b, the state space contains a 'lava' region which gives very large negative rewards for every timestep. Other than when in lava, the reward is simply the negative squared distance to the goal, forcing the agent to navigate to the goal as quickly as possible. Since the lava has a narrow path through, the actor is forced to explore a policy which will realize that it is safe and efficient to cross. Agents who fail to find this solution will be forced to go around, incurring penalties.

We see in both the sample complexity figures in Table 1, the learning curves in Figure 3, and visually in Figure 2, that BARL leverages EIG_O to greatly reduce the data requirements of learning controllers on the problems presented. We'd like to additionally point out several failure cases of related algorithms that BARL avoids. Though it performs well on the simplest environments (pendulum and cartpole), EIG_T suffers from an inability to focus on acquiring data relevant to the control problem and not just learning dynamics as the state space becomes higher-dimensional in the reacher problem, or less smooth as in the beta tracking problem. The MPC method performs well across the board and is competitive with BARL on the plasma problem but suffers from many more reqired samples in the smaller environments where the model uncertainty can point to meaningfully underexplored areas. PETS is strong across the board but suffers from more required samples due to both its neural network dynamics model and its inability to make transition queries.

All algorithms besides BARL suffer substantial instability on the lava path problem, which is designed to be challenging to explore in a sequential fashion and require a precise understanding of which areas are safe to enter. BARL manages to learn where it is safe to operate in a handful of queries, which is an exciting result and will bear further investigation. Figure 1b gives some intuition as to why: points are initally queried close to the start and as those dynamics are understood they are subsequently queried farther and farther along the execution paths. This allows BARL to use transition queries to avoid traversing well-understood areas of state space to reach the areas which are worth learning. We see a speedup in sample complexity reminiscent of a move from quadratic to linear, which mirrors some of the theoretical improvements given in the prior work discussed on tabular methods.

We further support our assertion that BARL is picking 'meaningful' points to the control problem by the evidence in Figure 4. Here, BARL is able to solve the reacher problem while EIG_T is not.

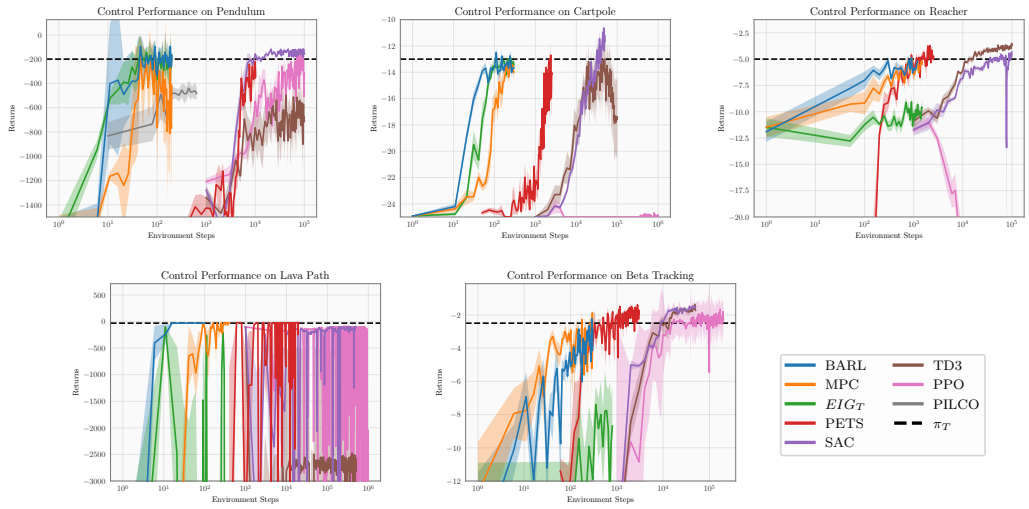

Figure 3: Learning Curves of RL methods on our control tasks giving control performance for a given amount of data average across 5 seeds. In all of these, the x-axis is on a logarithmic scale to account for widely varying data requirements. We see that though most algorithms end up reaching roughly the same performance on each task, BARL is substantially more efficient in most cases. The shaded region is the standard error of the average performance across the 5 seeds.

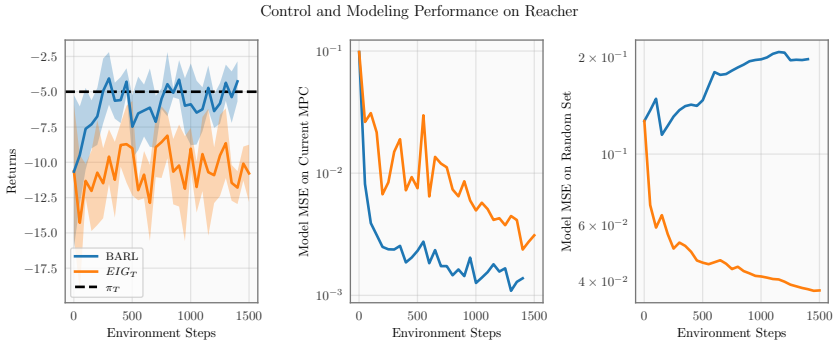

Figure 4: For a single run of BARL and of EIG_T using the same prior model, we evaluate control performance (by average returns over 5 trials) on the predictions used by the MPC procedure, and modeling error (by MSE) on a set of points uniformly sampled from the state-action space.

However, it has much worse model predictions on the random datapoints than EIG_T while doing a much better job modelling the datapoints used by the MPC procedure. Clearly, the EIG_O acquisition function captures in some way which data would be valuable to acquire to not just learn about the transition function but actually solve the control problem. We see this pattern across other tasks as well. In section B.1, we study whether the acquisition function we see here is able to work with a suboptimal controller on posterior samples of the dynamics. Our experiments show that EIG_O seems to work well even when the policy used to generate O is subpoptimal.

7 Discussion and Future Work

In this work, we proposed an acquisition function for reinforcement learning, and were able to apply to the setting of TQRL leading to a novel algorithm for addressing the problem of efficient data collection in RL. We experimented with several control problems and demonstrated that this approach leads to substantial improvements with respect to the sample efficiency.

However, there are some drawbacks to this method as well. Computing our proposed acquisition function relies on executing the entire control algorithm over a set of posterior function samples leading to higher computational requirements. In our current implementation we use GPs to model the transition function, which are computationally expensive and do not scale well to higher dimensions and large datasets. In the future, we plan to extend this idea to use other types of Bayesian models such as Bayesian neural networks and take advantage of GPU compute for better scalability.

REFERENCES

J. Abbate, R. Conlin, and E. Kolemen. Data-driven profile prediction for DIII-d. *Nuclear Fusion*, 61 (4):046027, mar 2021. doi: 10.1088/1741-4326/abe08d. URL https://doi.org/10.1088/1741-4326/abe08d.

Alekh Agarwal, Sham Kakade, and Lin F. Yang. Model-based reinforcement learning with a generative model is minimax optimal. In Jacob Abernethy and Shivani Agarwal (eds.), *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pp. 67–83. PMLR, 09–12 Jul 2020. URL https://proceedings.mlr.press/v125/agarwal20b.html.

Mohammad Gheshlaghi Azar, Rémi Munos, and Hilbert J Kappen. Minimax pac bounds on the sample complexity of reinforcement learning with a generative model. *Machine learning*, 91(3): 325–349, 2013.

Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Rémi Munos. Unifying count-based exploration and intrinsic motivation. *CoRR*, abs/1606.01868, 2016. URL http://arxiv.org/abs/1606.01868.

Joshua Breslau, Marina Gorelenkova, Francesca Poli, Jai Sachdev, Alexei Pankin, Gopan Perumpilly, and USDOE Office of Science. Transp, 6 2018. URL https://www.osti.gov/biblio/1489900.

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

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=H1lJJnR5Ym.

Kathryn Chaloner and Isabella Verdinelli. Bayesian experimental design: A review. *Statistical Science*, pp. 273–304, 1995.

Richard Y. Chen, Szymon Sidor, Pieter Abbeel, and John Schulman. UCB and infogain exploration via q-ensembles. *CoRR*, abs/1706.01502, 2017. URL http://arxiv.org/abs/1706.01502.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/3de568f8597b94bda53149c7d7f5958c-Paper.pdf.

Richard Dearden, Nir Friedman, and Stuart Russell. Bayesian q-learning. In *Aaai/iaai*, pp. 761–768, 1998.

Richard Dearden, Nir Friedman, and David Andre. Model-based bayesian exploration. *CoRR*, abs/1301.6690, 1999. URL http://arxiv.org/abs/1301.6690.

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 International Conference on Machine Learning*, ICML'11, pp. 465–472, Madison, WI, USA, 2011. Omnipress. ISBN 9781450306195.

Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1587–1596. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/fujimoto18a.html.

Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, and Aviv Tamar. Bayesian reinforcement learning: A survey. *CoRR*, abs/1609.04436, 2016. URL http://arxiv.org/abs/1609.04436.

Arthur Guez, David Silver, and Peter Dayan. Efficient bayes-adaptive reinforcement learning using sample-based search. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pp. 1025–1033, Red Hook, NY, USA, 2012. Curran Associates Inc.

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.

Eric Heiden, Miles Macklin, Yashraj S Narang, Dieter Fox, Animesh Garg, and Fabio Ramos. DiSECt: A Differentiable Simulation Engine for Autonomous Robotic Cutting. In *Proceedings of Robotics: Science and Systems*, Virtual, July 2021. doi: 10.15607/RSS.2021.XVII.067.

Antony Jameson and Massimiliano Fatica. Using computational fluid dynamics for aerodynamics, 2006.

Sham Machandranath Kakade. *On the sample complexity of reinforcement learning*. University of London, University College London (United Kingdom), 2003.

Sanket Kamthe and Marc Deisenroth. Data-efficient reinforcement learning with probabilistic model predictive control. In Amos Storkey and Fernando Perez-Cruz (eds.), *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pp. 1701–1710. PMLR, 09–11 Apr 2018. URL https://proceedings.mlr.press/v84/kamthe18a.html.

Michael Kearns, Yishay Mansour, and Andrew Y Ng. A sparse sampling algorithm for near-optimal planning in large markov decision processes. *Machine learning*, 49(2):193–208, 2002.

J Zico Kolter and Andrew Y Ng. Near-bayesian exploration in polynomial time. In *Proceedings of the 26th annual international conference on machine learning*, pp. 513–520, 2009.

Gen Li, Yuting Wei, Yuejie Chi, Yuantao Gu, and Yuxin Chen. Breaking the sample size barrier in model-based reinforcement learning with a generative model. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 12861–12872. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/96ea64f3a1aa2fd00c72faacf0cb8ac9-Paper.pdf.

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.

Manuel Lopes, Tobias Lang, Marc Toussaint, and Pierre-yves Oudeyer. Exploration in model-based reinforcement learning by empirically estimating learning progress. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/a0a080f42e6f13b3a2df133f073095dd-Paper.pdf.

David JC MacKay et al. Bayesian nonlinear modeling for the prediction competition. *ASHRAE transactions*, 100(2):1053–1062, 1994.

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

Radford M Neal. *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto, 1995.

Willie Neiswanger, Ke Alexander Wang, and Stefano Ermon. Bayesian algorithm execution: Estimating computable properties of black-box functions using mutual information. In *International Conference on Machine Learning*. PMLR, 2021.

Nikolay Nikolov, Johannes Kirschner, Felix Berkenkamp, and Andreas Krause. Information-directed exploration for deep reinforcement learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Byx83s09Km.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/8d8818c8e140c64c743113f563cf750f-Paper.pdf.

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.

Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5062–5071. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/pathak19a.html.

Luis Pineda, Brandon Amos, Amy Zhang, Nathan O. Lambert, and Roberto Calandra. Mbrl-lib: A modular library for model-based reinforcement learning. *Arxiv*, 2021. URL https://arxiv.org/abs/2104.10159.

Cristina Pinneri, Shambhuraj Sawant, Sebastian Blaes, Jan Achterhold, Joerg Stueckler, Michal Rolinek, and Georg Martius. Sample-efficient cross-entropy method for real-time planning. *arXiv preprint arXiv:2008.06389*, 2020.

Ali Rahimi, Benjamin Recht, et al. Random features for large-scale kernel machines. In *NIPS*, volume 3, pp. 5. Citeseer, 2007.

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

Daniel Russo and Benjamin Van Roy. Learning to optimize via information-directed sampling. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/301ad0e3bd5cb1627a2044908a42fdc2-Paper.pdf.

Ilya O Ryzhov and Warren B Powell. Information collection on a graph. *Operations Research*, 59(1): 188–201, 2011.

Ilya O Ryzhov, Martijn RK Mes, Warren B Powell, and Gerald van den Berg. Bayesian exploration for approximate dynamic programming. *Operations research*, 67(1):198–214, 2019.

Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pp. 222–227, 1991.

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

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1): 148–175, 2015.

Pranav Shyam, Wojciech Jaśkowski, and Faustino Gomez. Model-based active exploration. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5779–5788. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/shyam19a.html.

Trey Smith. *Probabilistic planning for robotic exploration*. Carnegie Mellon University, 2007.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 1998. ISBN 0-262-19398-1. URL http://www.cs.ualberta.ca/%7Esutton/book/ebook/the-book.html.

Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. #exploration: A study of count-based exploration for deep reinforcement learning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 2753–2762, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3a20f62a0af1aa152670bab3c602feed-Abstract.html.

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.

Tingwu Wang and Jimmy Ba. Exploring model-based planning with policy networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1exf64KwH.

Christopher KI Williams and Carl Edward Rasmussen. Gaussian processes for regression. 1996.

James Wilson, Viacheslav Borovitskiy, Alexander Terenin, Peter Mostowsky, and Marc Deisenroth. Efficiently sampling functions from gaussian process posteriors. In *International Conference on Machine Learning*, pp. 10292–10302. PMLR, 2020.

Control Problem	Pendulum	Cartpole	Lava Path	Reacher	Beta Tracking
Budget b	200	300	100	1500	300
# of points for optimization k	1,000	1,000	1,000	1,000	1,000
# of posterior function samples n	15	15	15	15	15

Table 2: BARL hyperparameters used for each control problem when run on the author's 24-core CPU machines.

Control Problem	Pendulum	Cartpole	Lava Path	Reacher	Beta Tracking
Sample O n times	20.7-18.5	38.7-33.9	33.4-35.8	216-276	7.5-4.9
Evaluate EIG_O at k points	3.3-5.5	7.5-12.7	7.6-9.3	21.3-147	7.5-13.6
Total for BARL Iteration	24-24.04	46.2-46.5	40.1-45.1	237-423	15-18.5
Evaluation for one episode	7.2-21.5	2.5-10.3	18-47.9	26.2-913.7	0.9-3.5

Table 3: Runtime in seconds for the phases of the BARL algorithm on all problems. The ranges given show the runtime for the operation at the beginning and at the end of training, as some operations run longer as more data is added.

A Training Details

We vary our budget based on our understanding of how much data would be required to solve the problem. The other hyperparameters of the BARL algorithm are constant but listed for completeness in Table 2.

For all of our experiments, we use a squared exponential kernel with automatic relevance determination (MacKay et al., 1994; Neal, 1995). The parameters of the kernel were estimated by maximizing the likelihood of the parameters after marginalizing over the posterior GP (Williams & Rasmussen, 1996).

To optimize the transition function, we simply sampled a set of points from the domain, evaluated the acquisition function, and chose the maximum of the set. This set was chosed uniformly for every problem but Reacher, for which we chose a random subset of $\cup_i \cup_j O_{ij}$ since the space of samples is 10-dimensional and uniform random sampling will not get good coverage of interesting regions of the state space.

A.1 Runtime Details

Based on these choices and the MPC hyperparameters below in Section B, each of these problems results in a varying runtime for the BARL algorithm. In Table 3, we report the time taken for an iteration of BARL and how it breaks down by step. We give these as ranges, as the computational time requires increases as the learning process continues since GP computational costs scale with the size of the dataset. We also include for completeness the time taken to execute the MPC policy on the ground truth problem, which is not strictly part of the BARL algorithm but still relevant to practitioners.

Clearly, BARL is a relatively slow algorithm computationally. But in settings where samples are scarce, BARL is much cheaper than alternative methods which might use less compute for the RL algorthms but require many more samples. When compared to the costs of running an hour-long simulation or running a costly experiments, spending a few minutes computing the acquisition function seems like a good use of resources.

B MPC Details

As we've discussed, we use model-predictive control in this work to choose actions which maximize future reward given a model of the dynamics. In particular we use the improved Cross-Entropy Method from Pinneri et al. (2020) to solve the optimization problem in Equation 2, which uses several

Control Problem	Pendulum	Cartpole	Lava Path	Reacher	Beta Tracking
Base number of samples	25	30	25	100	25
Number of elites	3	6	4	15	3
Planning horizon	20	15	20	15	5
Number of iCEM iterations	3	5	3	5	3
Replanning Period	6	1	6	1	2

Table 4: Hyperparameters used for optimization in MPC procedure for control problems.

tricks including colored noise samples and caching to reduce the number of queries to the planning model. There is a natural trade-off in any search method between computational cost and quality of actions found in terms of predicted reward. In this work, we chose hyperparameters for each task that were as computationally light as possible which attained a similar reward to larger hyperparameters when executing MPC using the ground-truth model (π_T, in our terms). As recommended by the original paper, we use $\beta = 3$ for the scaling exponent of the power spectrum density of sampled noise for action sequences, $\gamma = 1.25$ for the exponential decay of population size, and $\xi = 0.3$ for the amount of caching.

We manually tuned the base number of samples, planning horizon, number of elites to take from the sampled action sequences, number of iterations of planning, and the replanning period of the model. Here we give the ultimate values for those parameters, which were used for all ablations using our GP model and MPC. The values we used across all experiments for each problem are given in Table 4.

B.1 Robustness of EIG_O to a suboptimal controller

In order to compute EIG_O in this work, we perform model-predictive control on posterior transition function samples (execute $\pi_{T'_\ell}$ on T'_ℓ, in our notation). We assume that $\pi_{T'_\ell}$ is close to the optimal policy for the MDP with transition function T'_ℓ. However, this assumption could lead to pathologies in the method if it doesn't hold in practice. In this section, we emprically investigate the consequences of using a suboptimal controller when finding samples of O on posterior samples of the transition function.

In order to understand the sensitivity of EIG_O to the MPC policy executed on posterior samples, we ran experiments where we reduced the planning budget or horizon for the posterior function policy in order to see whether the acquisition function fails. In particular, we ran the reacher and cartpole experiments from the main paper with varying MPC budgets for the posterior function policy $\pi_{T'_\ell}$ and a fixed MPC budget at test time. This allows us to isolate the effect of a subpoptimal policy generating samples of O.

On the reacher experiment, we vary the number of CEM iterations ranging from 1 to 5. This is straightforwardly linked to the amount of search the policy conducts before executing an action. On the cartpole experiment, we varied the length of the planning horizon, which affects how far in the future the policy will consider actions as it is deciding what is the good immediate next action. In both cases, we see in Figure 5 that performance hardly changes as the budget for MPC is reduced. Only on the reacher problem when the number of CEM iterations is reduced to 1 (effectively reducing CEM to simple random search) do we see a significant drop in performance. This supports the notion that the quality of the approximation of the optimal policy in EIG_O is not critical to the performance of the acquisition function as a data selection strategy. We also plot in the figure the performance of an MPC controller on the ground truth dynamics with these reduced MPC budgets. It is clear that if we were to execute these degraded policies at test time, they would be much worse. We find it interesting and intend to further study the robustness of using cheaper policies to decide where to acquire data.

C Description of Continuous Control Problems

Lava Path. The lava path has 4-dimensional state (position and velocity) and 2-dimensional action (an applied force in the plane). The goal is to reach a fixed goal state from a relatively narrow uniform

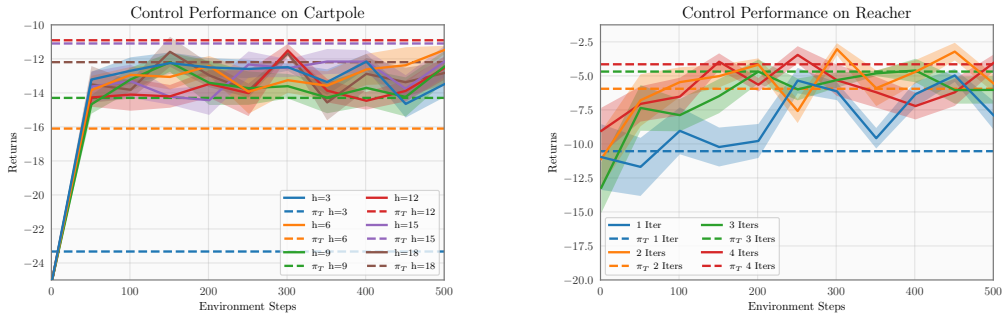

Figure 5: Performance of BARL when MPC budget for posterior function samples is varied while MPC test time budget is held constant. The error regions are the standard error of the return seen across 5 trials of the policy. The dashed lines are the performances that MPC with the equivalent hyperparameters achieves if executed at test time given the ground truth dynamics.

distribution over start states. As shown in Figure 1b, the state space contains a 'lava' region which gives very large negative rewards for every timestep. Other than when in lava, the reward is simply the negative squared distance to the goal, forcing the agent to navigate to the goal as quickly as possible. Since the lava has a narrow path through, the actor is forced to explore a policy which will realize that it is safe and efficient to cross. Agents who fail to find this solution will be forced to go around, incurring penalties.

Pendulum. The pendulum swing-up problem is the standard one found in the OpenAI gym (Brockman et al., 2016). The state space contains the angle of the pendulum and its first derivative and action space simply the scalar torque applied by the motor on the pendulum. The challenge in this problem is that the motor doesn't have enough torque to simply rotate the pendulum up from all positions and often requires a back-and-forth swing to achieve a vertically balanced position. The reward function here penalizes deviation from an upright pole and squared torque.

Cartpole. The cartpole swing-up problem has 4-dimensional state (position of the cart and its velocity, angle of the pole and its angular velocity) and a 1-dimensional action (horizontal force applied to the cart). Here, the difficulty lies in translating the horizontal motion of the cart into effective torque on the pole. The reward function here is a negative sigmoid function penalizing the distance betweent the tip of the pole and a centered upright goal position.

Reacher. The reacher problem simulates a 2-DOF robot arm aiming to move the end effector to a randomly resampled target provided. The problem requires joint angles and velocities as well as an indication of the direction of the goal, giving an 8-dimensional state space with the mentioned 2-D control. Our results on this problem are particularly encouraging as they show that BARL can scale to some problems with higher dimensionalities.

Beta Tracking (Nuclear Fusion). Finally, the beta tracking problem has 4-dimensional state consisting of the current normalized plasma performance β_N in the DIII-D tokamak. β_N is given by an appropriately normalized ratio between the plasma pressure and the magnetic pressure and is a common figure of merit in fusion energy research. In addition to β_n the state space contains its most recent change as well as the current power injection level and its most recent change. The action is the next change in the power injection level. The "ground-truth" dynamics for this problem are given by a neural network model learned from data processed as in Abbate et al. (2021). Control is done at a timestep of 200ms and the reward function is the negative absolute deviation from $\beta_n = 2$. Reliably controlling plasmas to sustain high performance is a major goal of research efforts for fusion energy, and though this is very much a simplification of the problem, we intend to extend and apply BARL to more realistic settings in the immediate future.