Thesis

Sample-Efficient Reinforcement Learning with applications in Nuclear Fusion

Viraj Mehta

CMU-RI-TR-23-86
December 7, 2023

Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Jeff Schneider (chair)
David Held
Deepak Pathak
Stefano Ermon (Stanford University)
Mark D. Boyer (Commonwealth Fusion Systems)

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This work was funded in part by DOE grant number DE-SC0021414.

Keywords: reinforcement learning, dynamical systems, experimental design, plasma control, nuclear fusion, machine learning

Abstract

In many practical applications of reinforcement learning (RL), it is expensive to observe state transitions from the environment. In the problem of plasma control for nuclear fusion, the motivating example of this thesis, determining 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 thesis, I address the problem of efficiently learning a policy from a relatively modest number of observations, motivated by the application of automated decision making and control to nuclear fusion. The first section presents four approaches developed to evaluate the prospective value of data in learning a good policy and discusses their performance, guarantees, and application. These approaches address the problem through the lenses of information theory, decision theory, the optimistic value gap, and learning from comparative feedback. We apply this last method to reinforcement learning from human feedback for the alignment of large language models. The second presents work which uses physical prior knowledge about the dynamics to more quickly learn an accurate model. Finally, I give an introduction to the problem setting of nuclear fusion, present recent work optimizing the design of plasma current rampdowns at the DIII-Dtokamak, and discuss future applications of AI in fusion.

Acknowledgements

I started my research career inauspiciously. Starting with the wrong motivations (getting into college), the wrong field for me (molecular biology), the wrong time (summers in high school), I was asked to thaw frozen tubes of ICU patient urine with my hands so that we could later centrifuge them and measure the melatonin levels present. Luckily, I found another lab at which we grew plants, applied chemical treatments of adenosine triphosphate and the like, and measured the effect of these hormones on the aperture of stomata. The PI, Greg Clark, was generous with his time and willing to help even very young students attempt to do science. I am grateful to him for showing me that persistence is the key element in research and for his belief that teenagers could be useful in a lab setting.

When I showed up as an undergrad to Stanford in the fall of 2014, I thought I would be a physicist or a mechanical engineer. I had fallen out of love with mathematics in high school and (besides a notable road trip spent learning to program a TI-73 calculator in BASIC) had never fallen in love with computers to start with. At the same time, I'd fallen in love with physics due to a pair of fantastic physics teachers: Mark and Nancy Misage. Though I only ever took one physics class (a fantastic course on modern physics given by Carl Wieman) after high school, these two gave me the fundamental understanding needed to have some intuition about the processes that drive tokamak physics and plasma control. I sincerely thank them for this. Through quirks of scheduling, social pressure, and some vague notion that I should start with broad and foundational classes I started with the math and computer science curriculum instead of those disciplines.

At Stanford, I learned the two foundational skills on which the bulk of my research sits: to program and to prove. For that I thank many professors in the Mathematics and Computer Science departments there. In particular my undergrad advisor Jan Vondrak was extremely patient with the various startups and diversions I distracted myself with while ultimately failing to write an honors thesis. In the summer of 2015, I interned at Google and learned that I didn't much like being told what to program outside of a pedagogical setting. So, I decided to run around Google and learn how the folks with jobs I thought were interesting got to those jobs. The common thread amongst the folks was that they had all gotten PhDs. So, I decided to jump back into research.

I joined Silvio Savarese's lab and came to the realization that drove me to this point: you can never *run out of research*. Until that point I'd never entirely intellectually engaged with the unbounded nature of the work but instead had been going through the motions. By working with Silvio, Animesh Garg, Andrey Kurenkov, Kuan Fang, JunYoung Gwak, Chris Choy, Yuke Zhu, Danfei Xu, and others at Stanford, I began to understand some of the key truths I continue to work to grasp: that asking good questions is at least as important as answering them, that it's critical to know when to kill a project, and that a cool method that doesn't scale is probably going to be worse than a simple method that does. I am deeply grateful for their mentorship and patience. I can still recall repeatedly screwing up an implementation of a projection from a depth image into a point cloud, only to learn in class the following semester that a projective transformation explained all the odd deformations I was getting.

I arrived at Carnegie Mellon fresh off stints in private equity and startups. Probably as a consequence of this, I was interested in working on something I could only ever

do in academia. Boy, did I find it. When I joined Jeff's lab and the fusion project, we did not know much about fusion. Over the following years, we slowly put together that picture. I learned from Jeff to drive a research agenda grounded in a real-world application, to reduce things to their essential components and solve the simplest version of a problem, and to keep a portfolio of projects moving incrementally forward. Jeff was flexible with me when I needed money for computers, willing to let me travel around the world presenting research, and always willing to jump into a problem and attack it from first principles. I will always cherish the hours spent going over RL algorithms, experimental designs, and strategies for presenting our work in his office. Jeff has been willing to entertain weird ideas, bad ideas, and the occasional good one. As the world continues to grapple with the progress in natural language understanding of recent years, Jeff prompted me to think about how this could be useful to us in fusion. Though language was neither of our area, this prompt eventually pushed us to build a copilot for nuclear reactors and to sketch out a program for AI augmented operation of tokamaks that I would never have thought to consider without him. I spent months at a time in LA, Truckee, New York City, Austin, and San Francisco during grad school. Jeff took this in stride while insisting I still retain some connection to Pittsburgh and the in-person life at CMU. In retrospect, I think this balance worked well for me and I'm glad we landed at this compromise.

As an undergrad I took the course on graphical models given by Stefano Ermon as my last machine learning course. He was able to install in me the coherent global picture of machine learning that persists to this day through that course. As we've worked together on subsequent research I've grown to appreciate how thoroughly he knows the literature and his knack for asking the hardest questions before the reviewers do.

Dan Boyer operated as a bridge for the CMU group as someone who was deeply immersed in the plasma control world while also remaining up to date on the state of the art in machine learning. He was quick to grok the physical prior knowledge I was hoping to use for the NDS paper and invaluable in giving me the perfect physical model to use for that work; I would have struggled mightily to find one without him. On our more recent work optimizing rampdowns, his early involvement was foundational in getting that project off on the right path.

I would also like to extend my gratitude to Deepak Pathak and David Held, members of my thesis committee. Their willingness to lend their expertise and provide oversight throughout this process has been invaluable. Their perspectives and insights during the committee meetings contributed to the shaping of this research. I am grateful for their time and commitment in reviewing and providing feedback on my work.

This thesis attempts to apply and extend machine learning techniques so that they can be applied to the problems of plasma control for nuclear fusion. For this problem setting to make sense at all, we needed a prime mover from the physics community. For this I must thank Egemen Kolemen. He is fearless, hard-charging, and funny. He doesn't pull punches, but he also believes that his students can accomplish huge things. He served as our host at Princeton on multiple occasions as well as our guide in San Diego at DIII-Dand otherwise. The wealth of experience he has on fusion experimental science is invaluable, and his leadership of our efforts has made much of this possible.

I benefited during my PhD from several other mentors who worked with me and

from whom I learned a tremendous amount. I took Andrej Risteski's class on deep learning my first semester and immediately appreciated his ontology of the field and the clarity of his presentation. I knew that I wanted to do theoretical research during grad school and that it wasn't part of our main research thrust. Andrej was willing to do a terrific amount of work to get our two projects over the line. He helped me to find math papers to read, proposed approaches, proved theorems himself, checked my proofs, interrogated results I presented (even late at night), wrote papers, gave advice and was in general a sincere and involved advisor as well as a fantastic research collaborator even though there was barely a formal relationship between us. I am sincerely grateful for his help and mentorship over the years. I'd also like to acknowledge and thank Fred Koehler and Chenghui Zhou for their help and collegiality in working on these projects.

Since he was a postdoc at CMU, Willie Neiswanger has been a friend and mentor over nearly all of my research. Several of my papers extend ideas of his to the control setting, and I've benefited from his advice on essentially every project I've attempted in grad school. Since I was young I've struggled with a tendency to produce work that aesthetically leaves a bit to be desired and failed to hold myself to a high standard of presentation. At this point the voice in my head that tells me to redo all the figures because they are slightly misaligned may as well be named Willie. He was that influential in my progress towards producing high-quality research outputs (the reader can decide for themselves if this thesis qualifies). It has been a joy working with him and I am deeply grateful for all his help.

Last year, Willie introduced me to a friend, Ilija Bogunovic, who was working on RL settings that were quite similar to those I was interested in. We immediately got along, and quickly collaborated on a paper. Since then, we've worked on several subsequent projects and shared drinks on several continents. I appreciate his mentorship, leadership of projects, and penetrating principled insights into decision making algorithms. I equally appreciate his warmth and good spirits every time we meet no matter where in the world it happens. I also thank his student, Xiang Li, and our collaborator Johannes Kirschner for their help with our first project together.

Besides academic mentorship, I benefited from the support of several CMU staff members who made my research possible. I must thank the longtime Auton cluster administrator, Predrag Punosevac. Besides keeping the computing substrate up and running, I always appreciated conversations about technologies in computing, programming languages, and hearing about your daughters. Similarly, I appreciate the support of Piotr Bartosiewicz who has succeeded him. From the very first day, Suzanne Lyons Muth has been the person who had all the administrative answers. I didn't have too many questions but when I did I needed answers. Suzanne had them. For that sincerely thank her. Finally, I spent a lot of Jeff's money during my travels and my training runs. Stephanie Matvey and Ashlyn Lacovara helped me out the dozens of times I needed reimbursed for things and I thank them for this.

At several points, I ran experiments at the DIII-D National Fusion Facility in San Diego. As I'll discuss later, this is an extremely complicated machine and it is not easy to effectively operate. Nevertheless, I was invited on several occasions to be a session leader and conduct experiments on the device due to the generosity of DIII-D management in specially allowing PhD students time to conduct research. I thank the folks responsible for allocating me this time and for managing the schedule, including

but not limited to Max Austin and Auna Moser for this. I thank them again for not being mad when the vessel was breached during one notable experiment in March 2023 and for simply adjusting the operations envelope to rule out similar events happening again. There was a large team of people responsible for the operations of the tokamak, so I can't list them all. However, I'd specifically like to acknowledge Jayson Barr for his guidance and mentorship on my projects at DIII-D. He taught me physics, was always willing to share time to make sure I understood *why* something was, and was invaluable in helping clear obstacles in every project I attempted. I'd also like to thank Al Hyatt, Nik Logan, Tyler Cote, James Yang, and many others on the operations teams at DIII-Dfor their help in managing the systems of the device during run days. Finally, I'd like to thank each of the session leaders who allowed us to piggyback rampdown experiments during their primary experiments over the course of 2022.

I'd like to acknowledge the students at Princeton and in the fusion community who were largely responsible for helping me learn what practically happens in fusion, choose interesting problems to work on, and help get experiments proposed and set up. I thank Rory Conlin, Andy Rothstein, Allen Wang, and Oak Nelson for their help during this time. I will always be grateful to Joe Abbate for everything he's done for us at CMU over the years. For questions ranging from "what was the pointname for minor radius again?" to "what's really stopping us from getting fusion?" Joe was graceful and generous with his time. We owe him a huge amount for building the infrastructural code that allows us to do data-driven research using DIII-D shots. We owe him equally for his patience in reading over our proposals and papers. Thanks, Joe, for being my fusion Sherpa.

Most of my PhD was spent as part of a group of three students attempting to connect the worlds of ML research and fusion. I worked with Ian Char and Youngseog Chung on probably dozens of things ranging from presentations and proposals to utility codebases, C and Fortran code, and eventually some of the work that appears in this thesis. They've been my friends and confidants through this process but also varyingly my advisors, instructors, assistants, QA folks and many other roles. Ian and I notably worked on a project, BATS, not included in this thesis that drove us to the brink of sanity one spring. I'm glad I was on that brink with him.

I worked with fantastic and hardworking master's students: Vikramjeet Das and Rohan Shah. They worked hard on projects that sometimes didn't pan out and it was extremely gratifying to see them grow as researchers.

To my officemates and labmates and occasional collaborators, Adam Villaflor, Brian Yang, Yeeho Song, Conor Igoe, Biswajit Paria, Ben Freed, Swapnil Pande, Tejus Gupta, and others, thank you for the intellectual sparring and the fun and the visits to trivia night and the references to interesting work.

Though grad school has been a lot of work, I've benefited from an amazing collection of friends who've supported me and cheered me on along the way. In the CMU community I'd especially like to thank Theophile Gervet, Minji Yoon, and Ojash Neopane (also a collaborator) for their friendship and support over all the years. Outside the CMU community there are countless many friends who've been there for me. From my time in Truckee (my favorite place I've ever lived) and otherwise I'd like to thank Lewin Cary, Nick Bien, Conrad Sayer, and Nick Powell. From Austin: Lincoln Valdez, Connor Gunn, Alma Florez-Perez, Jay Tyler, Keyur Mehta, Breck Spencer, and Jacob

Zodikoff. From Pittsburgh, Tristan Cunha and Domi Vamossy. From SF, Martín Hernandez, Tyler Cloyd, Jack Pigott, Adam Mosharrafa, Angela Nguyen, and Upal Saha. From New York, Connor and Maria Roberts, Aaron Polhamus, Blair Silverberg, Noah Steinberg, and Jared Madfes. From Stanford, Tyler Dougan, Gabriel Bianconi, Alec Villagomez, Sarah Radzihovsky, Meena Chetty, Wes Dixon, Sam Schwager, and so many others. I'd like to thank my girlfriend, Sam Whitney. Her support and fascination with my projects has been a tremendous bulwark when the going was tough. And the delight she takes in the small things adds even more joy to the bright spots.

Finally, I'd like to thank my family: Ravij, Sarita, Mom, and Dad. Since the earliest days our house has been one where academics have been prioritized but balance maintained. My life sparkles thanks to the joy and good humor my siblings bring to it. Since an early age my parents took care to enrich our lives and expose us to as much information as possible. They sacrificed their personal lives to make sure that my siblings and I were set up as well as possible to succeed in our lives. Thanks to them, I've been free to work on precisely what I wanted and felt supported doing so. This has been a long journey that's taken consistent hard work. I'm only capable of that thanks to my family.

Contents

List of Figures

List of Tables

1 | Introduction

Since AlexNet [Krizhevsky et al., 2012] won the ImageNet competition [Russakovsky et al., 2014] in 2012, machine learning has been in a state of near-constant acceleration. The numbers of papers, dollars, startups, students, FLOPs, parameters, and datapoints have all been growing in a rapid and accelerating manner. At the start of this wave, progress was driven by a belief "that general methods that leverage computation are ultimately the most effective" [Sutton, 2019] and is now driven by a sharper understanding that there are strong empirical relationships between scale and performance in model training [Kaplan et al., 2020, Hoffmann et al., 2022]. We have seen the fruits of these efforts in computer vision [He et al., 2016], natural language processing [Brown et al., 2020a, OpenAI, 2023], and image generation [Esser et al., 2021, Ramesh et al., 2022], where progress could scarcely have been imaginable even a decade ago. Though the community of researchers and practitioners is rightly energized by the benefits of scale, we inevitably desire to apply machine learning in situations where collecting large amounts of data is infeasible. These situations arise in a variety of contexts: data can be expensive to produce, arrive one-at-a-time, or be privacy-sensitive, requiring constraints on data collection and storage. In these cases, one ought to be more thoughtful about the methods used to collect data, train models, and make decisions downstream of those models.

This is further complicated in problems that involve decision making or control. Here, data would ideally be collected online by a policy determined by the learning agent so that any errors in the agent's understanding of the problem could be corrected by experience. In reinforcement learning (RL), we have seen greatest success in applications where samples are cheap, such as games like Go or Atari [Silver et al., 2017, Mnih et al., 2015] or the Mujoco suite of tasks [Todorov et al., 2012]. For these as well as real-world applications of RL such as ad targeting [Zhao et al., 2018] where large numbers of samples are available, bandit (the single-step special case) and RL algorithms are able to effectively improve decision making at scale. There have been few applications of RL in domains where samples are expensive. Recently, reinforcement learning from human feedback (RLHF) [Christiano et al., 2017] has been applied at substantial labeling expense to align large language models (LLMs) to human preferences.

Reinforcement learning deals with fully- or partially-observable Markov decision processes (MDPs), which are sequential decision making problems where the aim is to find a policy which maximizes long-term rewards attained on a system with stationary dynamics. There are a wide variety of real-world problems where strong learned control policies could be more effective than existing algorithms or human processes. In robotics, RL has been demonstrated in the real world at tremendous expense. The authors of Levine et al. [2018] were able to successfully learn a visual grasping policy in the real world using over 800,000 grasp attempts using $\sim$10 robots at a time over the course of two months of training. This cost is improving over time as pretraining using foundation models [Bommasani et al., 2021] allows policies to start from a base of general knowledge prior to the decision making problem at hand. Notwithstanding this, for many potential

applications of robotics, the high cost of data collection precludes the use of RL. Another application of note for reinforcement learning is in autonomous vehicles, where RL could in principle be used to train a driving policy. Here as with other applications, the cost and danger of allowing many samples to be collected by an untrained policy precludes the use of RL to solve the core task.

There are also broader potential applications of RL in the control of complex systems beyond those that are commonly seen as robotics. Potential military applications of reinforcement learning include automated sensor control, resource allocation, and real-time mission planning [Yang et al., 2019, Barde et al., 2019]. This will be increasingly important as war continues to incorporate more technology and produce more data in real time [Brose, 2020]. The model-predictive control of a wide variety of industrial systems has been commonplace for decades [Camacho and Alba, 2013], and the use of learning has seen promising initial applications in cooling data centers [Lazic et al., 2018]. Future developments in these areas could help by improving efficiency, incorporating a wider range of objectives, or reducing safety risk.

RL is also beginning to be used in areas where feedback comes from humans in varying guises of cost or personal relationship. The financial markets are an obvious example of this type of feedback. Problems like automated market making, efficient arbitrage, or even the electricity futures markets can be formulated as fully or partially observable MDPs where other participants transact or simply use electricity[Henry and Ernst, 2021]. Perhaps the highest-profile application of this type is the use of reinforcement learning to fine-tune large language models in order to accomplish a downstream task [Ouyang et al., 2022, Christiano et al., 2017] like summarization or question answering. RL can even be used to optimize educational strategies for optimal (human) learning [Bassen et al., 2020]. As with many of the tasks involving physical systems, applications involving feedback from humans also suffer from the large online sample collection requirements of reinforcement learning.

The work presented in this thesis has been broadly motivated by the specific goal of applying learning-based decision making to plasma control for nuclear fusion. Fusion power has for decades been a grand challenge of science and engineering [Barbarino, 2020]. Though humanity has been attempting to build fusion reactors since the 1950s, we have yet to build one that produces more energy than it consumes for longer than a few nanoseconds [Clery, 2022]. Using a similar mechanism to the one that powers the sun, a working nuclear fusion reactor would be a highly desirable power source with an essentially unlimited fuel supply in the ocean and our lithium reserves. In fact, "the top two inches of Lake Erie contain 1.6 times more energy [for fusion] than all the world's oil supplies" [Broad, 1991]. Put in concrete terms, fossil fuel-based energy generation relies on combustion while fusion exploits the $\sim 10^7$ times more energy available per gram of fuel when fused.

In order to achieve power production through fusion, deuterium and tritium need to be heated to temperatures exceeding 100 million degrees Celsius, which is 10 times hotter than the core of the sun. As no known materials can insulate effectively against these temperatures, the plasma is suspended in a donut-shaped magnetic field produced by a device called a tokamak. This is an exciting time in the study of fusion power: the International Thermonuclear Experimental Reactor (ITER) has been under construction in France since the early 2000s and is nearing completion [Rebut, 1995] and several startups have raised billions in venture capital to test alternate approaches to the problem. Besides its obvious implications for the future of human energy production, controlled fusion also serves as a fascinating application area due to several constraints imposed by the unique characteristics of the problem. These constraints appear in varying combinations in many of the potential applications of RL described above.

Naively, one would expect the equations for fluid dynamics (Navier-Stokes) and the equations

for electromagnetism (Maxwell's laws) to both apply to plasmas and in fact they do—the magnetohydrodynamic equations which govern plasma behavior are derived starting from these two sources. Numerical solutions to plasma dynamics can be extremely slow, up to hundreds of seconds of supercomputer time per ms of plasma time and approximations which make the computations tractable on even long timescales may affect the results of simulation [Fahey and Candy, 2004]. So, we must assume access only to an imperfect and computationally expensive simulator. Similarly because of the computationally expensive underlying dynamics, any physical model of composite quantities like total plasma current or stored energy will necessarily be a gross approximation to the true solution. There is a substantial amount of logged fusion data from scientific experiments over the past several years. However, most RL methods and similar require data to be collected during the learning process, as the agent must be able to receive feedback for its mistakes and successes in order to effectively learn. Throughout my PhD, I have been working with data taken from the DIII-D device in San Diego, CA and conducted several experiments on the machine. DIII-D is one of the world's most advanced tokamaks and is the leading experimental fusion research facility in the United States.

There are several high-level strategies that might be used to improve the data efficiency of automated decision making systems. These include:

- Improved data collection strategies
- Transfer learning from simulators or meta-learning from similar problems
- Offline RL from existing data
- Incorporation of prior knowledge about the world
- Learning from expert demonstrations

In this thesis, we focus on improving data collection strategies in part I and on incorporating prior knowledge in part II. However, some of our strategies apply to meta-reinforcement learning and to the alignment of large language models, which touch some of the the other strategies mentioned here. In part I, I will present several approaches which attempt to ameliorate the problem via a prospective evaluation of the value of acquiring a particular observation and then choosing to acquire those with maximal value, including an application of one particular method to the problem of aligning language models to human preferences. Then, in part II I will present work in which we leverage prior knowledge about a system in order to reduce the number of datapoints required to identify and control it. Finally, in part III, I'll discuss the application that motivated much of the work in this thesis: plasma control for nuclear fusion. Then, I'll discuss a successful series of experiments applying Bayesian Optimization (BO) techniques to the problem of ramping down the plasma current at the end of a shot. Finally, I'll discuss the opportunities and challenges I see in further applications of machine learning to the problems in fusion.

1.1 Preliminaries

In this thesis we deal with discrete- and continuous-time dynamical systems, Markov decision processes, and contextual bandits and for consistency use common notation throughout. We define a dynamical system as a subcomponent of a MDP, consisting of a state space $\mathcal{S}$, an action space $\mathcal{A}$, a stochastic dynamics function $T : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ that gives a distribution over the next state $\Delta(\mathcal{S}) = P(\mathcal{S})$ or the time derivative of the state $\Delta(\mathcal{S}) = \frac{ds}{dt}$ for some $s \in \mathcal{S}$. A Markov decision process augments a dynamical system with a start state distribution p_0, a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and either a time discounting factor γ or a horizon H. Assuming discrete time (as

we do in all but part II) and a list of policies π consisting of $\pi_t : \mathcal{S} \to P(\mathcal{A})$, the dynamics evolve by first sampling $s_0 \sim p_0$ then alternatively sampling $a_t \sim \pi_t(s_t)$ and $s_{t+1} \sim T(s_t, a_t)$.

The goal of an MDP is to choose actions via some decision making policy that maximize the (possibly discounted) integral or sum of expected rewards over time. In the finite-horizon discrete time setting as addressed in Chapter 2, the objective is

$$J_T(\pi) = \mathbb{E}_{a_t \sim \pi_t(s_t), s_{t+1} \sim T(s_t, a_t)} \left[\sum_{i=0}^{H-1} r(s_t, a_t, s_{t+1}) \right]. \tag{1.1}$$

In the infinite-horizon discounted setting, the analogous objective is

$$J(\pi) = \mathbb{E}_{a_t \sim \pi_t(s_t), s_{t+1} \sim T(s_t, a_t)} \left[\sum_{i=0}^{\infty} \gamma^i r(s_t, a_t, s_{t+1}) \right]. \tag{1.2}$$

Single-step decision making is a special case of the MDP where $H = 1$. Traditionally the general single-step decision making problem is called a *contextual multi-armed bandit*, and when we make a special assumption of a Gaussian process prior on the reward we call the problem contextual Bayesian optimization. Out of tradition we call the context space $\mathcal{X}$ though it serves the same role as the state space $\mathcal{S}$. In either case, the objective is to find a policy $\pi : \mathcal{X} \to \mathcal{A}$ that maximizes

$$J(\pi) = \mathbb{E}_{x \sim P(\mathcal{X})} \left[r(x, \pi(x)) \right] \tag{1.3}$$

for some distribution over contexts $P(\mathcal{X})$. The non-contextual variants are equivalent to a singleton set $\mathcal{X}$. This single-step setting also applies to the design of feedforward control trajectories if we set $\mathcal{A}$ to be a sequence of controls and optimize over the entire feedforward sequence as was demonstrated in Tesch et al. [2013]; we use this setup in chapter 2 and chapter 9. We also often make RKHS assumptions on the reward, dynamics, or value functions. For a description of RKHS basics, see Rasmussen and Williams [2008].

Access Models In this work we primarily address two scenarios in which the agent interacts with the environment in order to collect data that can be used to infer a good policy. The more common one, which we call *online* is where the agent is initialized at a state $s_0 \sim p_0$ and for H steps executes an action a_t and is presented with the next state $s_{t+1} \sim T(s_t, a_t)$ and reward $r(s_t, a_t, s_{t+1})$ in order to update its estimates. The other setting we cover is the *active* setting (which has variously been called RL with a generative model or transition query RL), in which an agent chooses s_t and a_t and recieves $s'_t \sim T(s_t, a_t)$ and $r_t = r(s_t, a_t, s'_t)$. Here, the difference is that the states can be disjoint and the agent gets a choice of state.

1.1.1 Model-Predictive Control

Model-Predictive control (MPC) is a core technique used in the automatic control of dynamical systems and in model-based reinforcement learning [Chua et al., 2018, Wang and Ba, 2020]. Using an estimated dynamics model $\hat{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ 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

$$\max_{a_0, \ldots, a_h \in \mathcal{A}} \mathbb{E}_{s_{t+1} \sim T(s_t, a_t)} \left[\sum_{i=0}^{h} r(s_t, a_t, s_{t+1}) \right]. \tag{1.4}$$

Planning is typically redone periodically, often every timestep, as actions are executed in the real environment. The cross-entropy method (CEM) is often used to solve this optimization problem. In these works, we often use an improved variant of CEM described in Pinneri et al. [2020]. We will refer to π_T as the stochastic policy obtained by running MPC over a dynamics function T. In section 2.4.2, we give a Bayesian formulation of MPC which accounts for uncertainty in the dynamics.

1.1.2 Value Functions and Model-Free RL

For a policy π_t, $t \in [H]$, $s \in \mathcal{S}$, and $a \in \mathcal{A}$, the value function $V_h^\pi : \mathcal{S} \to \mathbb{R}$ and the Q-function $Q_h^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ are given by:

$$V_h^\pi(s) = \mathbb{E}_{a_t \sim \pi_t(s_t), s_{t+1} \sim T(s_t, a_t)} \Big[\sum_{t=h}^{H} r(s_t, a_t) \mid s_h = s \Big] \tag{1.5}$$

$$Q_h^\pi(s, a) = \mathbb{E}_{a_t \sim \pi_t(s_t), s_{t+1} \sim T(s_t, a_t)} \Big[\sum_{t=h}^{H} r(s_t, a_t) \mid s_h = s, a_h = a \Big] \tag{1.6}$$

Here, the hth value function denotes the return a policy might accumulate from the hth action onward. We use π^* to denote the optimal policy, and we abbreviate $V_h^{\pi*}, Q_h^{\pi*}$ as V_h^*, Q_h^*, respectively. We also have $V_h^*(s) = \sup_\pi V_h^\pi(s)$ for all $s \in \mathcal{S}$ and $h \in [H]$. In cases where the state and action spaces are finite, this is efficiently solvable using dynamic programming.

Bellman Operator The Bellman optimality equation [Bellman, 1966] plays a fundamental role in defining the optimal policy in MDPs. In terms of the value function $V^*(s)$, the Bellman equation for a discounted infinite-horizon MDP is

$$V^*(s) = \max_{a \in \mathcal{A}} \mathbb{E}_{s' \sim T(s,a)} [r(s, a, s') + \gamma V^*(s')], \tag{1.7}$$

where γ is the discount factor, and the expectation is over s'.

The Bellman operator $\mathcal{T}$, corresponding to this equation, operates on a value function V and is defined as

$$(\mathcal{T}V)(s) = \max_{a \in \mathcal{A}} \mathbb{E}_{s' \sim T(s,a)} [r(s, a, s') + \gamma V(s')]. \tag{1.8}$$

Fixed points of this operator correspond to the optimal value function, i.e., $V^* = \mathcal{T}V^*$.

Policy Iteration Policy iteration [Sutton and Barto, 1998] is a classic algorithm for finding the optimal policy in infinite horizon discrete-time MDPs. The algorithm alternates between policy evaluation and policy improvement steps.

Given a policy π, policy evaluation computes the value function $V^\pi(s)$ for all states $s \in \mathcal{S}$. The value function satisfies the Bellman expectation equation:

$$V^\pi(s) = \mathbb{E}_{a \sim \pi, s' \sim T(s,a)} \left[r(s, a, s') + \gamma V^\pi(s') \right], \tag{1.9}$$

and can be computed iteratively until convergence.

Once the value function for a policy π has been computed, the policy can be improved by acting greedily with respect to V^π. Specifically, a new policy π' is generated as follows:

$$\pi'(s) = \operatorname{argmax}_{a \in \mathcal{A}} \mathbb{E}_{s' \sim T(s,a)} \left[r(s, a, s') + \gamma V^\pi(s') \right]. \tag{1.10}$$

By alternating the policy evaluation and policy improvement steps we are guaranteed to find a fixed point, which corresponds to V^*.

1.1.3 Actor-Critic Methods

Actor-Critic algorithms consist of two main components: an actor that selects actions and a critic that estimates the value function of those actions. The actor and critic are typically parameterized by θ and ϕ, respectively. These methods often use neural networks as the parameterized function approximators and therefore scale best out of the methods describe here.

Soft Actor-Critic (SAC) SAC [Haarnoja et al., 2018] aims to maximize the expected return while also encouraging sufficient exploration, through the use of a maximum entropy framework. The policy is usually stochastic, parameterized by θ, and aims to solve

$$\max_{\theta} \mathbb{E}_{a_t \sim \pi_\theta(s_t), s_{t+1} \sim T(s_t, a_t)} \left[\sum_{i=0}^{\infty} \gamma^i (r(s_i, a_i, s_{i+1}) + \alpha \mathbb{H}(\pi(\cdot|s_i))) \right], \tag{1.11}$$

where α is a temperature parameter and $\mathbb{H}$ is the entropy.

Proximal Policy Optimization (PPO) PPO [Schulman et al., 2017] is another policy optimization algorithm that aims to make the largest update possible to the policy in the direction of local improvement without diverging too much from the policy that collected the data used to determine that direction. The core idea is to apply an estimator of the policy gradient subject to a KL constraint that ensures the new policy is not too different from the old policy:

$$\max_{\theta} \mathbb{E}_{a_t \sim \pi_\theta(s_t), s_{t+1} \sim T(s_t, a_t)} \left[\frac{\pi_\theta(a|s)}{\pi_{\text{ref}}(a|s)} (Q^{\pi}_{\text{ref}}(s, a) - V^{\pi}_{\text{ref}}(s, a)) \right], \tag{1.12}$$

subject to

$$\left| \mathbb{E}_{a_t \sim \pi_\theta(s_t), s_{t+1} \sim T(s_t, a_t)} \left[\pi_\theta(a|s) / \pi_{\text{ref}}(a|s) - 1 \right] \right| \leq \epsilon, \tag{1.13}$$

where π_{ref} is the old policy and ϵ is a hyperparameter.

Part I

Sample Efficient Decision Making through Better Choice of Data

As discussed in the introduction, RL has suffered for years from a curse of poor sample complexity. State-of-the-art model-free RL 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 like plasma control where each sample is expensive, even this smaller cost can be prohibitive for the practical application of RL.

In general if the sample budget is limited and the learning mechanisms are fixed, one obvious method by which to improve downstream policy performance is to be sure to collect the most useful data in order to learn a good policy, or in other words, better exploration. There are a handful of general strategies that are used in Bayesian optimization [Frazier, 2018, Shahriari et al., 2015] for these purposes and have been well studied in that context. These include:

- Thompson sampling [Russo et al., 2018], where actions are played which optimize a posterior sample of the model.
- information-based methods such as those proposed in Hennig and Schuler [2012] and Hernández-Lobato et al. [2014], where actions are played which maximally gather information about the optimum.
- upper-confidence bound based methods [Srinivas et al., 2009], where actions are selected optimistically in order to rule out uncertain but promising areas.
- knowledge gradient [Frazier et al., 2008], which takes actions which are expected to lead to a better eventual policy.

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 and Verdinelli, 1995]. We aim to leverage these ideas for data-efficiency in reinforcement learning.

One theme of this thesis is that by generalizing some of these ideas from the single-step BO setting to the MDP setting, we can make better use of samples in reinforcement learning problems. Along the way, we also contribute to the literature on contextual Bayesian optimization. In this chapter, we present work that extends the information-based, confidence-bound-based, and knowledge-gradient-based methods to the reinforcement learning setting. Each of these is suitable to different types of problems and performs well under different assumptions. In chapter 2, we present work from Mehta et al. [2022c] and Mehta et al. [2022a], where we develop an acquisition function for model-based reinforcement learning that encourages information gain about the optimal trajectory. Next, in chapter 3, we explore the relationship between the decision-theoretic H-entropy and the Bayes-adaptive MDP. By maximizing this generalized information, we develop an exploration strategy that we extend to the metalearning of policies for continuous control problems.

Finally, in Chapters 4 and 5, we present work from Li et al. [2023] and currently under submission, where we develop a method that guarantees efficient near-optimal policy identification. Finally, we extend that method to the case of comparative feedback, and apply the resulting method to 3 datasets of human preferences in order to train and evaluate their alignment to the associated goals.

1.2 Prior Work on Sample-Efficient RL

Exploration in Reinforcement Learning The most common strategy for exploration in RL is to execute a greedy policy with some form of added stochasticity. The simplest approach, ϵ-greedy exploration as used in Mnih et al. [2013], takes the current action thought to be best with probability $1 - \epsilon$ and a random action with probability ϵ. Other methods use added Ornstein-Uhlenbeck action noise [Lillicrap et al., 2015] to the greedy policy, or entropy bonuses [Haarnoja et al., 2018] to the policy or value function objectives to add noise to a policy which is otherwise optimizing the RL objective.

Tabular RL is often solved by choosing actions based on upper confidence bounds on the value function [Chen et al., 2017b, Lee et al., 2021], but explicitly computing and optimizing these bounds in the continuous setting is substantially more challenging. Recent work [Curi et al., 2020] approximates this method by computing one-step confidence bounds on the dynamics and training a 'hallucinated' policy which chooses perturbations within these bound to maximize expected policy performance. Another recent work [Ash et al., 2022] uses anti-concentration inequalities to approximate upper confidence bounds in MDPs with discrete actions.

Thompson sampling (TS) [Russo et al., 2018], which samples a realization of the MDP from the posterior and acts optimally as if the realization was the true model, can be applied for exploration in a model-free manner as in [Osband et al., 2016a] or in a model-based manner as in Strens [2000]. As the posterior over MDP dynamics or value functions can be high-dimensional and difficult to represent, the performance of TS can be hindered by approximation errors using both Gaussian processes and ensembles of neural networks. Curi et al. [2020] recently investigated this and found that this was potentially due to an insufficiently expressive posterior over entire transition functions, implying that it may be quite difficult to solve tasks using sampled models. Similarly, the posterior over action-value functions in Osband et al. [2016a] is only roughly approximated by training a bootstrapped ensemble of neural networks.

There is also a rich literature of Bayesian methods for exploration, which are typically computationally expensive and hard to use, though they have attractive theoretical properties. These methods build upon the fundamental idea of the Bayes-adaptive MDP [Ross et al., 2007], which we detail in section 3.3. Kolter and 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. However, these methods don't scale to continuous problems and don't provide a way to choose states to query. These methods were further extended with the development of knowledge gradient policies [Ryzhov et al., 2019, Ryzhov and Powell, 2011], which approximate the value function of the Bayes-adaptive MDP, and information-directed sampling (IDS) [Russo and 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 yet solve fully continuous problems and computes the information gain with respect to the dynamics rather than some notion of the optimal policy. In a

similar spirit, Arumugam and Van Roy [2021] provide a further generalization of IDS which can also be applied to RL. One recent work very close to ours is Lindner et al. [2021], which actively queries an expensive reward function (instead of dynamics as in this work) to learn a Bayesian model of reward. Another very relevant recent paper [Ball et al., 2020] gives an acquisition strategy in policy space that iteratively trains a data-collection policy in the model that trades off exploration against exploitation using methods from active learning. Achterhold and Stueckler [2021] use techniques from BOED to efficiently calibrate a Neural Process representation of a distribution of dynamics to a particular instance, but this calibration doesn't include information about the task. A tutorial on Bayesian RL 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], a density estimate [Bellemare et al., 2016], or predictive entropy [Buisson-Fenet et al., 2020]. Another work, Plan2Explore [Sekar et al., 2020], prospectively plans to find areas of novelty where the dynamics are uncertain. However, these methods all assume that there is no reward function and are inefficient for the setting of this paper, as they spend time exploring areas of state space which can be quickly determined to be bad for maximizing reward on a task.

2 | Information theoretic approaches: Can we measure what we might learn?

Motivated by our opening questions, in this chapter we present work published in Mehta et al. [2022c] and Mehta et al. [2022a] which presents and evaluates a novel objective for information-based exploration in model-based reinforcement learning. We begin in the active 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. Although this setting has been studied in the tabular case, to the best of our knowledge it had not been studied in the continuous MDP literature.

The costly transition functions present in many applications we have discussed prompted 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. As initial work in this direction [Mehta et al., 2022c], we drew 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. 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. It also leverages the current estimates of which states the optimal policy will visit and values potential queries accordingly. Furthermore, our acquisition function is 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. We assessed the performance of our acquisition function as a data selection strategy in the active setting. Using this method, which we call *Bayesian active reinforcement learning* (BARL), 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.

After this, we extend the data selection criterion to the standard MDP setting, where we are able to use MPC with a modified cost function in order to explicitly plan for future information gain about the optimal trajectory. In order to do so, we extend the cost function to jointly account for the information gain of observing a set of random variables rather than a single observation. This method, which we denote *Trajectory Information Planning* (TIP), also is able to improve in sample complexity over a wide range of model-based and model-free RL methods on a handful of continuous control problems up to a moderate number of dimensions.

2.1 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 improvement in the performance of the policy on the MDP (conditioned on all previously observed data) when observing one additional state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$. As acquisition functions are greedy, they aren't necessarily optimal data-selection strategies given a fixed budget, compared to non-myopic strategies such as solving the Bayes-adaptive MDP. However, greedy strategies using mutual information are tractable and are often effective due to the submodularity of the expected information gain. More specifically, our acquisition function is an expected information gain (EIG) or equivalently the mutual information (MI) between a query of our transition model and a representation of the optimal policy, as we elaborate in this section.

A typical approach in a Bayesian setting might be to gather data such that the entropy $\mathbb{H}[\pi^*]$ of the belief of the optimal policy π^* is minimized. However, a full specification of π^* includes the behavior of the policy in all parts of the state space including states that are not visited at all, or visited less often in rollouts of π^* when the start state is sampled from p_0. As a result, not all points in the state space are equally important when learning an optimal policy aimed at maximizing the expected reward. Optimizing the entropy of π^* would lead to a uniform treatment of all the points in the state space, and hence would be far from optimal for the standard goal in RL.

We instead propose to minimize the entropy of the *optimal trajectory* $\tau^* = \{s_i\}_{i=1}^H$ defined as 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. The optimal trajectory is completely specified by π^* and the randomness arising from the MDP. Furthermore, τ^* contains the necessary information needed about the transition function T to solve the MDP, since any state that could ever be visited by π^* is in the support of τ^*. We empirically observe that this leads to an efficient strategy for active RL.

The randomness in τ^* arises from three sources: the start state distribution p_0, the dynamics T constituting the *aleatoric uncertainty*, and the uncertainty in our estimate of the model T due to our limited experience which constitutes the *epistemic uncertainty*. The first two sources of uncertainty being aleatoric in nature cannot be reduced by experience. Our proposed acquisition function based on information gain naturally leads to reduction in the epistemic uncertainty about τ^* as desired. Finally, our acquisition function for a given state-action pair (s, a) is given as

$$\begin{aligned} \mathrm{EIG}_{\tau^*}(s, a) &= \mathbb{E}_{s' \sim T(s,a|D)} \Big[\mathbb{H}[\tau^* \mid D] - \mathbb{H}[\tau^* \mid D \cup \{(s, a, s')\}] \Big] \\ &= \mathbb{E}_{s_0 \sim p_0} \Big[\mathbb{E}_{s' \sim T(s,a|D)} \Big[\mathbb{H}[\tau^* \mid D, s_0] - \mathbb{H}\big[\tau^* \mid D \cup \{(s, a, s')\}, s_0\big] \Big] \Big] . \end{aligned} \tag{2.1}$$

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

2.1.1 Estimating EIG_{τ^*} via Posterior Function Sampling

For EIG_{τ^*} to be of practical benefit, we must be able to tractably approximate it. Here we show how to obtain such an approximation. By the symmetry of MI, we can rewrite Equation (2.1) as

$$\mathrm{EIG}_{\tau^*}(s, a) = \mathbb{E}_{s_0 \sim p_0} \left[\mathbb{E}_{\tau^* \sim P(\tau^*|D)} \left[\mathbb{H}[s' \mid s, a, D, s_0] - \mathbb{H}[s' \mid s, a, \tau^*, D, s_0] \right] \right] . \tag{2.2}$$

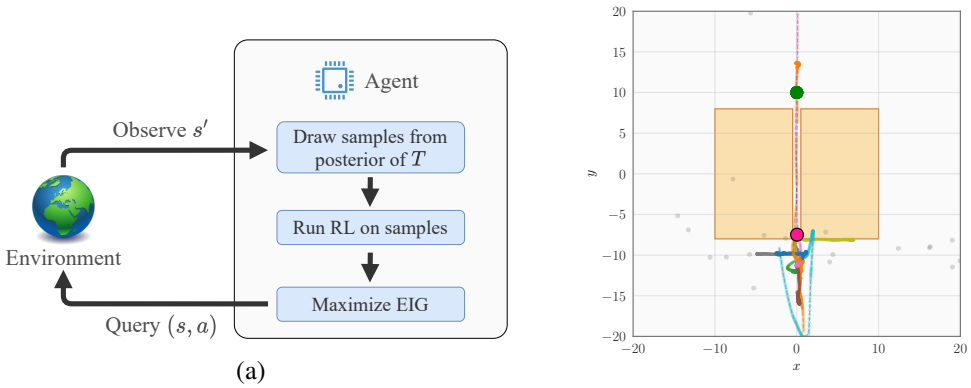

Figure 2.1: (a) A diagram of the BARL data-collection loop. (b) An illustration of the EIG_{τ^*} computation over several sample paths τ_i^* (multi-colors) sampled from $P(\tau^* \mid D)$ for a dataset of past queries (grey points). The optimizer (in pink) is a point that is maximally informative when learning a model for crossing the path between the lava pools (orange rectangles) to the goal (green).

Since $\mathbb{H}[s' \mid s, a, D, s_0] = \mathbb{H}[s' \mid s, a, D]$ doesn't depend on τ^* 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 the transition function $P(T \mid D)$. In order to compute the other term, we must take samples $\tau_{ij}^* \sim P(\tau^* \mid D)$. 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. We then 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 τ_{ij}^*. This is an expression of the generative process for τ^* as described in the previous section that accounts for the uncertainty in T. Formally, we can approximate EIG_{τ^*} via Monte-Carlo as

$$\text{EIG}_{\tau^*}(s, a) \approx \mathbb{H}[s' \mid s, a, D] - \frac{1}{mn} \sum_{i \in [m]} \sum_{j \in [n]} \mathbb{H}[s' | s, a, \tau_{ij}^*, D]. \tag{2.3}$$

Finally, we must calculate the entropy $\mathbb{H}[s'|s, a, \tau_i^*, D]$. For this, we follow a similar strategy as Neiswanger et al. [2021]. In particular, since τ_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 here, we modify the computation of the acquisition function to factor p_0 as an irreducible source of uncertainty. We also extend the function being queried to be vector-valued.

2.1.2 Bayesian Active Reinforcement Learning

In this work, we take a simple approach for nonlinear control in continuous spaces and assume a Gaussian process (GP) prior $P(T)$ to model the dynamics. Though computationally expensive, this choice ensures that we can easily approximate all necessary quantities. However, we note that the development of the acquisition function is general and any Bayesian model could be used in principle.

The transition function $T : \mathcal{S} \times \mathcal{A} \to p(\mathcal{S})$ (dynamics) can be modeled with a GP due to its non-parametric nature and ability to capture uncertainties in T. 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 as independent GPs. More specifically, we model the change in state $\Delta(s, a) = T(s, a) - s$ rather than the final state $T(s, a)$ directly. This is helpful for continuous control problems since the state often changes by only a small 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], based on sparse-GPs and random Fourier approximations of kernels [Rahimi and Recht, 2007], allowing us to approximately but efficiently sample from the GP posterior conditioned on the observations.

Assuming access to a generative model and an initial dataset D (for which, in practice, we use one randomly sampled datapoint (s, a, s')), we compute EIG_{τ^*} for D by running MPC on posterior function samples and approximate $\mathrm{argmax}_{s\in\mathcal{S},a\in\mathcal{A}} \mathrm{EIG}_{\tau^*}(s, a)$ 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 (1.4) 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 2.1a. We discuss details of training hyperparameters and the GP model in Appendix A.2.

Algorithm 1 Bayesian active reinforcement learning (BARL) using EIG_{τ^*}

Inputs: transition function query budget b, number of points for optimization k, number of posterior function samples n.
Initialize $(x_0, y_0) \sim U(\mathcal{S} \times \mathcal{A})$, $x'_0 \sim T(x_0, y_0)$, and $D \leftarrow \{(x_0, y_0, x'_0)\}$.
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 $\{\tau^*{}_\ell\}_{\ell=1}^n$ by executing MPC policy $\pi_{T'_\ell}$ on the dynamics of T'_ℓ starting from s_0.
  $(x_1, y_1), \ldots, (x_k, y_k) \sim U(\mathcal{S} \times \mathcal{A})$.
  $(x^*, y^*) \leftarrow \mathrm{argmax}_{\{(x_i,y_i)\}_{i=1}^k} \mathrm{EIG}_{\tau^*}(x_i, y_i)$.
  $D \leftarrow D \cup \{(x^*, y^*, x'^*)\}$ where $x'^* \sim T(x^*, y^*)$.
end for
return $\pi_{\hat{T}}$ where $\hat{T}(s, a) = \mathbb{E}_{T\sim P(T|D)}[T(s, a)]$

2.2 Experiments for BARL

The aim of our study of acquisition functions for RL is to reduce the sample complexity of learning good policies in continuous spaces, under expensive dynamics. Here, we demonstrate the effectiveness of using EIG_{τ^*} to leverage transition queries by comparing against 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 proposed method, BARL, greatly outperforms other methods across the board. In particular, BARL uses $5 - 1,000\times$ less data to solve problems than state-of-the-art model-based RL algorithms and $10^3 - 10^5\times$ less data than model-free RL algorithms. In this section we primarily

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	5200	6000	57000	13000
Cartpole	**91**	161	121	1625	31000	18000	N/A
Beta Tracking	96	**36**	N/A	300	9000	6000	16000
Reacher	**251**	751	N/A	700	23000	13000	N/A

Table 2.1: **Sample Complexity:** Median number of samples across 5 seeds required to reach 'solved' performance, averaged across 5 trials. We determine 'solved' performance by running an MPC policy (similar to the one used for evaluation) on the ground truth dynamics to predict actions. We record 'N/A' when the median run is unable to solve the problem by the end of training.

focus on the performance of the controller, and in section A.2.3 we also discuss the runtime of the algorithm. We discuss comparison methods in section A.2.1 and the control problems we evaluated them on in section A.2.2.

We see in both the sample complexity figures in Table 2.1 and the learning curves in Figure 2.2 that BARL leverages EIG_{τ^*} to significantly 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 reasonably well across the board and is competitive with BARL on the plasma problem but requires relatively more samples in 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 2.1b gives some intuition as to why: points are initially 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.

2.3 Does BARL choose 'meaningful' datapoints?

We further support our assertion that BARL is picking 'meaningful' points to the control problem by the evidence in Figure 2.3. Here, BARL is able to solve the reacher problem while EIG_T is not. However, BARL has much worse model predictions on random data than EIG_T while doing a much better job modeling data used by the MPC procedure. Clearly, the EIG_{τ^*} 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 A.3.1, we study whether the acquisition function we see here is able to work with a

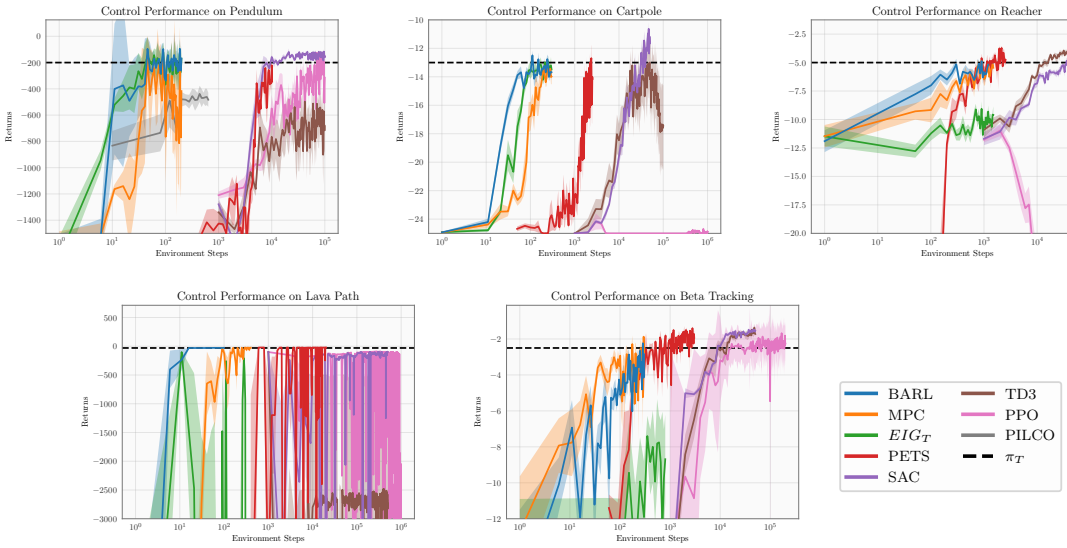

Figure 2.2: Learning Curves of RL methods, showing control performance averaged across 5 seeds. In each, the x-axis is on a logarithmic scale to account for widely varying data requirements. We see that though most algorithms end up reaching roughly the same performance on each task, BARL is substantially more efficient in most cases. The shaded region is the standard error of the average performance across the 5 seeds. We additionally include a plot of the performance of the PILCO algorithm [Deisenroth and Rasmussen, 2011] on Pendulum. PILCO makes assumptions about the initial state distribution and suffers from numerical instability under long control horizon so we were unable to reach representative performance on the other problems.

suboptimal controller on posterior samples of the dynamics. Our experiments show that EIG_{τ^*} seems to work well even when the policy used to generate τ^* is suboptimal.

2.4 Trajectory Information Planning

In this rest of this chapter, we extend the BARL method to the standard RL setting, where an agent takes sequential actions in the environment up to some max trajectory length in order to collect data that can be used to learn a policy. Our method consists of a generic framework for Bayesian (or approximately Bayesian) model-predictive control and a novel cost function for planning that allows us to explicitly plan to find the maximal amount of new information relevant to our task. In Section 2.4.2, we describe the MPC framework and highlight that many prior methods approximate this framework while using a greedy cost function that corresponds to the future negative expected rewards or a pure exploration cost function that corresponds to future information about the dynamics. Afterwards, in Section 2.4.3, we derive our new cost function and describe how it is computed. The overall method we introduce simply applies this planning framework with our new cost function.

2.4.1 Preliminaries for TIP

We will denote trajectories as $\tau \sim p(\tau \mid \pi, T)$ where $\tau = [(s_0, a_0), \ldots, (s_{H-1}, a_{H-1}), s_H]$ generated by $s_0 \sim p_0$, $a_i \sim \pi(s_i)$, and $s_{i+1} \sim T(s_i, a_i)$. We can write the return of a trajectory as $R(\tau) = \sum_{i=0}^{H-1} r(s_i, a_i, s_{i+1})$ for the states and actions s_i, a_i that make up τ. The MDP objective

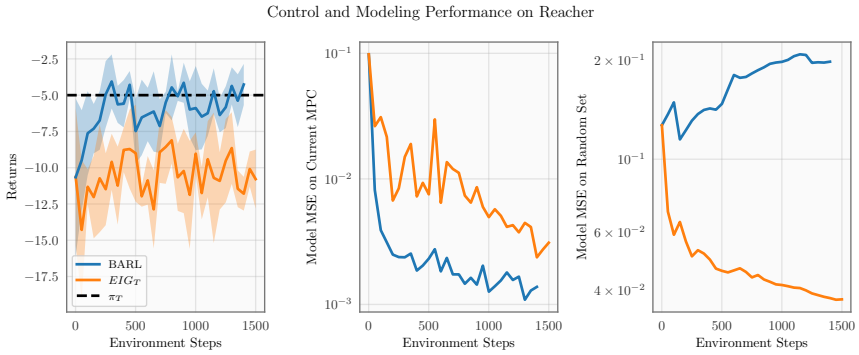

Figure 2.3: For a single run of BARL and of EIG_T using the same prior model, we evaluate control performance, as well as modeling error, on both the predictions used by the MPC procedure and on a set of points uniformly sampled from the state-action space.

can then be written as

$$J_T(\pi) = \mathbb{E}_{\tau \sim p(\tau|\pi,T)} \left[R(\tau) \right]. \tag{2.4}$$

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 use τ^* to denote optimal trajectories, i.e. $\tau^* \sim p(\tau \mid \pi^*, T)$.

For the open-loop setting, we assume a fixed start state s_0 and aim to find an action sequence $a_0, \ldots, a_{H-1}$ that maximizes the sum of rewards in an episode. We will slightly abuse notation and write $\tau \sim p(\tau \mid a_{0:H-1})$ and $J_T(a_{0:H-1})$ with these actions fixed in place of a reactive policy, and again use τ^* to refer to the trajectories generated by an optimal action sequence.

We assume in this work that applying planning algorithms like [Pinneri et al., 2020] to a dynamics function T will result in a trajectory that approximates τ^*.

2.4.2 Model-Predictive Control in Bayesian Model-Based RL

In this section, we give a formulation of Bayesian planning for control that generalizes ideas from methods such as PILCO [Deisenroth and Rasmussen, 2011] and PETS [Chua et al., 2018]. This formulation highlights these methods' inherently greedy nature and hints at a possible solution. The objective of Bayesian planning is to find the h-step action sequence that maximizes the expected future returns under model uncertainty. That is,

$$\underset{a_0,\ldots,a_{h-1}}{\operatorname{argmin}} \; \mathbb{E}_{T' \sim P(T|D), \tau_e \sim P(\tau|s_0=s, a_{0:h-1}, T')} \left[C(\tau_e) \right] \tag{2.5}$$

for some cost function C over trajectories and some start state s. If operating in the open-loop control setting, the agent executes the sequence of actions found without replanning. This procedure can also be extended to closed-loop control via model-predictive control (MPC), which involves re-planning (2.5) at every state the agent visits and playing the first action from the optimal sequence. Concretely, the MPC policy for our Bayesian setting is as follows:

$$\pi_{\text{MPC}}(s) = \underset{a_0}{\operatorname{argmin}} \min_{a_1,\ldots,a_{h-1}} \mathbb{E}_{T' \sim P(T|D), \tau_e \sim P(\tau|s_0=s, a_{0:h-1}, T')} \left[C(\tau_e) \right] \tag{2.6}$$

Whether we do open-loop control or closed-loop control via MPC, the cost function C, is integral to how the agent will behave. Prior work has predominantly focused on two types of cost

function:

$$\underbrace{C_g(\tau) = -R(\tau)}_{\text{Greedy Exploration}} \qquad \underbrace{C_e(\tau) = -\sum_{i=0}^{h} \mathbb{H}\left[T(s_i, a_i) \mid D\right]}_{\text{Task-Agnostic Exploration}} \tag{2.7}$$

Previous works such as Kamthe and Deisenroth [2018] and PETS [Chua et al., 2018] use the greedy exploration cost function, C_g. This cost function incentivizes trajectories that achieve high rewards over the next h transitions on average. In works that focus on task-agnostic exploration such as Sekar et al. [2020] and Shyam et al. [2019], the cost function C_e (or similar) is used to encourage the agent to find areas of the state space in which the model is maximally uncertain. Note that we use π_g to refer to the greedy policy given by using (2.6) with C_g.

The optimization problem in (2.6) is typically approximately solved in one of three ways: Deisenroth and Rasmussen [2011] and Curi et al. [2020] directly backpropagate through the estimated dynamics and reward functions to find policy parameters that would generate good actions, Janner et al. [2019b] use an actor-critic method trained via rollouts in the model alongside the data collected to find a policy, and Chua et al. [2018] and Mehta et al. [2022c] use the cross-entropy method [De Boer et al., 2005] to find action sequences which directly maximize the reward over the estimated dynamics. In this work, we use a version of the last method given in Pinneri et al. [2020], denoted iCEM, to directly find action sequences that optimize the cost function being used. We approximate the expectation by playing the actions on multiple samples from the posterior $P(T \mid D)$. Algorithm 2 gives a formal description of the method and Section A.5.5 provides further details.

Algorithm 2 Bayesian Model-Predictive Control with Cost Function C

Inputs: transition function episode query budget b, number of posterior function samples k, planning horizon h.
Initialize $D \leftarrow \emptyset$.
for $i \in [1, \ldots, b]$ **do**
  Sample start state $s_0 \sim p_0$.
  for $t \in [0, \ldots, H-1]$ **do**
    Sample posterior functions $\{T'_\ell\}_{\ell=1}^{k} \sim P(T' \mid D)$.
    Approximately find $\operatorname{argmin}_{a_0,\ldots,a_{h-1}} \sum_{\ell=1}^{k} \mathbb{E}_{\tau_\ell \sim p(\tau \mid T'_\ell, a_0, \ldots, a_{h-1})}\left[C(\tau_\ell)\right]$ via iCEM.
    Execute action a_0 by sampling $s_{t+1} \sim T(s_t, a_0)$.
    Update dataset $D \leftarrow D \cup \{(s_t, a_0, s_{t+1}\}$.
  end for
end for
return π_g for the posterior $P(T' \mid D)$.

2.4.3 A Task-Specific Cost Function based on Trajectory Information

In this work, we aim to explore by choosing actions that maximize the conditional expected information gain (EIG) about the optimal trajectory τ^*. This is the same overall goal as that of Mehta et al. [2022c], where the EIG_{τ^*} acquisition function was introduced for this purpose. However, in this paper we generalize this acquisition function in order to allow for sequential information collection, and account for the redundant information that could be collected between

timesteps. As discussed at length in Osband et al. [2019], it is essential to reason about how an action taken at the current timestep will affect the possibility of learning something useful in future timesteps. In other words, exploration must be *deep* and not greedy. Explicit examples are given in Osband et al. [2019] where the time to find an ϵ-optimal policy in a tabular MDP is exponential in the state size unless exploration can be coordinated over large numbers of timesteps rather than being conducted independently at each action. As the EIG_{τ^*} acquisition function is only defined over a single state-action pair and mutual information is submodular, we cannot naively use the acquisition function as is (or sum it over many datapoints) to choose actions that lead to good long-term exploration. This is clear in e.g. navigation tasks, where the nearby points visited over trajectories will provide redundant information about the local environment.

We therefore give a cost function that generalizes EIG_{τ^*} by taking a set of points to query and computing the *joint* expected information gain from observing the set. Our cost function is non-Markovian in the state space of the MDP, but it is Markovian in the dataset, which represents a point in the belief space of the agent about the dynamics. Let $\mathcal{X} = \{x : x \subseteq \mathcal{S} \times \mathcal{A}, |x| < \infty\}$ be the set of finite subsets of the set of all state-action pairs. Our cost function $C_{\tau^*} : \mathcal{X} \to \mathbb{R}$ is defined below to be the negative *joint expected information gain* about the optimal trajectory τ^* for a subset $X \in \mathcal{X}$. In particular, assuming an existing dataset D, a set of h query points $X = \{(s_i, a_i)\}_{i \in [h]}$, and a random set of next states $S' = \{s'_i \sim T(s_i, a_i), i \in [h]\}$,

$$C_{\tau^*}(X) = \mathbb{E}_{S' \sim p(S'|X,D)} \left[\mathbb{H} \left[\tau^* \mid D \cup S' \right] \right] - \mathbb{H} \left[\tau^* \mid D \right]. \tag{2.8}$$

This formulation of C_{τ^*} forces our method to handle the redundant information among queries—it is likely that $I(s'_1, \tau^*) + I(s'_2, \tau^*) > I(\{s'_1, s'_2\}, \tau^*)$ and our method should avoid this overestimation. However, as written, this function relies on computing entropies on high-dimensional trajectories where the form of the joint distribution of the elements is unknown. To tractably estimate this quantity, we use the fact that $C_{\tau^*}(X) = -I(S', \tau^*) = -I(\tau^*, S')$ for the mutual information I. This allows us to exchange τ^* and our set of queries so that τ^* is giving information about the posterior predictive distribution of our set. In other words,

$$C_{\tau^*}(X) = \mathbb{E}_{\tau^* \sim p(\tau^*|D)} \left[\mathbb{H} \left[S' \mid D \cup \tau^* \right] \right] - \mathbb{H} \left[S' \mid D \right]. \tag{2.9}$$

In order to compute the right-hand term, we must take samples $\tau^*_{ij} \sim P(\tau^* \mid D), i = 1, \ldots, m, j = 1, \ldots, n$. 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. We then run a planning procedure using iCEM [Pinneri et al., 2020] on each of the posterior functions from $s_0^{(i)}$ using T'_{ij} for T (using our assumption that planning can generate approximately optimal trajectories given ground-truth dynamics), giving our sampled τ^*_{ij}. Formally, we can approximate C_{τ^*} via Monte-Carlo as

$$C_{\tau^*}(X) \approx \frac{1}{mn} \left(\sum_{i \in [m]} \sum_{j \in [n]} \mathbb{H}[S' | D \cup \tau^*_{ij}] \right) - \mathbb{H}[S' \mid D]. \tag{2.10}$$

Assuming the dynamics are modelled with a Gaussian process, we can compute the joint Gaussian probability of the next states S' [Rasmussen and Williams, 2008]. As the entropy of a multivariate Gaussian depends only on the log-determinant of the covariance, $\log |\Sigma|$, we can tractably compute the joint entropy of the model predictions $\mathbb{H}[S' \mid D]$ and optimize it with a zeroth order optimization

algorithm. Finally, we must calculate the entropy $\mathbb{H}[S'|D \cup \tau_{ij}^*]$. For this, we follow a similar strategy as Neiswanger et al. [2021]: since τ_i^* is a set of states given by the transition model, we can treat them as additional noiseless datapoints for our dynamics model and condition on them before computing the joint covariance matrix for S'. Given this newly generalized acquisition function, we can instantiate a method of planning in order to maximize future information gained. We give the concrete procedure for computing our acquisition function in Algorithm 3, noting that trajectories τ_{ij}^* do not depend on the query set X and can be cached for various values of X as long as the dataset D does not change.

Our ultimate procedure, which we name *Trajectory Information Planning* (TIP), is quite simple: run model-based RL using MPC as in Algorithm 2, but set the cost function to be $C_{\tau^*}(\tau)$ instead of C_g or C_e, and compute this cost function using Algorithm 3. At test time, we return to planning with C_g as the cost function and greedily attempt to maximize returns rather than performing exploration. We can also formulate an open-loop variant of our method, oTIP, which involves planning once and then executing the entire action sequence.

Algorithm 3 Computation of C_{τ^*}

Inputs: dataset $D = \{(s_k, a_k, s'_k)\}$, query set X, number of start state samples m, number of posterior function samples n.
Sample m start states $\{s_0^{(i)}\}_{i=1}^m \sim p_0$.
for $i \in [m]$ **do**
  Sample n posterior functions $\{T'_j\}_{j=1}^n \sim P(T' \mid D)$.
  for $j \in [n]$ **do**
    Set $\pi_j^* \leftarrow \pi_{\text{MPC}}$ using C_g and a singleton posterior $P(T \mid D) = \delta(T'_j)$ as in (2.6).
    Compute τ_{ij}^* by executing π_j^* on T'_j starting from $s_0^{(i)}$.
  end for
end for
Compute the joint posterior covariance $\Sigma^{S'} \mid D$ across all points in X.
Compute the joint posterior covariances $\Sigma_{ij}^{S'} \mid D \cup \tau_{ij}\ \forall i \in [n], j \in [m]$ across all points in X.
return $\log|\Sigma^{S'}| - \frac{1}{nm}\sum_{i\in[n],j\in[m]} \log|\Sigma_{ij}^{S'}|$.

2.4.4 Computational Cost and Implementation Details

Though the TIP algorithm is designed for settings where samples are expensive, it is important to understand, both theoretically and practically, the computational cost of this method. For ease of notation, we make the simplifying assumption that the planning algorithm used (in this case, iCEM from [Pinneri et al., 2020]) evaluates p action sequences consisting of h (the planning horizon) actions and that our current dataset is of size N. In order to efficiently sample functions from the posterior over dynamics functions, we use the method from Wilson et al. [2020]. This reduces the naive complexity of querying these functions from $O(N^3)$ to a one-time $O(N)$ cost and then $O(1)$ for additional queries. As we derive in Section A.5.1, the computational complexity of one TIP planning iteration is $O\left(nm\left((N+H)^3 + ph\,(N+H)^2\right)\right)$. The two asymptotically expensive operations are (1) computing the Cholesky decompositions of the nm kernel matrices for datasets $D \cup \tau_{ij}^*$ and (2) solving the triangular systems using the cached Cholesky decompositions in order to compute the covariance matrices $\Sigma_{ij}^{S'} \mid D \cup \tau_{ij}^*$ for each of the p action sequences used by the planning algorithm.

However, our implementation choices mean that in practice these operations are not the most expensive step. The covariance matrix computations, which are the theoretical bottleneck, are implemented in JAX [Bradbury et al., 2018], allowing them to be compiled to much faster machine code and vectorized across large batches of queries. In fact, the most expensive operation in practice is planning on the sampled transition functions T_i' to sample optimal trajectories τ_{ij}^*. This is due to the fact that in practice p is large and we implemented the planner in NumPy [Harris et al., 2020] so it cannot be compiled together with the Tensorflow [Abadi et al., 2015] code from Wilson et al. [2020], which is used for predicting which states will be visited for the planner. We give further information on the implementation in Section A.5.

2.5 Experiments for TIP

The aim of our development of the TIP algorithm and the C_{τ^*} acquisition function for RL is to reduce the sample complexity of learning effective policies in continuous MDPs given limited access to expensive dynamics. In this section we demonstrate the effectiveness of TIP in quickly learning a good policy by comparing against a variety of state-of-the-art reinforcement learning algorithms and strong baselines (including some that use the TQRL setting from [Mehta et al., 2022c], which is also known as RL with a generative model in Kakade [2003] and other works [Azar et al., 2013, Agarwal et al., 2020]).

In particular, we compare the average return across five evaluation episodes across five random seeds of each algorithm on five closed-loop control problems. For sample complexity we assess the median amount of data taken by each algorithm to 'solve' the problem across five seeds with the threshold performance given by an MPC controller using the ground truth dynamics. We evaluate the open-loop variant of our method, oTIP, against three comparison methods on three control problems suitable for open-loop control. In particular, to be suitable for open-loop control, the problem cannot be dynamically unstable (as Pendulum and Cartpole famously are) and must have a relatively short control horizon and fixed start state. Here too, we assess the average return as open-loop trials are conducted as well as the number of timesteps required to achieve 'solved' performance.

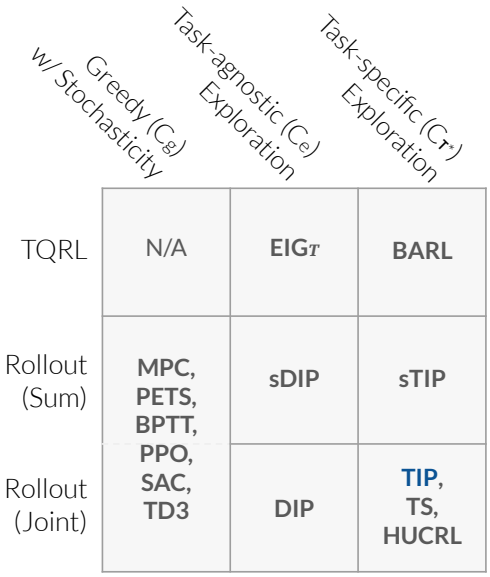

Figure 2.4: Our comparison methods can be broken down by the type of cost function used and how the methods do or do not handle sequential acquisition of information. As C_g is a sum, it naturally handles future timesteps jointly. For the other information quantities, it is possible to upper-bound information acquired by summing each separate mutual information, or to compute them jointly.

Comparison Methods We use several model-based and model-free comparison methods in this work. We compare to several published model-based methods. These include **PETS** [Chua et al., 2018], as implemented by Pineda et al. [2021], which uses a probabilistic ensemble of neural networks and CEM over particle samples to do MPC. We also compare against three model-based techniques from the HUCRL [Curi et al., 2020] implementation: **HUCRL** itself, which relies on hallucinating dynamics perturbations as a way of realizing an upper confidence bound on the policy, model-based Thompson Sampling (**TS**), which samples from the posterior over models

Environment	TIP	sTIP	DIP	MPC	PETS	SAC	FEEF	RHC	HUCRL	TS	BARL	EIG_T
Pendulum	**21**	36	36	46	5.6k	7k	800	>40k	>50k	>50k	**21**	56
Cartpole	131	141	161	201	1.63k	32k	>2.5k	>5k	>6k	>6k	**111**	121
β Tracking	**46**	76	276	76	330	12k	300	>3k	480	420	186	>1k
β + Rotation	**201**	>500	>500	>500	400	30k	>2k	>2k	>5k	>5k	>500	>1k
Reacher	**251**	>400	>1k	751	700	23k	>5k	1.5k	6.6k	4.5k	**251**	>1.5k

Table 2.2: **Sample Complexity:** Median number of samples across five seeds required to reach 'solved' performance, averaged across five trials. We determine 'solved' performance by running an MPC policy (similar to the one used for evaluation) on the ground truth dynamics to predict actions. We record $> n$ when the median run is unable to solve the problem by the end of training after collecting n datapoints. The methods in the rightmost section operate in the TQRL setting and therefore have more flexible access to the MDP dynamics for data collection. The full set of methods are shown in Section A.8 as well as boxplots depicting the data in Figure A.2.

and chooses optimal actions for that sample, and a greedy model-based neural network method relying on backpropagation through time (**BPTT**). We also compare against the Free Energy of the Expected Future method from Tschantz et al. [2020], which treats directed exploration as a process of actively collecting information for inference on a reward-biased generative model. A further comparison is with Receding Horizon Curiosity (RHC) [Schultheis et al., 2020], which does online Bayesian system identification over a linear model in order to quickly find a model of the environment dynamics. Our model-free comparison methods, Soft Actor-Critic (**SAC**) [Haarnoja et al., 2018], an actor-critic method that uses an entropy bonus over the policy to encourage more exploration, and two others (TD3 and PPO), are in the appendix.

Finally, we compare against various ablations of the proposed method. These vary across two axes as described in Figure 2.4: the cost function they use and how they handle sequential queries. Besides these differences, they use the same GP model and iCEM planning algorithm with the same hyperparameters, so they are truly comparable methods. The three cost functions used are C_g, C_{τ^*}, and C_e. **DIP**, **oDIP**, and $\textbf{EIG}_T$ all use C_e but compute, respectively, the expected joint entropy of the action sequence, the sum of the pointwise entropies of the action sequence, and the individual pointwise entropies in the TQRL setting. These methods are very similar in spirit to [Shyam et al., 2019, Pathak et al., 2019] in that they plan for future information gain about the dynamics, but we chose to compare in a way that controls for difference in the model and planning algorithm. **MPC** uses C_g and is very close to the method in [Kamthe and Deisenroth, 2018]. Like TIP, **BARL** [Mehta et al., 2022c] and **sTIP** use C_{τ^*}. BARL operates in the TQRL setting and can therefore use the simpler EIG_{τ^*} acquisition function. sTIP investigates the use of $-\sum_{s_i,a_i \in S} \text{EIG}_{\tau^*}(s_i, a_i)$ as a cost function for planning. This computes individual information gains for each future observation without accounting for the information they may have in common and is therefore an overestimate of the joint information gain. In the open-loop setting we compare against **oDIP** and **oMPC**, the open-loop variants of DIP and MPC, and Bayesian optimization (**BO**) as implemented by Pedregosa et al. [2011]. oDIP plans an action sequence to minimize the joint C_e and executes the actions found for each open-loop trial, while oMPC does the same thing using C_g. We give additional details on the comparison methods in Section A.6.

Control Problems Our closed loop control problems are the standard underactuated **Pendulum** swing-up task (Pendulum-v0 from Brockman et al. [2016]) with 2D states and 1D actions, a **Cartpole** swing-up task with a sparse reward function, 2D s, and 1D actions, a 2-DOF robot arm

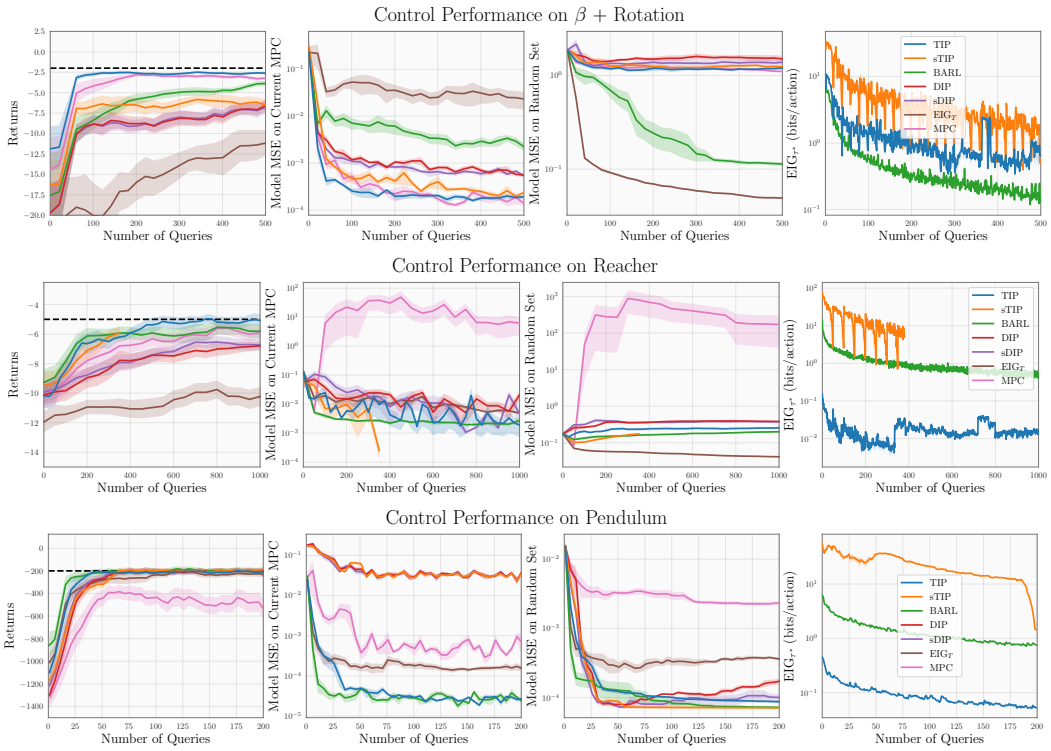

Figure 2.5: **Control and Modeling Details for TIP and Ablations.** Column 1: Learning curves for our ablation methods, all of which use the same planner and model. Column 2: Dynamics model accuracy on the points used by the planner to choose actions during MPC. Column 3: Dynamics model accuracy on a uniformly random test set in $\tilde{\mathcal{S}}$. Column 4: EIG_{τ^*} values normalized by the number of actions planned. sTIP was truncated on Reacher as it exceeded the wall time budget.

control problem where the end effector is moved to a goal (**Reacher**-v2 from Brockman et al. [2016]) with 10D states and 2D actions, a simplified β **Tracking** problem from plasma control [Char et al., 2019, Mehta et al., 2021] (similar in design but not identical to the one from Mehta et al. [2022c]) trained using with 4D states and 1D actions, and a more complicated problem in plasma control where β **+ Rotation** are tracked with 10D states and 2D actions. Our open loop control problems are a navigation problem with hazards (**Lava Path**) from [Mehta et al., 2022c] and two regulation problems with different **Nonlinear Gain** functions. The Lava Path problem has 4D states and 2D actions and the nonlinear regulation problems have 2D states and 2D actions. Full details on these problems are available in Section A.7.

Results As can be seen in Table 2.2, TIP is able to reach solved performance more quickly across the board than the model-based and model-free external baselines, often using a fraction or even orders of magnitude less data than other methods. For many of our ablation methods we see failures to solve some of the problems even though the model is demonstrated by TIP to be able to sufficiently predict the dynamics. This is especially apparent on the harder plasma control environment, β+Rotation, where TIP is the only method using our GP which is able to solve the problem. We believe that this is because the data acquired through exploration by the ablation methods is less useful for control than the data TIP collects. This is underscored by the second column of Figure 2.5, where it is clear that TIP achieves the lowest modeling error on the points actually needed during the execution of the policy but not on the uniform test set. In

particular we find it interesting that TIP outperforms BARL on the β + Rotation environment, as BARL should in principle have a strictly stronger access to the problem and is optimizing the same quantity with fewer constraints. We hypothesis that this may be due to the fact that BARL optimizes the acquisition function EIG_{τ^*} by simply uniformly sampling a set of points and choosing the one that evaluates to the largest value. Our more sophisticated optimization algorithm and forced initialization at the start state distribution seems to allow us to collect more information in this case. This interpretation is bolstered by the fact that on the problems where TIP outperforms BARL, we see that TIP is actually collecting more information per action than BARL as evidenced by larger EIG values. We also see clearly that there is value in computing the C_{τ^*} function rather than summing over EIG_{τ^*} values, as TIP outperforms sTIP across the board. Additionally, there is clear evidence for the value of task-specific exploration as the task-agnostic exploration methods (EIG_T, DIP, sDIP) underperform both in returns and model error on the trajectories visited.

For the open-loop experiments (Table 2.3), we also see strong performance from oTIP. As the model-based methods benefit from observing many model transitions for each open-loop trial, it is unsurprising that they are more sample-efficient than the BO method. Within the model-based techniques, oTIP is the most sample efficient. We believe that this is for much the same reasons as in the closed-loop case—exciting evidence that the C_{τ^*} cost function can be applied in a variety of settings.

Environment	oTIP	oMPC	oDIP	BO
Nonlinear Gain 1	**41**	91	51	210
Nonlinear Gain 2	**51**	61	>200	60
Lava Path	**41**	101	101	>2k

Table 2.3: **Open Loop Sample Complexity:** Median number of samples required to reach 'solved' performance, averaged across five trials. We determine 'solved' performance by running an MPC policy on the ground truth dynamics to predict actions. We record $> n$ when the median run is unable to solve the problem by the end of training after collecting n datapoints.

3 | Applying generalized decision-theoretic entropies to Bayesian RL

Given a prior on the dynamics of an MDP, the Bayes-optimal solution for minimizing cumulative regret given infinite computational resources is to solve the Bayes-adaptive MDP (BAMDP) [Duff, 2002], which explicitly models the evolution of the belief states of the agent as the dynamics unroll. That said, there are many real-world use cases where it is incorrect to optimize for cumulative rewards. Concretely, in any use case where a final Markovian policy is deployed after a 'practice' phase where performance is unimportant, an agent should instead aim to maximize the expected reward of the deployed policy. This is true in many competitions like chess or racing, as well as many applications in science, where samples are costly but we ultimately only care about the performance of the final method at test time rather than during training. Ideally, we'd like the agent to do as well as possible with the limited amount of information received using the finite exploration budget. Here, an agent ought to be motivated to take risks while exploring in order to learn vital information to use when deployed. In this work, we make precise some of these shortcomings of the BAMDP and present a closely related solution built from old ideas in Bayesian decision theory.

Over sixty years ago, DeGroot [1962] defined a generalization of entropy that applies to general decision making problems, parameterized by a loss function ℓ and action-set $\mathfrak{A}$. "$H_{\ell,\mathfrak{A}}$-*entropy*" is a measure of the information content of a probability distribution that is *relevant* for a particular decision making problem. This captures the critical idea that uncertainty-reducing information is valuable, but this uncertainty does not always affect the decision by an expectation-maximizing agent. For example, in a game of blackjack—where a total over 21 busts—a player who is holding 15 and is told the next card is uniformly drawn between 7 and K needs no additional information to make a decision (in this case, to stand) even though there is still uncertainty remaining in the value of the card. Here, the $H_{\ell,\mathfrak{A}}$-entropy of the next card is zero although the Shannon entropy would be ≈ 2. Recent work [Neiswanger et al., 2022] showed that the "*Expected* $H_{\ell,\mathfrak{A}}$*-information Gain*" (EHIG) is a generalization of several acquisition strategies from BO. For example, under straightforward choices for ℓ and $\mathfrak{A}$, choosing new data that maximizes the EHIG is equivalent to *expected improvement* [Močkus, 1975] and *knowledge gradient* [Frazier et al., 2008]—two of the most effective strategies for simple regret minimization in BO [Balandat et al., 2020].

In this chapter, we present an application of $H_{\ell,\mathfrak{A}}$-entropy to MDPs by presenting an exploration strategy for sample-efficient RL that aims to maximize the EHIG. We show theoretically that under our simple regret setting, solving this EHIG-MDP is the optimal way to explore, and that this strategy can outperform the BAMDP solution by a substantial amount. To verify this, we compare these two methods under an exact implementation on small tabular MDPs, and provide an empirical confirmation of our theoretical results.

In order to apply our method to continuous and high-dimensional problems where a prior is

much harder to specify, we borrow ideas from meta-learning to synthesize an effective prior. In particular, by learning from many environments drawn from a distribution, an agent can learn this distribution as a prior and infer against it. Inspired by previous work that uses meta-reinforcement learning for a scalable approximation to the BAMDP [Zintgraf et al., 2020], we develop a meta-RL method for approximately solving the EHIG-MDP. We demonstrate on four meta-RL tasks that these methods perform similarly when the exploration task is simple, and that our proposed method outperforms existing baselines when the required exploration behavior is more complex.

The full contributions of this chapter are as follows: **(1)** We show theoretically and empirically how the BAMDP, long viewed as the gold standard in Bayesian RL, is suboptimal for our simple regret setting. **(2)** Using methods from Bayesian decision theory involving generalized entropies, we derive a Bayes-optimal exploration algorithm for this setting and empirically demonstrate its benefits on an exact implementation with tabular MDPs. **(3)** Using techniques from meta-RL we develop a scalable method for higher-dimensional continuous spaces, and show its strong performance empirically against other Bayesian RL and meta-learning methods on multiple benchmark problems.

3.1 Related Work

$H_{\ell,\mathfrak{A}}$-Entropy Motivated by decision problems in optimal experimental design, past work on Bayesian decision theory has proposed a general family of of *decision-theoretic entropy* functions [DeGroot, 1962, Rao, 1984, Grünwald and Dawid, 2004], each parameterized by a problem-specific loss function ℓ and action set $\mathfrak{A}$. Methods in this area have aimed to maximally reduce this entropy for a model parameter, in order to carry out Bayes-optimal experimental design for different types of decision problems. In order to distinguish these entropy functions from the well-known Shannon entropy (which is a special case of this family), previous works have used the notation $H_{\ell,\mathfrak{A}}$*-entropy* which we also adopt in this paper.

In recent years, this family of $H_{\ell,\mathfrak{A}}$-entropy functions has been applied to estimating divergences between distributions [Zhao et al., 2021], and in Bayesian optimization [Neiswanger et al., 2022]. In the latter case, the $H_{\ell,\mathfrak{A}}$-entropy is used to define an expected information gain acquisition function for BO, in which new data is collected that maximally reduces the $H_{\ell,\mathfrak{A}}$-entropy, in expectation. By varying the loss and action set, this family can recover many common acquisition functions, including expected improvement [Močkus, 1975], entropy search [Villemonteix et al., 2009, Hennig and Schuler, 2012], and knowledge gradient [Frazier et al., 2008]. In this paper, we substantially extend the use of $H_{\ell,\mathfrak{A}}$-entropy to the MDP setting. In particular, our action under this framework is a choice of test-time policy and our loss is the negative expected return. In order to explore optimally, we combine the original ideas from DeGroot [1962] for multistep exploration with the expected MDP dynamics.

Meta-Reinforcement Learning As it is difficult to specify an accurate and tractable prior for high-dimensional and complex environments, a typical approach is to learn the distribution of possible MDPs by executing actions on many MDPs sampled from some distribution—in other words, to carry out meta-reinforcement learning.

One of the foundational methods, model agnostic meta-learning (MAML) [Finn et al., 2017], considers any model, parameterized by θ, trained to optimize for few-shot performance over a family of tasks. The gradient descent step for θ seeks to minimize total loss incurred by the model over a set of sampled tasks on a "look-forward" basis—as if the model has taken a single-task gradient

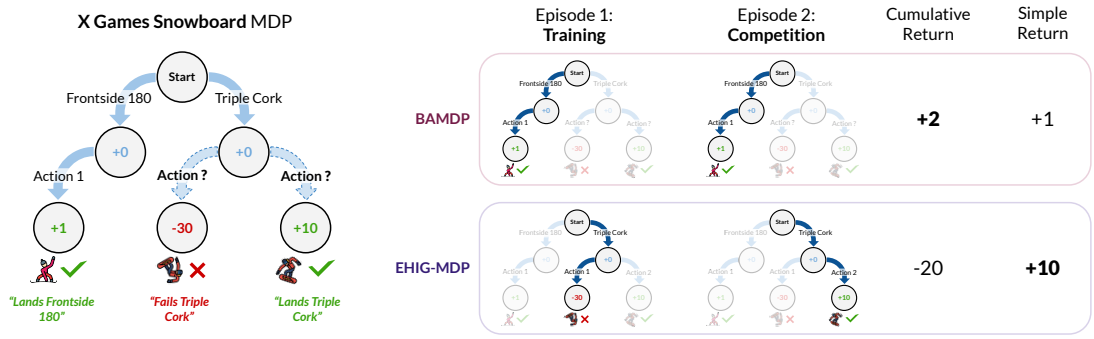

Figure 3.1: **Left:** An illustration of the *X Games Snowboard* MDP. If the *Triple Cork* trick is chosen, there is uncertainty about which action leads to landing or failing the trick. **Right:** Illustration of suboptimal exploration under the BAMDP, which aims to maximize cumulative return (even during training episodes), and improved exploration under the EHIG-MDP, which aims to maximize simple return (only during the competition episode).

descent step to θ_i. This is a classical way of doing meta-learning and unfortunately suffers from low sample efficiency.

Another method, RL^2, [Duan et al., 2016] constructs the learning process over a sequence of trials, each with a finite number of episodes of a task randomly drawn from a known task distribution. The policy model is fed the standard *state, action, reward* triple along with a termination flag signifying the end of a trial, and it's trained to optimize the rewards over the entire trial (not just a single episode). In this way, it pushes the model to optimize learning for a new task quickly within a limited number of episodes.

A more sophisticated approach to meta-reinforcement learning is to cast the problem as a Bayes Adaptive MDP [Duff, 2002], but this becomes computationally impossible for all but the smallest scale problems. Previous methods to get around this, such as posterior sampling [Osband et al., 2016b], often stray too far from Bayes optimal behavior. In VariBAD [Zintgraf et al., 2020], the BAMDP is set up for meta-learning as a tractable approximate problem in two steps: (i) details of an environment's reward and transition functions are mapped to embeddings in latent space via a learned model; (ii) the training objective for this model is also approximated by its evidence lower-bound, which is much easier to compute. However, as we discuss in this paper, the BAMDP solution attempts to trade off between exploration and exploitation. This can be suboptimal when the primary goal is to find and deploy a good test policy.

3.2 Problem Setting

We consider the Bayesian RL setting where we have a distribution over MDPs $\langle \mathcal{S}, \mathcal{A}, T, R, p_0, h \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $T : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ is a stochastic transition function, $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is a reward function, $p_0 \in \mathcal{P}(\mathcal{S})$ is the initial state distribution, and $h \in \mathbb{N}$ is the episode length. As we operate in the Bayesian setting, we assume that the transition and reward function are distributed according to a prior belief $b_0 = p(R, T)$ in a belief space $\mathcal{B}$.

We assume that the agent can perform inference and maintain beliefs $b_{\tau:t} = P(R, T \mid \tau_{:t})$, which correspond to the posterior MDP given the agent's experience $\tau_{:t} = \{(s_0, a_0, s'_0, r_0), \ldots, (s_t, a_t, s'_t, r_t)\}$. The agent collects experience by exploring for H timesteps in episodes of length h. A Bayesian agent in this setting is specified by two policies: an exploration policy $\pi_x : \mathcal{S} \times \mathcal{B} \to \mathcal{A}$ and an exploitation (test) policy $\pi_e : \mathcal{S} \times \mathcal{B} \to \mathcal{A}$. The aim of an agent is to maximize the performance of the exploitation policy when passed the belief obtained by executing the exploration policy on an

environment sampled from the prior. Formally, the goal is to maximize the *simple return*

$$J(\pi_x, \pi_e) = \mathbb{E}_{R,T\sim b_0}\left[\mathbb{E}_{\tau_{:H}|R,T,\pi_x}\left[J_{R,T}(\pi_e(\cdot, b_{\tau_{:H}}))\right]\right] \tag{3.1}$$

where trajectories $\tau_{:H}$ are generated inductively by $\tau_{:t} = \{(s_t, a_t, s'_t, r_t)\} \cup \tau_{:t-1}$, $\tau_{:0} = \emptyset$, $s_t = s'_{t-1}$ (or $s_t \sim p_0$ if t is a multiple of h), $a_t = \pi_x(s_t, b_{\tau:t-1})$, $s'_t \sim T(s_t, a_t)$, $r_t = R(s_t, a_t, s'_t)$, and

$$J_{R,T}(\pi) = \mathbb{E}_{s_0\sim p_0,\ \tau_h|\pi,R,T,s_0}\left[\sum_{t=0}^{h-1} r_t\right], \tag{3.2}$$

where τ_h is a h-step trajectory induced by the policy π and an initial state s_0, under a given (R, T), and $r_t = R(s_t, a_t, s_{t+1})$ is the reward collected at time step t. We also refer to the *simple regret* of π_x, π_e to mean the exploration-aware optimality gap: $\max_{\pi_1,\pi_2} J(\pi_1, \pi_2) - J(\pi_x, \pi_e)$. Finally, we refer to the *cumulative return* as the rewards obtained by the exploration policy over the course of exploration. It is important to note that in this setup, no inference is performed at test time—in other words, the test policy given the final belief state from exploration is Markovian.

Intuitively, we want an exploration policy that finds a belief such that the test policy has the information it needs to perform well on expectation. In this work we note that, given a test-time belief b, there exists an optimal exploitation policy $\pi^*_e(\cdot, b) = \max_{\pi:\mathcal{S}\to\mathcal{A}} \mathbb{E}_{R,T\sim b}\left[J_{R,T}(\pi)\right]$ that achieves an expected return equal to the *Bayes return*, $\mathrm{BR}(b) = \mathbb{E}_{R,T\sim b}\left[J_{R,T}(\pi^*_e(\cdot \mid b)\right]$. Therefore, the primary problem addressed in this paper is finding an exploration policy π_x that reliably leads to a belief with high *Bayes return*. Another important characteristic of objective (3.1) to note is that it doesn't directly involve the optimal policies for any particular R and T—it rather involves the limits of the knowledge of an agent, given a prior distribution over MDPs and the ability to execute H consecutive actions.

3.3 Bayes-Adaptive MDPs are 'Fearful'

The Bayes-adaptive MDP is typically viewed as the optimal solution to a Bayesian RL problem given unlimited computation. In this section, we define the BAMDP and discuss why it fails in simple-regret settings where we have a practice or training regime, and give a theoretical statement characterizing these failures. In the BAMDP, the belief state is modeled as evolving alongside the environment state and the dynamics of the transition model are those consistent with the updated posterior belief. In particular, the BAMDP is constructed by considering hyperstates $s^+_t \in \mathcal{S}^+ = s_t, b_{\tau:t-1} \in \mathcal{S} \times \mathcal{B}$ that evolve according to dynamics $T^+(s^+_t, a_t) = \mathbb{E}_{T\sim b_{\tau:t-1}}\left[T(s_t, a_t)\right]\delta(b_{\tau:t})$, and rewards $R^+(s^+_t, a_t, s'^+_t) = \mathbb{E}_{R\sim b_{\tau:t-1}}\left[R(s_t, a_t, a'_t)\right]$. This augmented MDP is a well-formed MDP, albeit exponentially large ($O(H|\mathcal{S}|^H)$ for a generic Dirichlet-Multinomial prior), which in finite settings can be solved exactly using methods like policy iteration [Sutton and Barto, 1998].

Specifically, the exploration policy π^b_x for the BAMDP agent is the one that solves the BAMDP over H timesteps. Intuitively, the optimal policy in the BAMDP will maximize the rewards expected over the exploration budget of the agent. This is often the desired behavior, but in a simple regret setting this can lead to pathologies. For example, if an agent needs to take a risky decision one time in order to know the correct action, it may simply avoid the risk so as to avoid the probability of a bad outcome. In general, when the goal is to identify a good policy with a limited number of samples, we will show that the BAMDP strategy is suboptimal.

Consider the following toy example depicted in Figure 3.1: an agent is a snowboarder training for the *X Games*. On each run, it can either attempt the Triple Cork (a big trick) or a Frontside 180

(a small trick). The big trick is potentially much more valuable, but, if it is chosen, the agent is unsure which action (1 or 2) will lead to the large reward. If the agent chooses the wrong action, it will fail to complete the trick, which incurs a large negative reward. A BAMDP agent given an exploration budget of one episode prior to test time will choose the Frontside 180 twice as this leads to a higher expected return over a window of 2 episodes (i.e., one for training, and one for the X Games competition). Clearly, an optimal agent will attempt the Triple Cork in practice, find out which action leads to the large reward, and execute the correct action on the Triple Cork at the X Games. This conservatism is in fact an essential component of the BAMDP which makes it unsuitable for the Bayesian simple regret setting, a fact that we capture in the following theorem.

Theorem 1. *For every number of states $|\mathcal{S}| \geq 7$, exploration budget $H > |\mathcal{S}| - 2$ and $\epsilon > 0$ there exists a distribution over MDPs with $|\mathcal{S}|$ states with rewards bounded in $[0, 1]$ such that*

$$\max_{\pi_x} J(\pi_x, \hat{\pi}_N) - J(\pi_x^b, \hat{\pi}_N) \geq \frac{1}{2\lfloor \frac{H}{|\mathcal{S}|-2} \rfloor} - \epsilon.$$

Note that the optimality gap we are lower bounding here is between the BAMDP exploration policy and the best possible exploration policy given the true prior and a particular budget, not that between the BAMDP policy and the optimal policy for an MDP. We would ideally see no regret at all in this comparison and in fact achieve this through our algorithm in the following section. Of course, over a long horizon of data acquisition, the lower bound on simple regret will approach zero, but in many settings where sample efficiency is paramount these asymptotic equivalences are not helpful.

The proof of this theorem is in Section B.1.1. At a high level, we prove this by making the notion of the X Games problem precise and showing that the Q values of the BAMDP policy will force it to be conservative. The solution to this pathology of the BAMDP policy in the simple regret setting is to adjust the reward function. As it turns out, the correct reward function can be derived from the notion of the expected H-information gain.

3.4 Fearless RL Through $H_{\ell,\mathfrak{A}}$-Information

In order to generally solve the problems with the BAMDP, we take a step back and consider a generic Bayesian decision making problem. Say we have a prior distribution b_0 over functions $f : \mathcal{X} \to \mathcal{Y}$, an action space $\mathfrak{A}$, and a loss function $\ell : (\mathcal{X} \to \mathcal{Y}) \times \mathfrak{A} \to \mathbb{R}$. In general, the goal is to acquire data that allows us to find an action $a \in \mathfrak{A}$ that minimizes $\ell(f, a)$ for the true function f, which is a draw from our prior. Assuming no further data can be collected and the current belief is b, a Bayesian agent would prefer the *Bayes action*, $\operatorname{argmin}_{a \in \mathfrak{A}} \mathbb{E}_{f \sim b} [\ell(f, a)]$ which achieves on expectation the *Bayes Risk*. As initially described in DeGroot [1962], the Bayes Risk can be viewed as a generalized kind of entropy—the usual Shannon entropy is a special case (if $\mathfrak{A} = P(\mathcal{X} \to \mathcal{Y})$ and $\ell(f, a) = -\log a(f)$, see Neiswanger et al. [2022] for details) and many important properties of entropy (such as concavity and monotonic decrease as observations are made) are preserved for other settings of ℓ and $\mathfrak{A}$. Suppose now that the agent is capable of making an observation y_x of the function value at a point $x \in \mathcal{X}$ of its choosing. From that observation, the agent can infer an updated belief $b' = p(f \mid b, y_x)$. As established in these prior works, an agent would be well-justified in choosing to observe the point that maximizes the *expected $H_{\ell,\mathfrak{A}}$-information gain* (EHIG), or equivalently, the point that maximally improves the Bayes risk. This can be written

$$\text{EHIG}_{\ell,\mathfrak{A}}(x \mid b) = \min_{a \in \mathfrak{A}} \mathbb{E}_{f \sim b} [\ell(f, a)] - \mathbb{E}_{y_x \sim p(y_x \mid b)} \left[\min_{a \in \mathfrak{A}} \mathbb{E}_{f \sim b'} [\ell(f, a)] \right]. \tag{3.3}$$

Per DeGroot [1962], this quantity is nonnegative and directly captures the expected improvement in the decision problem of interest. It can also be extended to address additional observations. In the following section, we give an application of this general decision making objective to Bayesian MDPs under the simple regret setting.

3.4.1 EHIG for MDPs

Suppose we are operating in the Bayesian RL setting presented in Section 3.2. What Bayesian decision problem might this be? Ultimately, the decision to be made is a choice of *policy* for the evaluation given the eventual belief. This decision will be evaluated on the expected return of the policy choice. Put more formally, in our Bayesian decision problem, $\mathfrak{A}$ is the set of all policies $\pi : \mathcal{S} \to \mathcal{A}$, f is the Cartesian product of functions R and T, and $\ell(R, T, \pi) = -J_{R,T}(\pi)$. Put in simple language, the Bayes risk here is the negative Bayes return from Section 3.2. The H-information is measured by the improvement in the Bayes return as data are acquired from the MDP.

In our simple regret setting, the goal is to take actions such that the eventual Bayes return is large. With that in mind, we can construct an exploration MDP that takes into account both the posterior dynamics of the MDP as well as the evolution of the belief as observations are made. This will be similar to the BAMDP but instead will focus on the Bayes return of the test policy. We note that this EHIG-MDP has the same state space, action space, and dynamics as the BAMDP. This is the natural way to represent the multistep inference required for optimal exploration in our setting.

Since the $H_{\ell,\mathfrak{A}}$-information is equal to the negative Bayes return of the belief, and the initial Bayes return is a constant, the $H_{\ell,\mathfrak{A}}$-information gain from exploration will be a constant offset from the Bayes return of the final belief. Given this, in the EHIG-MDP, all rewards are zero with the exception of the final timestep. For this final timestep, the reward will simply be the Bayes return of the final belief; i.e., for the state $s^+ = [s; b]$ that the policy transitions to at the Hth step, the reward is $\mathrm{BR}(b)$. This MDP encourages a Bayesian agent to maximize the joint expected $H_{\ell,\mathfrak{A}}$-information gain of all observations during exploration. The EHIG-maximizing policy π_h that solves the EHIG-MDP has the desirable property that it is optimal according to objective (3.1). We can formalize this intuition with the following theorem, showing that the EHIG-MDP rectifies the problems with the BAMDP.

Theorem 2. *For all distributions over MDPs b_0 and exploration budgets H, $J(\pi_h, \hat{\pi}_N) = \max_{\pi_x} J(\pi_x, \hat{\pi}_N)$.*

The proof of this theorem is in Section B.1.2. At a high level we prove this result via induction on the $Q_H^{\pi_h}$ function through time and show that for any number of remaining timesteps, executing π_h will maximize the eventual test-time Bayes return.

3.4.2 Exactly Solving the EHIG-MDP in Tabular Cases

As we present in Algorithm 4, the overall process of exploration is simple. We compute or approximate the EHIG policy π_h and we execute it on the environment while performing (possibly approximate) inference at every step. At test time we execute or approximate the Bayes policy given the up-to-date belief. In this section we discuss the exact algorithm, which we empirically investigate on a pair of distributions over finite MDPs that highlight the advantages of EHIG-based exploration. When operating in a tabular setting, we can exactly compute the quantities involved in our EHIG-MDP in order to explore optimally. However, this ends up being extremely computationally costly. This method in its exact form is only practical to use on small finite MDPs as the computational complexity makes even dozens of states or timesteps difficult to contemplate.

In particular, we represent a table of size $O(H|\mathcal{S}|^H)$ for each state-belief pair possible in the problem. We then can compute the Bayes return for every terminal belief (attainable after making H observations) by performing policy iteration [Sutton and Barto, 1998] on the MDP implied by the posterior predictive distribution [Ghavamzadeh et al., 2016]. From these Bayes returns and the (known) dynamics of the EHIG-MDP, we can again use policy iteration to find the EHIG policy π_h. For the uncorrelated Dirichlet priors typically used in the tabular setting this will incur no error in computing returns.

3.4.3 Empirical Performance of the Exact EHIG Policy

In order to investigate empirically the conclusions of Theorems 1 and 2, we implemented the exact EHIG-MDP and the BAMDP algorithms and applied them to a pair of small tabular MDP priors. Due to the heavy computational requirements of the exact implementation we implemented these methods in Julia[1] to take advantage of compilation and effective multithreading. In particular, we explicitly represent the belief MDP and its value functions through large matrices and apply a combinatorial indexing scheme to map rows of these matrices to particular states in the belief MDP. We give details on this setup in Section B.2.

We execute the exact BAMDP and EHIG-MDP agents on two tabular distributions over environments *LavaRun* and *SkateTrick* depicted in Figure 3.2 for five timesteps of exploration. Both of these environments are accompanied by Dirichlet-Multinomial conjugate priors which provide a distribution over the correct reward and dynamics with some uncertainty that requires exploration to resolve. We evaluate the agents on the two environments supported by the belief distribution, where a different action corresponds to each arrow on the graph and we plot the returns achieved over varying exploration budgets. In both of these environments, there is a single bit of information needed in order to be able to reliably achieve the highest returns. However there is some cost associated with finding that information, which dissuades the BAMDP policy from finding it. In the SkateTrick environments, the BAMDP agent chooses the conservative choice of the left branch in order to avoid the uncertainty in the right branch, even though the return achievable by making the correct decision is much greater. In the LavaRun environments there are two paths to an uncertain decision with negative expected value; the BAMDP agent takes the longer path and even suffers the negative reward in state 2 in order to avoid the difficult decision at state 4. This can be seen in the fact that the BAMDP agent requires more exploration to solve the MDP.

Algorithm 4 Exploration via the EHIG-MDP

Inputs: Prior belief b_0, exploration budget H.
Initialize $b \leftarrow b_0$, $\tau_0 = \emptyset$.
Compute π_h for b and H. ▷ Compute the EHIG-maximizing policy π_h
for $t \in [1, \ldots, H]$ **do**
  Let $a_t = \pi_h(s_t, b)$. ▷ Choose an action according to π_h
  Observe $s'_t \sim T(s_t, a_t)$ and $r_t \sim R(s_t, a_t, s'_t)$. ▷ Take a step in the environment
  Let $\tau_t = \{(s_t, a_t, s'_t, r_t)\} \cup \tau_{t-1}$, $b = b_{\tau_t}$. ▷ Update the experience and beliefs
end for
return $\hat{\pi}_N(\cdot, b)$. ▷ Return the optimal exploitation policy

[1] https://julialang.org/

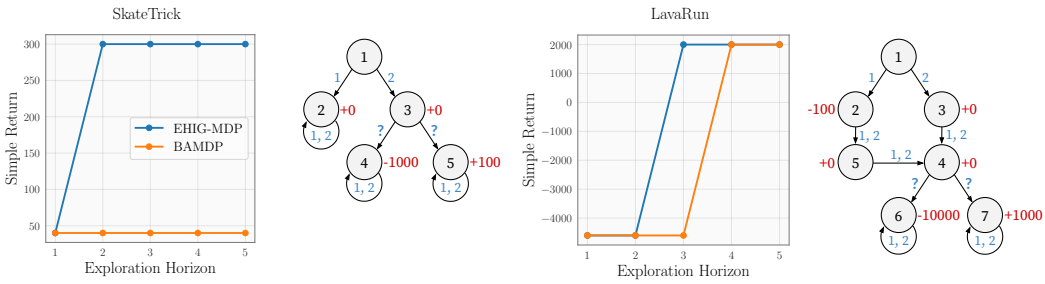

Figure 3.2: We plot the simple return of agents with BAMDP and EHIG exploration policies (exact implementations without approximation) on two tabular MDP distributions. In the LavaRun environment, the BAMDP agent fails to solve the problem with horizon 3 because it is unwilling to 'cross lava' in order to obtain the necessary solution. In the SkateTrick environment, even with a 5 step horizon, the BAMDP agent is not willing to try the dangerous trick in order to learn how to 'land it' at test time.

3.5 Scalable Approximation of the EHIG-MDP

In order to apply this EHIG strategy to MDPs of interest for many applications, we must be able to scale these methods to problems with continuous and even high-dimensional state and action spaces by approximating the various inference and policy optimization components of the EHIG-MDP algorithm. At a high level, an EHIG agent needs to be able to approximately infer the current belief $b_{\tau:t}$ after observing a trajectory τ taken from a particular $T, R \sim b_0$. It then needs to estimate two different policies: the EHIG policy π_h and the Bayes policy $\hat{\pi}_N$ that is able to perform optimally given a particular belief. As with the finite setting, we take inspiration from the BAMDP in order to design an implementation capable of satisfying these conditions.

For larger MDPs it is in general difficult to specify a quality prior for which inference is tractable Fortuin [2022], Mehta et al. [2022b]. In order to have some source of data from which to learn the prior distribution of possible MDPs, we use a Bayesian formulation of meta-learning, following Zintgraf et al. [2020], for our continuous methods and experiments. In this setting, the agent is able to collect trajectories from environments drawn from b_0, in order to estimate beliefs over the reward and transition functions. During this *meta-training* phase, the agent executes actions and observes transitions and rewards with which to update its parameters for inference and decision making. At *meta-test* time, a new environment is drawn from the prior and the agent executes H actions while performing inference, and is then evaluated on the performance of its estimate of the best policy given its current belief. At test time, this setting is identical to the one discussed above where policies and beliefs can be exactly calculated, in that there is a short exploration phase of H timesteps on a new environment prior to the estimated best Markovian policy being evaluated. However, for continuous spaces we require the meta-training phase in order to learn to explore and solve environments drawn from this distribution.

As discussed in Section 3.1, Zintgraf et al. [2020] introduced many of these ideas with the VariBAD method. In this work, we modify the VariBAD approach so that it optimizes the EHIG objective. As in VariBAD, we assume a normal prior on a latent variable $m \sim \mathcal{N}(0, I)$ that is fed into a multiheaded decoder with heads $s'_t \sim p^T_\theta(m, s_t, a_t)$ and $r \sim p^R_\theta(m, s_t, a_t, s'_t)$ and use a GRU [Chung et al., 2014] based encoder $q_\phi(\tau)$ that produces a variational belief b parameterizing the posterior $p(m \mid b)$ given a trajectory. The encoder is trained via maximization of the ELBO lower bound on the likelihood as in Kingma and Welling [2013]. In the VariBAD method, the policy is trained via PPO to maximize the discounted future returns over the course of exploration and inference.

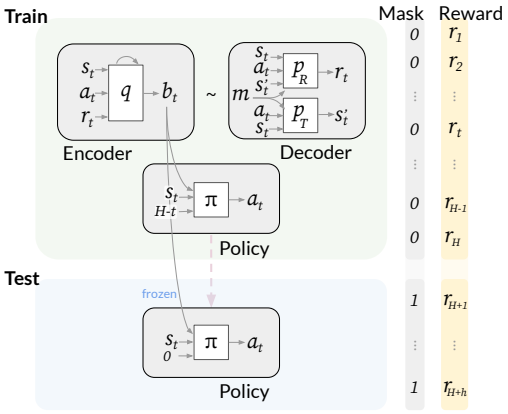

Figure 3.3: *Architecture for scalable EHIG-MDP.* During a particular meta-RL trial, the recurrent encoder q processes all previous training data into an up-to-date belief state b_t, which is passed to the policy π at train time, along with the current MDP state and the number of exploration timesteps remaining. After H timesteps of exploration have elapsed, the belief state is frozen and passed to the policy along with the MDP state and 0 (denoting the exploration remaining). The rewards used for policy optimization are masked for data collected during training but not for test-time data, though they are always visible to the encoder and decoder. Our decoder and encoder architecture builds on Zintgraf et al. [2020].

As we aim to address the simple regret setting with a reflex policy at test time, we only perform inference during the exploration phase. In order to accomplish the EHIG objective, we train the policy using PPO [Schulman et al., 2017] with a concatenated trajectory of exploration and test data *with the exploration rewards masked*. In order to make the state space Markovian given this change, we augment the state space with the number of remaining exploration timesteps counting down from H to 0. We write this as $\pi_\theta(s, b, t)$ for notational convenience. We illustrate our full architecture in Figure 3.3.

This addition breaks the policy into essentially two functions: at test time (with $t = 0$) the policy is passed the current state and a constant belief vector, and aims to maximize the returns of the trajectory as is typical in PPO; while during exploration (with $t > 0$) the policy is incentivized to find belief states b that will lead to good performance at test time. This can be easily seen by considering the value function of π_θ. As there are no rewards observed by π_θ for $t > 0$, the value function of the policy must depend on the expected returns after the exploration expires. Since the first state of the test trajectory is sampled from p_0, the final state of exploration will not matter. Concretely, $V(s, b, 1) = \mathbb{E}_{s',a,r|s,b,\pi_h}\left[\text{BR}(\{(s, a, r, s')\} \cup b)\right]$, exactly as desired for a policy maximizing EHIG.

3.5.1 Experiments Using the EHIG Approximation

We execute the scalable approximation to the EHIG-MDP method on four meta-reinforcement learning environments alongside three baselines in order to evaluate its performance on continuous environments. Here we set H to be one episode. We executed all policies on meta-training episodes consisting of one exploration and one test episode. At meta-test time we also execute one exploration episode each across sixteen environment samples. After the exploration episode, we evaluate the performance of the Markovian test-time policy from each method by executing it for 10 episodes and averaging the performance. As meta-training progresses, we evaluated the policy with a meta-test trial every 16,000 frames of experience. We provide additional details on experiments in Section B.3.

Environments We run our meta-RL algorithms on four environments: **HalfCheetahVel**, **SparsePoint**, **LavaPoint**, and **BetaLimit**. These are designed to test various aspects of exploration in meta-RL. HalfCheetahVel and SparsePoint were borrowed from Zintgraf et al. [2020]. In particular, HalfCheetahVel is the standard HalfCheetah task from Brockman et al. [2016] but with a reward

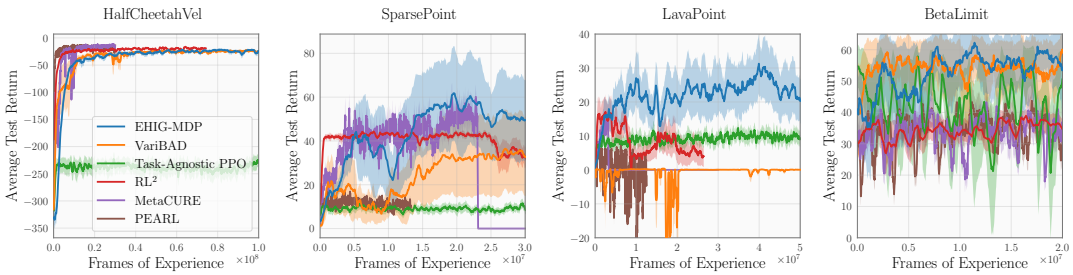

Figure 3.4: *Simple returns of each meta-learning method over the course of meta-training*. At each evaluation, we plot the mean performance over 16 environment samples and 10 evaluation episodes of the test policy after one exploration episode in each environment, and smooth using a moving average. We shade error regions corresponding to the standard deviation of our return estimates over 5 random seeds.

function that penalizes the difference between the current velocity and a target velocity that is uniformly sampled for each MDP sample. SparsePoint is a 2d position control problem with sparse rewards where an agent is only rewarded in the immediate vicinity of the goal, which is randomly sampled from a semicircle in the domain. LavaPoint is similar to SparsePoint, except that a portion of the domain is 'lava' which severely penalizes the agent for entering it. BetaLimit is a nuclear fusion environment similar to those used in Mehta et al. [2022a] in which the 'β-limit' varies across episodes. β is a figure of plasma performance often used nuclear fusion. However an overly high β can lead to instabilities and plasma disruptions. The maximal achievable β on a particular reactor may vary between days or even shots based on factors about the machine that may or may not be observable. In this environment, the agent aims to maximize β by controlling simulated plasma actuators while not exceeding a randomly sampled β-limit for which it is penalized.

Baselines Besides the EHIG method, we compare against five other baselines. The most closely related baseline is VariBAD [Zintgraf et al., 2020], which as discussed uses a similar inference and policy optimization framework to approximate the BAMDP policy. We also compare against RL^2, a foundational meta-RL algorithm that uses a recurrent neural network to represent the policy for a particular meta-RL trial and updates the weights using the TRPO algorithm [Schulman et al., 2015]. Due to its high computational cost, we were not able to run RL^2 for as many frames of experience as other methods. We additionally compare against MetaCURE [Zhang et al., 2021], which uses an exploration policy that maximizes mutual information between the trajectory and the observed task to efficiently explore and a similar test policy to ours. Another baseline we include is PEARL [Rakelly et al., 2019], which uses a similar variational model for task inference and conducts approximate posterior (Thompson) sampling for exploration. Finally, we compare against a task-agnostic implementation of PPO [Schulman et al., 2017] in order to ascertain whether our meta-RL methods are doing anything more sophisticated than simply attempting to learn a generally good policy.

Results We observe in Figure 3.4 that the EHIG-MDP policy generally performs well across our environments. For the HalfCheetahVel problem where the exploration is extremely simple (a subtraction suffices), all the meta-learning methods are able to quickly and effectively solve the problem. For the SparsePoint environment, there is general difficulty in solving the environment but we see that the most effective exploration is through the EHIG-MDP method and the MetaCURE method which explicitly gathers information about the task during exploration. This is also true

for the LavaPoint environment, where VariBAD performs poorly. As this environment is similar to the LavaRun environment in Section 3.4.3, it is unsurprising that this method is stymied here. MetaCURE also fails here—we imagine that this could be due to the Shannon information over the task identification being less robust than H-information, and warrants further study. The RL^2 method also shows unstable performance on the LavaPoint environment, which may be due to the stark differences in reward inside and outside of the lava region. Finally, on the BetaLimit environment, we see that EHIG and VariBAD are both best able to handle the exploration and policy learning to achieve a relatively high β value while remaining relatively stable and avoiding going over the β limit.

In general, we see these results as encouraging that even when approximated, the EHIG-MDP is able to explore efficiently and perform well in the simple regret setting. We include additional information on experiments in Appendix Section B.3.

4 | Efficiently identifying the value function implies sample-efficient decision making

Often, applications of reinforcement learning require algorithms that are capable of learning and planning in *large state spaces*. Many existing approaches require a large amount of training data to obtain good policies, and efficient active exploration in large state spaces is still an open problem. Moreover, when deploying policies trained on simulators in real-world applications, a crucial requirement is that the policy performs well in *any* state that it might encounter. In particular, at training time, the learning approach has to sufficiently explore the state space. This is of particular importance when at test time, the controller must be deployed in real-world settings where a guarantee on policy performance would provide confidence.

In this chapter, we first formally study the active RL setting. Our objective is to learn a near-optimal policy by actively querying the simulator with a state-action pair chosen by the learning algorithm. Our method queries the state where uncertainty about the value function is maximized. We prove that this method identifies a near-optimal policy efficiently and empirically validate the performance of our algorithm in both the MDP and contextual BO settings, where it outperforms a set of strong baseline methods in returns after a small number of samples. Inspired by previous works, we make a structural assumption in the kernel setting, which states that the Bellman operator maps any bounded value function to one with a bounded reproducing kernel Hilbert space (RKHS) norm. In particular, this assumption implies that the reward and the optimal Q-function can be represented by an RKHS function. We propose a novel approach based on least-squares value iteration (LSVI). The algorithm is designed to actively explore uncertain states based on the uncertainty in the Q-estimates, and makes use of optimism for action selection and pessimism for estimating a near-optimal policy.

Next, we extend the selection rule of this method to the *dueling contextual bandit* setting by incorporating ideas from Xu et al. [2020], where feedback is only available as a comparison between two chosen actions. We are able to recover similar theoretical guarantees and demonstrate that they hold in a synthetic setting.

Contributions We propose a novel kernelized algorithm for best policy identification in reinforcement learning with a generative model. Our sampling strategy actively explores (i) states for which the best action is the most uncertain and (ii) the corresponding "optimistic" actions. We prove sample complexity guarantees for finding an ϵ-optimal policy *uniformly over any given initial state*. Our bounds scale with the maximum information gain of the corresponding reproducing kernel Hilbert space but *do not* explicitly scale with the number of states or actions. When specialized to the offline contextual Bayesian optimization (BO) setting [Char et al., 2019], we improve upon sample

complexity guarantees from prior work. Finally, we include experimental evaluations on several RL and BO benchmark tasks. The former of these includes one of the first empirical evaluations of the model-free optimistic value iteration algorithms with function approximation [Yang et al., 2020].

4.1 Related Work

Reinforcement learning with function approximation dates back to at least [Bellman et al., 1963, Daniel, 1976, Schweitzer and Seidmann, 1985]. A majority of work is in the *online* setting where the learning agent interacts with the environment while (typically) minimizing regret. Upper confidence bound algorithms, originally developed in the bandit setting [Lattimore and Szepesvári, 2020] (also, frequently used in the related setting of best-arm identification, e.g., [Gabillon et al., 2011, Kalyanakrishnan et al., 2012, Soare et al., 2014]) , have been successfully applied to tabular Markov decision processes (MDPs) [Auer and Ortner, 2006, Auer et al., 2008], and extended to RL with function approximation. Jin et al. [2020] propose the LSVI-UCB algorithm in the linear MDP setting that achieves a near-optimal regret bound. Yang et al. [2020], Domingues et al. [2021] extend this work to the non-linear function approximation setting. These works are closely related to ours in that we make use of LSVI and confidence bounds for the Q-function in the kernelized setting. Unlike previous works, we consider the generative model setting and derive bounds on the sample complexity that hold uniformly over the initial state. There are many more alternative parametric models that admit sample efficient algorithms [e.g., Ayoub et al., 2020, Zhou et al., 2021, Du et al., 2021, Zanette et al., 2020, Liu and Su, 2022].

In the *generative model* setting, the learner has access to a simulator that for any given state-action pair returns a next-state sample from the transition kernel. This provides additional flexibility to obtain data from states that are otherwise hard to reach in the environment. For the tabular case, matching upper and lower bounds are shown by Azar et al. [2012, 2013]. In the generative model setting with function approximation, Lattimore et al. [2020] show that policy iteration can be used to compute a near-optimal policy given features such that the Q-function of any policy can be approximated by a linear function. Their algorithm uses a D-experimental design to roll out policies from a sufficiently diverse set of states. The POLITEX algorithm [Abbasi-Yadkori et al., 2019, Szepesvári, 2022] can be used in lieu of policy iteration and leads to tighter bounds on the approximation error. A similar approach based on LSVI is analyzed by Agarwal et al. [2019, Chapter 3].

An important special case of the MDP setting is the *contextual bandit setting*. When combined with linear function approximation, this recovers the contextual linear bandit setting [Abbasi-Yadkori et al., 2011], and contextual Bayesian optimization when using kernel features [Srinivas et al., 2009, Krause and Ong, 2011]. Various works consider the case where the learner has control over the choice of context during training time. Char et al. [2019] propose a variant based on Thompson sampling. Pearce and Branke [2018], Pearce et al. [2020] also propose variants that leverage ideas from the knowledge gradient [Frazier et al., 2008]. The latter works lack theoretical guarantees, while our result (from section 4.5) improves upon the sample complexity guarantee of Char et al. [2019]. The approach by Kirschner et al. [2020] for the distributionally robust setting can be specialized to our setting, in which case they recover similar bounds but only for a fixed context distribution.

4.2 Problem Setting for AE-LSVI analysis

We consider a time-varying episodic MDP $\left(\mathcal{S}, \mathcal{A}, H, (T_h)_{h\in[H]}, (r_h)_{h\in[H]}\right)$ with state space $\mathcal{S}$, action space $\mathcal{A}$, horizon $H \in \mathbb{N}$, Markov transition kernel $(T_h)_{h\in[H]}$ and deterministic reward functions $(r_h : \mathcal{S}\times\mathcal{A} \to [0,1])_{h\in[H]}$. In particular, for each $h \in [H]$, we let $T_h(s,a)$ denote the probability transition kernel when action a is taken at state $s \in \mathcal{S}$ in step $h \in [H]$. A *policy* consists of H functions $\pi = (\pi_h)_{h\in[H]}$ where for all $h \in [H]$, $\pi_h(\cdot|s)$ is a probability distribution over the action set $\mathcal{A}$. In particular, $\pi_h(a|s)$ is the probability that the agent takes action a in state s at step h.

We assume the active access model, in which the agent interacts with the environment in the following way: Let N denote the number of episodes and H the horizon, i.e., the number of steps in each episode. Then for each $t \in [N], h \in [H]$, the agent chooses $s_h^t \in \mathcal{S}$, $a_h^t \in \mathcal{A}$, obtains the reward $r_h(s_h^t, a_h^t)$ and observes the new state $s'_{h,t} \sim T_h(s_h^t, a_h^t)$.

The goal is to find an ϵ-optimal policy while minimizing the number of necessary episodes N. More precisely, for a fixed precision $\epsilon > 0$ and horizon $H \in \mathbb{N}$, the goal of the learner is to output a policy $\hat{\pi}_N$ after a suitable number of episodes $N > 0$ such that $\|V_1^* - V_1^{\hat{\pi}_N}\|_{\ell^\infty(\mathcal{S})} \le \epsilon$.

Finally, we also recall the Bellman equation that is associated to some policy π:

$$V_{H+1}^\pi = 0, \quad Q_h^\pi(s,a) = r_h(s,a) + \mathbb{E}_{s'\sim T_h(\cdot|s,a)}[V_{h+1}^\pi(s')], \quad V_h^\pi(s) = \mathbb{E}_{a\sim\pi_h(a|s)}[Q_h^\pi(s,a)], \tag{4.1}$$

and the Bellman optimality equation:

$$V_{H+1}^* = 0, \quad Q_h^*(s,a) = r_h(s,a) + \mathbb{E}_{s'\sim T_h(\cdot|s,a)}[V_{h+1}^*(s')], \quad V_h^*(s) = \max_{a\in\mathcal{A}} Q_h^*(s,a). \tag{4.2}$$

It follows that the optimal policy π^* is the greedy policy with respect to $\{Q_h^*\}_{h\in[H]}$, a property that is going to be useful later on when defining our active exploration strategy. We use the reproducing kernel Hilbert space (RKHS) function class to represent functions such as the reward functions $\{r_h\}_{h\in[H]}$ and the optimal Q-functions $\{Q_h^*\}_{h\in[H]}$ (see the formal statement in Assumption 1). In particular, we consider a space of well-behaved functions defined on $\mathcal{X} = \mathcal{S}\times\mathcal{A}$, where $\mathcal{H}$ denotes an RKHS defined on $\mathcal{X}$ induced by some continuous, positive definite kernel function $k : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$. We also assume that (i) $\mathcal{X} \subset \mathbb{R}^d$ is a compact set, (ii) the kernel function is bounded $k(x,x') \le 1$ for all $x, x' \in \mathcal{X}$, and (iii) every $f \in \mathcal{H}$ has a bounded RKHS norm, i.e., $\|f\|_{\mathcal{H}} \le B_Q H$ for some fixed positive constant $B_Q > 0$.

4.3 AE-LSVI Algorithm

Our algorithm runs in episodes $t \in [N]$ of horizon H. As in the kernel least-squares value iteration [Yang et al., 2020], at the beginning of every episode t, it solves a sequence of kernel ridge regression problems based on the data obtained in the previous $t-1$ episodes to obtain value function estimates $\{\hat{Q}_h^t\}_{h=1}^H$:

$$\hat{Q}_h^t \in \operatorname{argmin}_{f\in\mathcal{H}} \left\{ \sum_{i=1}^{t-1} \left(r_h(s_h^i, a_h^i) + V_{h+1}^t(s'_{h,i}) - f(s_h^i, a_h^i)\right)^2 + \lambda\|f\|_{\mathcal{H}}^2 \right\}, \tag{4.3}$$

where λ is the regularization parameter. Recalling that $x \in \mathcal{X} \coloneqq \mathcal{S}\times\mathcal{A}$, the solution of the problem in (4.3) can be written in closed form as follows:

$$\hat{Q}_h^t(x) \coloneqq k_h^t(x)^T (K_h^t + \lambda I)^{-1} Y_h^t, \tag{4.4}$$

where $k_h^t(x) \in \mathbb{R}^{t-1}$, the kernel matrix $K_h^t \in \mathbb{R}^{(t-1)\times(t-1)}$ and observations $Y_h^t \in \mathbb{R}^{t-1}$ are given as follows:

$$k_h^t(x) \coloneqq [k(x_h^1, x), \ldots, k(x_h^{t-1}, x)],\ K_h^t \coloneqq \left[k(x_h^i, x_h^{i'})\right]_{i,i'\in[t-1]},\ [Y_h^t]_i \coloneqq r_h(s_h^i, a_h^i) + V_{h+1}^t(s'_{h,i}).$$

Next, we can also compute the uncertainty function $\sigma_h^t(\cdot,\cdot)$ in the closed form:

$$\sigma_h^t(s,a) = \tfrac{1}{\lambda^{1/2}}\big(k(x,x) - k_h^t(x)^T(K_h^t + \lambda I)^{-1}k_h^t(x)\big)^{1/2}. \tag{4.5}$$

We recall that each reward function is bounded in $[0,1]$. We use $[\,\cdot\,]_0^{H-h+1}$ to denote the truncation to the interval $[0, H-h+1]$ and we define the optimistic $\overline{Q}_h^t$ and pessimistic $\underline{Q}_h^t$ value estimates (i.e., upper and lower confidence bound of Q_h^*; see lemma 1 and lemma 2):

$$\overline{Q}_h^t(\cdot,\cdot) \coloneqq \left[\hat{Q}_h^t(\cdot,\cdot) + \beta\sigma_h^t(\cdot,\cdot)\right]_0^{H-h+1}, \quad \overline{V}_h^t(\cdot) \coloneqq \max_{a\in\mathcal{A}} \overline{Q}_h^t(\cdot,a), \tag{4.6}$$

$$\hat{Q}_h^t(\cdot) \coloneqq k_h^t(\cdot)^T(K_h^t + \lambda I)^{-1}\overline{Y}_h^t, \quad [\overline{Y}_h^t]_i \coloneqq r_h(s_h^i, a_h^i) + \overline{V}_{h+1}^t(s'_{h,i}). \tag{4.7}$$

Similarly, we have

$$\underline{Q}_h^t(\cdot,\cdot) \coloneqq \left[\check{Q}_h^t(\cdot,\cdot) - \beta\sigma_h^t(\cdot,\cdot)\right]_0^{H-h+1}, \quad \underline{V}_h^t(\cdot) \coloneqq \max_{a\in\mathcal{A}} \underline{Q}_h^t(\cdot,a), \tag{4.8}$$

$$\check{Q}_h^t(\cdot) \coloneqq k_h^t(\cdot)^T(K_h^t + \lambda I)^{-1}\underline{Y}_h^t, \quad [\underline{Y}_h^t]_i \coloneqq r_h(s_h^i, a_h^i) + \underline{V}_{h+1}^t(s'_{h,i}). \tag{4.9}$$

Our proposed algorithm AE-LSVI is presented in algorithm 5. At each h, the algorithm uses optimistic and pessimistic value estimates from (4.6) and (4.8) (computed based on the data collected in previous episodes), and selects s_h^t and a_h^t as:

$$s_h^t \in \operatorname{argmax}_{s\in\mathcal{S}}\left[\max_{a\in A}\overline{Q}_h^t(s,a) - \max_{a\in A}\underline{Q}_h^t(s,a)\right], \tag{4.10}$$

$$a_h^t \in \operatorname{argmax}_{a\in\mathcal{A}}\ \overline{Q}_h^t(s_h^t,a). \tag{4.11}$$

The main intuition behind the proposed sampling rules is as follows. Since the optimal policy π^* is the greedy policy with respect to $\{Q_h^*\}_{h\in[H]}$, we do not need to learn $\{Q_h^*\}_{h\in[H]}$ everywhere on $\mathcal{S}\times\mathcal{A}$. Hence, it is sufficient to focus on discovering the best actions for each state. Our active exploration strategy is explicitly designed to focus on (i) states for which the best action is the most uncertain (eq. (4.10)) and (ii) corresponding best "optimistic" actions (eq. (4.11)).

We use $\hat{\pi}_N$ to denote the final reported policy returned by AE-LSVI (see 5). There are various reasonable greedy-based choices for $\hat{\pi}_N$. The simplest one is to return $\hat{\pi}_{N,h}(\cdot) = \operatorname{argmax}_{a\in\mathcal{A}}\hat{Q}_h^N(\cdot,a)$, but in our theory and experiments, we focus on equating $\hat{\pi}_N$ with the policy with the highest lower confidence estimate $\underline{Q}_h^t(s,a)$. Our sampling strategy combined with the proposed policy reporting rule allows for discovering an ϵ-optimal policy uniformly over any given initial state as we formally show in the next section.

4.4 Theoretical Results

We use the structural assumption for the kernel setting from Yang et al. [2020] which states that the Bellman operator maps any bounded value function to a function with a bounded RKHS norm.

Algorithm 5 AE-LSVI (Active Exploration with Least-Squares Value Iteration)

Require: kernel function $k(\cdot,\cdot)$, exploration parameter $\beta > 0$, regularizer $\lambda \geq 1$

for $t = 1, \ldots, N$ **do**
  for $h \in \{1, \ldots, H\}$ **do**
    Set $\overline{Q}^t_{H+1}, \underline{Q}^t_{H+1}$ as the zero functions
    for $h = H, \ldots, 1$ **do**
      Obtain $\overline{Q}^t_h$ and $\underline{Q}^t_h$ from (4.6) and (4.8)
    end for
    Choose $s^t_h \in \operatorname{argmax}_{s \in S} \left[\max_{a \in A} \overline{Q}^t_h(s,a) - \max_{a \in A} \underline{Q}^t_h(s,a) \right]$
    Choose $a^t_h \in \operatorname{argmax}_{a \in \mathcal{A}} \overline{Q}^t_h(s^t_h, a)$
    Observe the reward $r_h(s^t_h, a^t_h)$ and the next state $s'_{h,t} \sim T_h\big(\cdot \,|s^t_h, a^t_h\big)$
  end for
end for
Output the policy estimate $\hat{\pi}_N$ such that $\hat{\pi}_{N,h}(\cdot) = \operatorname{argmax}_{a \in \mathcal{A}} \max_{t \in [N]} \underline{Q}^t_h(s,a)$

Assumption 1. *Let $B_Q > 0$ be a fixed positive constant. Let $k : (\mathcal{S} \times \mathcal{A})^2 \to \mathbb{R}$ be a continuous kernel function on a compact set $\mathcal{S} \times \mathcal{A} \subset \mathbb{R}^d$ such that $\sup_{x,x' \in \mathcal{S} \times \mathcal{A}} k(x,x') \leq 1$. We assume that $\|\mathcal{T}^*_h Q\|_{\mathcal{H}} \leq B_Q H$ for all functions $Q : \mathcal{S} \times \mathcal{A} \to [0,H]$ and all $h \in [H]$, where $\mathcal{T}^*_h$ denotes the Bellman optimality operator, i.e.,*

$$\mathcal{T}^*_h Q(s,a) = r_h(s,a) + \mathbb{E}_{s' \sim \mathbb{P}_h(\cdot|s,a)} \Big[\max_{a' \in \mathcal{A}} Q(s',a') \Big]. \tag{4.12}$$

Assumption 1 implies that for every $h \in [H]$, both $r_h(\cdot,\cdot)$ and $Q^*_h(\cdot,\cdot)$ are elements of the set $\{f \in \mathcal{H} : \|f\|_{\mathcal{H}} \leq B_Q H\}$. Conversely, a sufficient condition for Assumption 1 to be satisfied with $B_Q = 2$ is that $\{r_h(\cdot,\cdot), T_h(s'|\cdot,\cdot)\} \subseteq \{f \in \mathcal{H} : \|f\|_{\mathcal{H}} \leq 1\}$ for all $h \in [H]$ and $s' \in \mathcal{S}$ [Yang et al., 2020]. Moreover, only assuming $Q^*_h \in \mathcal{H}, \|Q^*_h\| \leq B_Q H$ for all $h \in [H]$ is not enough in order to obtain sample size guarantees which are polynomial in H and d [Du et al., 2020].

The main quantity that characterizes the complexity of the RKHS function class in the kernelized setting is the maximum information gain [Srinivas et al., 2009]

$$\Gamma_k(N,\lambda) := \sup_{D \subseteq \mathcal{S} \times \mathcal{A}, |D| \leq N} \tfrac{1}{2} \ln |I + \lambda^{-1} K_{D,D}|, \tag{4.13}$$

where $K_{D,D}$ denotes the Gram matrix, $|\cdot|$ denotes the determinant, $\lambda > 0$ is a regularization parameter, and the index k indicates the kernel. This quantity is known to be sublinear in N for most of the popularly used kernels [Srinivas et al., 2009].

Further, we define the set of possible optimistic and pessimistic value functions

$$\begin{aligned} \mathcal{Q}(N,h,b) = \Big\{ Q(\cdot,\cdot) = \big[\hat{Q}(\cdot,\cdot) \pm \beta \sigma_D(\cdot,\cdot)\big]_0^{H-h+1} \;: \\ \hat{Q} \in \mathcal{H}, \|\hat{Q}\|_{\mathcal{H}} \leq 2H\sqrt{\Gamma_k(N,\lambda)}, \beta \in [0,b], D \subseteq \mathcal{S} \times \mathcal{A}, |D| \leq N \Big\}, \end{aligned} \tag{4.14}$$

where $b > 0$ and $\sigma_D(\cdot,\cdot)$ is of the form (4.5) computed with a data set $D \subseteq \mathcal{S} \times \mathcal{A}$, and denote its ℓ^∞-covering number as $N_\infty(\epsilon, N, h, b)$. [1] Our sample complexity bounds depend on $b_N > 0$

[1] The results on b_N hold despite Yang et al. [2020, Lemma D.1] being stated only in the case of the smaller class obtained from only *adding* $+\beta\sigma_D(\cdot,\cdot)$ in the definition of $Q(\cdot,\cdot)$ in $\mathcal{Q}(N,h,b)$ in eq. (4.14).

defined as the smallest number that satisfies the following inequality:

$$8\Gamma_k\big(N, \tfrac{N+1}{N}\big) + 8\log N_\infty(H/N, N, h, b_N) + 16\log(2NH) + 22 + 2B_Q^2\big(\tfrac{N+1}{N}\big) \leq (b_N/H)^2 \tag{4.15}$$

For many kernel functions, b_N has a sublinear dependence on N. For instance, $b_N = \mathcal{O}(\gamma H\sqrt{\log(\gamma NH)})$ for bounded and continuously differentiable kernels with γ-finite spectrum and $b_N = \mathcal{O}(H\sqrt{NH}\log(N)^{1/\gamma})$ for bounded and continuously differentiable kernels with γ-exponential decay. See [Yang et al., 2020, Corollary 4.4] for more details.

We recall that the Bellman equation implies that $\overline{Q}^t_{h+1}(\cdot)$, $\underline{Q}^t_h(\cdot)$ are upper and lower confidence bounds for Q^*_h for all $h \in [H]$, respectively (see lemma 2), while the target functions of kernel ridge regressions are $\mathcal{T}^*_h\overline{Q}^t_{h+1}(\cdot)$ and $\mathcal{T}^*_h\underline{Q}^t_{h+1}(\cdot)$. As a technical tool, we use the following concentration result that follows from [Yang et al., 2020, Lemma 5.2].

Lemma 1. *Consider the setup of Assumption 1, and $\overline{Q}^t_{h+1}(\cdot)$, $\underline{Q}^t_h(\cdot)$ $\sigma^t_h(\cdot)$ from eqs. (4.5), (4.6) and (4.8) computed with $\lambda = 1 + 1/N$ and $\beta = b_N$ from eq. (4.15). Then with probability at least $1 - (2N^2H^2)^{-1}$, the following holds for all $t \in [N]$, $h \in [H]$ and all $(s, a) \in \mathcal{S} \times \mathcal{A}$:*

$$0 \leq \overline{Q}^t_h(s, a) - \mathcal{T}^*_h\overline{Q}^t_{h+1}(s, a) \leq 2\beta\sigma^t_h(s, a), \tag{4.16}$$

$$0 \leq \mathcal{T}^*_h\underline{Q}^t_{h+1}(s, a) - \underline{Q}^t_h(s, a) \leq 2\beta\sigma^t_h(s, a). \tag{4.17}$$

With the previous confidence lemma in place, we state our main theorem that characterizes the sample complexity of AE-LSVI . The proof is given in appendix C.1.2.

Theorem 3. *Consider the setting of Lemma 1 and let $H \in \mathbb{N}$ be a fixed horizon. When running algorithm 5 for N episodes, then with probability at least $1 - (2N^2H^2)^{-1}$, the best-policy estimate $\hat{\pi}_N$ (Algorithm 5, Line 12) satisfies:*

$$\|V^*_1 - V^{\hat{\pi}_N}_1\|_{\ell^\infty(\mathcal{S})} \leq 2\sqrt{3}\beta H(H+1)\sqrt{\tfrac{\Gamma_k(N,\lambda)}{N}}. \tag{4.18}$$

In other words, for a given fixed precision $\epsilon > 0$, after $N = O\Big(\frac{\beta^2H^4\Gamma_k(N,\lambda)}{\epsilon^2}\Big)$ episodes (or $O\Big(\frac{\beta^2H^5\Gamma_k(N,\lambda)}{\epsilon^2}\Big)$ samples) $\|V^*_1 - V^{\hat{\pi}_N}_1\|_{\ell^\infty(\mathcal{S})} \leq \epsilon$ holds with probability at least $1 - (2N^2H^2)^{-1}$.

The obtained result is general since it holds for any kernel function that satisfies Assumption 1. To obtain concrete kernel-dependent regret bounds it remains to specify the kernel and the bounds for the corresponding maximum information gain in eq. (4.15). These are summarized in Yang et al. [2020] for the most widely used kernels (see Assumption 4.3 and its discussion).

In the special case of linear kernels with the feature dimension d, our sample complexity guarantee reduces to $\tilde{O}(\frac{d^3H^7}{\epsilon^2})$. Better bounds (in terms of d) for this special case are known $\tilde{O}(\frac{d^2H^7}{\epsilon^2})$, see, e.g., Agarwal et al. [2019, Theorem 3.3]. These bounds are obtained by the LSVI algorithm with D-optimal design. Unlike this algorithm, AE-LSVI uses optimism for active exploration and such a performance gap is present even in the simpler linear bandit setting where optimistic algorithms are known to attain worse sample complexity guarantees [Lattimore and Szepesvári, 2020, Chapter 22]. The special case also includes the linear MDP setting, which assumes linear reward functions and linear transition kernels. For linear MDPs it is possible to find a policy π satisfying $V_1(s_1) - V^\pi_1(s_1) \leq \epsilon$ using $\tilde{O}(d^2H^3/\epsilon^2)$ samples [Hu et al., 2022]; in our setting of Assumption 1, such a policy π can be found using $O(H^5\beta^2\Gamma_k(N, \lambda)/\epsilon^2)$ samples [Yang et al., 2020]. Both results hold with at least a constant probability. However, they require that the initial state s_1 is fixed for all episodes. In contrast, the result of theorem 3 holds uniformly over the entire state space.

4.5 Application to Offline Contextual Bayesian Optimization

In this section, we specialize algorithm 5 to the offline contextual Bayesian optimization setting [Char et al., 2019]. We show that in this setting the proposed active exploration scheme leads to new sample complexity bounds that hold *uniformly* over the context space.

The offline contextual Bayesian optimization setting is similar to the one considered in Section 4.2 when $H = 1$. In particular, instead of having H different functions to learn, we have a single unknown objective $Q^* : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ that we learn about (from noisy point evaluations). Here, we refer to $\mathcal{S}$ as the context space, and assume that both $\mathcal{S}$ and $\mathcal{A}$ are compact sets. As before, we use a shorthand notation $\mathcal{X} = \mathcal{S} \times \mathcal{A}$. In each round $t \in [T]$, the learner chooses a context-action pair $(s^t, a^t) \in \mathcal{S} \times \mathcal{A}$ and observes $y_t = Q^*(s^t, a^t) + \eta_t$ (with independent sub-Gaussian noise). To choose (s^t, a^t) at each round t, we make use of the same active exploration strategy from eqs. (4.10) and (4.11). Our complete algorithm for the offline BO setting can be found in Appendix C.1.3 (see algorithm 8).

We define $\hat{Q}^t : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ (and $\sigma^t : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$) similarly as $\hat{Q}^t_h$ (resp. σ^t_h) from eq. (4.4) (resp. eq. (4.5)) but with the modification of ignoring the index h and defining $Y_t := (y_i)_{i=1}^{t-1} \in \mathbb{R}^{t-1}$. We further define the upper and lower confidence bounds for Q^* as:

$$\overline{Q}^t(\cdot,\cdot) = \hat{Q}^t(\cdot,\cdot) + \beta_t \sigma^t(\cdot,\cdot), \quad \underline{Q}^t(\cdot,\cdot) = \hat{Q}^t(\cdot,\cdot) - \beta_t \sigma^t(\cdot,\cdot). \tag{4.19}$$

When $Q^* \in \mathcal{H}$ and $\|Q^*\|_{\mathcal{H}} \leq B$ correspond to some known kernel (such that $k(x, x') \leq 1$ for all $x, x' \in \mathcal{X}$), then $(\beta_t)_{t\in[T]}$ is a non-decreasing sequence of parameters that can be chosen according to Abbasi-Yadkori [2012, Theorem 3.11] to yield valid confidence bounds. Similarly, in case of $Q^* \sim \mathrm{GP}_{\mathcal{X}}(0, k)$ (Bayesian setting), we can utilize Gaussian Process confidence bounds [Srinivas et al., 2009] and use the corresponding $(\beta_t)_{t\in[T]}$ sequence. In what follows, we assume that $(\beta_t(\delta))_{t\in[T]}$ is a non-decreasing sequence such that with probability at least $1 - \delta$,

$$\underline{Q}^t(s,a) \leq Q^*(s,a) \leq \overline{Q}^t(s,a) \tag{4.20}$$

holds for all $t \in [N]$ and $(s, a) \in \mathcal{S} \times \mathcal{A}$.

Corollary 1. *Assume $(\beta_t(\delta))_{t\in[T]}$ is set to satisfy eq. (4.20). Fix $\epsilon \in (0, 1)$ and run Algorithm 8 for*

$$N \geq \frac{12\beta_N^2 \Gamma_k(N, \lambda)}{\epsilon^2} \tag{4.21}$$

rounds. Then, for every $s \in \mathcal{S}$, the reported policy $\hat{\pi}_N(\cdot)$ computed as in Line 6 (Algorithm 8) satisfies $Q^*(s, \hat{\pi}_N(s)) \geq \max_{a\in\mathcal{A}} Q^*(s, a) - \epsilon$ with probability at least $1 - \delta$.

We briefly compare the result obtained in corollary 1 with related results from the literature. In the Bayesian setting, Char et al. [2019, Theorem 1] obtain a sample complexity that scales as $\mathbb{E}[N] = O\big(|\mathcal{S}|^3|\mathcal{A}|\Gamma_k(N, \lambda)/\epsilon^2\big)$ in expectation for a given context distribution. In comparison, our result obtained in eq. (4.21) holds in ℓ^∞-norm over the context space (i.e., implies bounds for *any* context distribution). When specialized to the finite set $\mathcal{X} = \mathcal{S} \times \mathcal{A}$ and when $f \sim \mathrm{GP}_{\mathcal{X}}(0, k)$, the result of corollary 1 holds with $\beta_N = O(\log(|\mathcal{X}|N^2))$ [Srinivas et al., 2009], which then results in $N = O\big(\frac{\log^2(|\mathcal{X}|N^2)\Gamma_k(N,\lambda)}{\epsilon^2}\big)$ leading to a significant improvement for large discrete context spaces. In the setting of distributionally robust Bayesian optimization (DRBO), Kirschner et al. [2020] obtain a result with the same dependency as ours. However, their bound holds only for a *fixed* contextual distribution and degenerates as a function of the distance between the training and test distributions.

4.6 Experiments

4.6.1 Reinforcement Learning Experiments

In the previous sections we presented the AE-LSVI algorithm, which provably identifies a near-optimal policy in polynomial time given access to a generative model of the MDP dynamics. Here, we test the AE-LSVI algorithm empirically, and additionally provide one of the first empirical evaluation of the LSVI-UCB method from Yang et al. [2020] on standard benchmarks. We evaluate AE-LSVI and LSVI-UCB on four MDPs from the literature as well as four synthetic contextual BO problems from Char et al. [2019]. We discuss details of our implementation in Appendix C.2.1.

Each environment has a discrete action space. For continuous environments, we discretize the action space into 10 bins per dimension but model the value function in the original continuous state and action space. All methods besides DDQN are initialized by executing a random policy for two episodes. In between exploration episodes, the pessimistic policy $\hat{\pi}_N$ is evaluated by executing it for 10 episodes in the environment.

Initial State Distribution To evaluate the policies found by each method, we must initialize the policy at initial states drawn from some distribution p_0 at test time. As AE-LSVI does not explicitly consider the initial state distribution, for each environment we choose both a standard p_0 from the literature as well as a an alternate distribution p_0' that is translated in the state space, i.e., $p_0'(s) = p_0(s - \Delta_s)$ for some Δ_s. The alternate distribution allows us evaluate the best policy estimate in an area of state space that is not explicitly given to agents. We evaluate each policy using initial states sampled from p_0' as a proxy for understanding how well the optimal policy has been identified in regions of the state space beyond where it was initialized. We give a complete description of the various p_0' for each environment in Appendix C.2.2. In Table 4.1, we present results for each method and environment when initialized on p_0, which is the typical setup for training and evaluating RL algorithms in the literature. In Table 4.2 we present results for each method evaluated for the initial state distribution p_0'.

Comparison Methods Besides AE-LSVI and LSVI-UCB , we compare against several ablations and methods taken from the literature. As a naive baseline for performance in active exploration, we randomly sample state-action pairs from the MDP, evaluate the next states and rewards, and fit Q-functions to that data as in the other methods, executing the policy given by the Q-function mean (**Random**). We also perform uncertainty sampling (**US**) on the Q-function, choosing state-action pairs at each step that maximize $\sigma_h^t(\cdot, \cdot)$ as in eq. (4.5). Additionally, we compare against three online RL baselines: the Double DQN algorithm [Van Hasselt et al., 2016] where an epsilon-greedy approach is used for exploration (**DDQN**), the bootstrapped DQN [Osband et al., 2016b] which keeps an ensemble of Q-functions and does exploration acting according to a sampled Q-function each exploratory rollout (**BDQN**), and a greedy exploration algorithm (**Greedy**) that chooses $\operatorname{argmax}_a \hat{Q}_h^t(s, a)$ at every step h for a given state s but uses the same value iteration procedure used in the main methods. The experiments are conducted with a default exploration bonus $\beta = 0.5$, however, we also empirically analyze the performance for other β-values in Appendix C.2.3.

Environments We evaluate all methods on four environments: a **Cartpole** swing-up problem with dense rewards, a nonlinear **Navigation** problem, and two problems (β **Tracking** and β **+ Rotation**) in plasma control from Mehta et al. [2022a], in which plasma is driven to a desired target state. We give further information on the environments used in Appendix C.2.2.

Results As our bound on the value function error uses the $\ell^\infty(\S)$-norm, our method provably finds an approximately optimal policy regardless of the initial distribution. The LSVI-UCB method is able to quickly learn a policy for the initial state distribution p_0 given at training time, as it is designed to minimize regret on the episodic MDP initialized at p_0. This can be seen clearly in

Environment	AE-LSVI	Random	US	LSVI-UCB	DDQN	BDQN	Greedy
Cartpole	15.2 ± 0.5	13.6 ± 0.5	13.6 ± 0.6	17.1 ± 0.7	$\mathbf{19.3 \pm 0.7}$	$\mathbf{19.0 \pm 0.8}$	17.2 ± 0.4
Navigation	6.0 ± 1.7	6.7 ± 1.4	8.9 ± 0.7	$\mathbf{12.9 \pm 0.2}$	7.3 ± 1.5	7.2 ± 0.9	10.9 ± 1.5
β Tracking	12.7 ± 0.3	11.6 ± 0.4	11.7 ± 0.2	$\mathbf{13.8 \pm 0.1}$	13.4 ± 0.2	$\mathbf{13.9 \pm 0.1}$	12.9 ± 0.3
β + Rotation	15.2 ± 0.6	15.2 ± 0.6	15.1 ± 0.4	$\mathbf{17.8 \pm 0.1}$	15.1 ± 0.4	14.2 ± 0.8	$\mathbf{17.9 \pm 0.1}$

Table 4.1: Average Return $\pm$ standard error of executing the identified best policy on the MDP starting from p_0 over 5 seeds after collecting 1000 timesteps of data through the use of a generative model (left of line) or episodes starting from p_0 (right of line).

Environment	AE-LSVI	Random	US	LSVI-UCB	DDQN	BDQN	Greedy
Cartpole	$\mathbf{16.8 \pm 0.4}$	12.9 ± 0.4	14.5 ± 0.3	12.9 ± 0.3 (14.2 ± 0.6)	15.3 ± 0.6 (13.7 ± 1.3)	16.1 ± 0.5 (13.0 ± 1.2)	13.3 ± 0.5 $(\mathbf{16.7 \pm 0.2})$
Navigation	$\mathbf{22.3 \pm 0.4}$	15.3 ± 0.8	17.5 ± 1.3	13.6 ± 0.6 (20.6 ± 1.1)	17.1 ± 2.4 (18.1 ± 2.6)	21.4 ± 1.2 (18.4 ± 2.1)	15.2 ± 1.6 (14.0 ± 0.8)
β Tracking	$\mathbf{14.0 \pm 0.4}$	9.2 ± 0.9	12.5 ± 0.1	13.3 ± 0.3 $(\mathbf{13.7 \pm 0.2})$	$\mathbf{13.8 \pm 0.1}$ $(\mathbf{13.7 \pm 0.2})$	$\mathbf{14.0 \pm 0.1}$ $(\mathbf{13.7 \pm 0.1})$	12.5 ± 0.4 $(\mathbf{13.8 \pm 0.1})$
β + Rotation	$\mathbf{14.3 \pm 0.2}$	12.8 ± 1.4	13.3 ± 0.5	10.1 ± 0.4 (12.7 ± 0.3)	12.9 ± 1.1 (13.4 ± 0.3)	13.7 ± 0.8 (12.7 ± 1.2)	12.8 ± 0.7 (7.5 ± 0.2)

Table 4.2: Average Return $\pm$ standard error of executing the identified best policy on the MDP starting from p_0' over 5 seeds after collecting 1000 timesteps of data through the use of a generative model (left) and online RL methods (right). For online methods, numbers without parentheses refer to training from episodes starting from p_0, whereas numbers in parentheses use the uniform distribution on the state space as initial states during training.

Table 4.1, which shows that after 1000 samples, LSVI-UCB performs the best on nearly every environment. In the online setting when the start state distribution is known, greedy and ϵ-greedy methods like DDQN also perform relatively well. We also see in Table 4.1 that AE-LSVI does not perform particularly well compared to the online methods given the 1,000-sample budget. This is to be expected, as the online methods naturally collect data that is reachable from p_0 and in particular LSVI-UCB is designed to minimize regret on episodes beginning from p_0. However, this focus on performing well when starting from p_0 comes at the expense of active exploration and identifying the best policy uniformly across the state space.

As shown in Table 4.2, AE-LSVI outperforms the baselines when evaluated on a *different* initial state distribution p_0', even when the online algorithms are initialized from a uniform initial state distribution p_0 during training. This is unsurprising, as AE-LSVI is precisely built for this setting and identifies the best action uniformly across the state space, unlike LSVI-UCB which aims to minimize regret starting from an initial state distribution. We see that uncertainty sampling outperforms a random data selection strategy and is comparable to the online methods. However, as we discuss above (in section 4.3), in general it is the uncertainty in the value of the best action at a state and not the uncertainty in the value of a state-action pair that needs to be reduced in order to more efficiently find the best policy. We see that, in general, the online methods perform better on p_0' when they train on episodes uniformly initialized on the state space. This suggests that in these cases, it is helpful to make sure that the evaluation distribution p_0' is supported by the training distribution p_0. We also note that (as we describe in Appendix C.2.2) the maximum possible score on **Navigation** starting from p_0' is higher than that from p_0 due to a starting distribution closer to the goal. We believe that these results give empirical support to the theoretical claims of Section 4.4.

4.6.2 Offline Contextual Bayesian Optimization Experiments

We test the performance of AE-LSVI (Algorithm 8 in Appendix C.1.3) in the offline contextual Bayesian optimization setting. In particular, we test the algorithm on the optimization problems

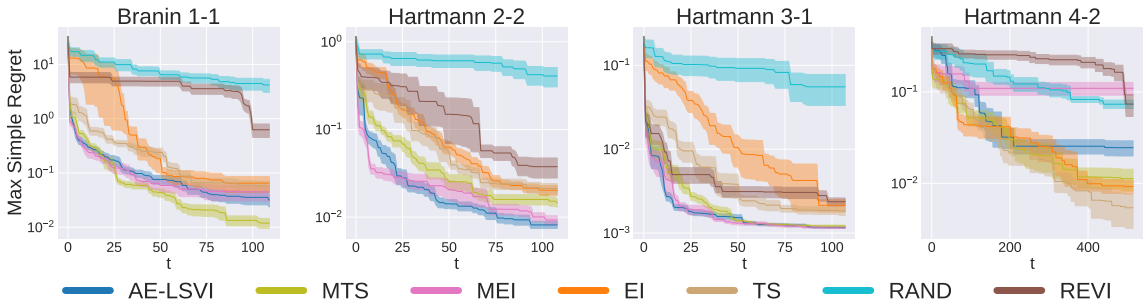

Figure 4.1: The maximum simple regret seen in any given context for the offline contextual Bayesian optimization experiments. The shaded regions show the standard error over 10 different seeds.

presented in Section 3 of Char et al. [2019], each having a discrete context space but continuous action space. In all experiments, we average over 10 seeds. At the beginning of each experiment, the values corresponding to five actions, chosen uniformly at random, are observed for each context. Every time new data is observed, the hyperparameters of the GP are tuned according to the marginal likelihood. We leverage the Dragonfly library for these experiments [Kandasamy et al., 2020].

Comparison Methods For baselines, we compare against the Multi-task Thompson Sampling (**MTS**) method presented by Char et al. [2019], which picks context and action based on the largest improvement over what has been seen according to samples from the posterior. In addition, we compare to the strategy of picking the context with the greatest expected improvement. This method was presented by Swersky et al. [2013], and we refer to it as Multi-task Expected Improvement (**MEI**), following Char et al. [2019]. We also compare against the **REVI** algorithm [Pearce and Branke, 2018], which picks contexts and actions that will increase the posterior mean the most across all contexts. Additionally, we show the performance of naive Thompson sampling (**TS**) and expected improvement (**EI**), where contexts are picked in a round robin fashion. Lastly, we show the performance of randomly selecting contexts and actions at each time step (**RAND**).

Experiment Tasks To evaluate the method in the case where the objective function is correlated in context space, we take a higher dimensional function and assign some dimensions to context space and the rest to action space. A single GP with a squared exponential kernel is then used to model the objective function. In particular, the Branin-Hoo [Branin, 1972], Hartmann 4, and Hartmann 6 [Picheny et al., 2013] functions are used to create Branin 1-1, Hartmann 2-2, Hartmann 3-1, and Hartmann 4-2, where the first number corresponds to the context dimension and the second to the action dimension. These functions have 10, 9, 8, and 16 equispaced contexts, respectively.

Results Figure 4.1 shows the maximum simple regret seen in any given context as a function of t values observed. As seen from these plots, AE-LSVI often is one of the best performing methods. The only task that AE-LSVI struggles on is Hartmann 4-2. We believe that estimating the amount of improvement to be gained at each context is difficult for this benchmark task. This is supported by the fact none of the more sophisticated methods outperforms the baseline that applies **EI** in a round-robin fashion. It is likely that improved modeling or hyperparameter selection is needed in order for these methods to achieve the highest performance on this task.

5 | Efficiently learning policies from comparative feedback by choosing optimal data

5.1 Introduction

The alignment of foundation models with user preferences has gained unprecedented importance due to the widespread utilization of large language models (LLMs). The established pipeline for alignment in LLMs, as outlined in Stiennon et al. [2020] and Ouyang et al. [2022], comprises two essential steps given a pretrained LLM. First, in the Supervised Fine-Tuning (SFT) phase, the LLM undergoes fine-tuning via supervised learning with examples demonstrating the desired behavior. In the second step, Reinforcement Learning from Human Feedback (RLHF), a policy generates multiple completions for each conversation prefix (prompt) in a training set; users then give ordinal preferences amongst the set of completions for a particular prompt. These preferences are used to train a 'reward model' via a ranking loss like the Bradley-Terry-Luce (BTL) model [Bradley and Terry, 1952]. Finally, the policy is trained, typically via Proximal Policy Optimization (PPO) [Schulman et al., 2017], to optimize the reward model while not moving too far from the SFT-trained policy.

As these models continue to scale and their areas of application broaden, the number of roles for which we need to align them increases as does the overall scale of human-generated training data requirements. Data annotation for preference-based learning is already a substantial cost for companies that train LLMs. This cost is likely to grow alongside the industry. This is especially acute for LLMs in specialized areas, where the cost of expert feedback can be substantially higher.

In this work, we take advantage of the fact that we control which prompts and completions we provide to human labelers in order to make efficient use of their efforts. Drawing on recent advancements in active exploration for reinforcement learning [Li et al., 2023] and in black-box optimization [Xu et al., 2020], we introduce a method for assessing the value of collecting preferences on specific datapoints that is both prospective and task-focused. First, we formalize this setting as a *dueling contextual bandit problem* and design an efficient algorithm that offers polynomial worst-case sample complexity guarantees regarding the policy's performance. Next, we extend these ideas to a more real-world setting: choosing datapoints for the training of LLM assistants. Here, we build atop recent work [Rafailov et al., 2023], which allows us to apply active data selection to an RLHF process using a supervised objective and single model. We evaluate the method on three datasets: the Stanford Human Preferences dataset [Ethayarajh et al., 2022], the Anthropic Helpful-Harmless dataset [Bai et al., 2022], and a third dataset (which we contribute to the literature) that extends an existing dataset of Jeopardy! questions and answers to evaluate the ability of an alignment method to avoid hallucinations. We find that our algorithm can boost performance

by over 10% on the preference datasets when performing RLHF with a modest human-feedback sample budget, and that our method is best at avoiding hallucinations on our Jeopardy! dataset.

5.2 Related Work

Learning from Comparative Feedback Reinforcement learning from comparative human feedback has been studied for more than a decade, including work by Fürnkranz et al. [2012], Akour [2014] and, notably, Christiano et al. [2017], which enabled sample-efficient collection of human feedback for deep reinforcement learning (RL) by training a reward model as the RL target. In the Atari test case, where naive deep RL would have necessitated thousands of hours of gameplay, they accomplished superior performance with just 5,500 or several hours of human queries.

Many recent approaches have demonstrated the effectiveness of using human feedback to enhance stylistic continuation [Ziegler et al., 2019], text summarization [Stiennon et al., 2020], translation [Kreutzer et al., 2018], semantic parsing [Lawrence and Riezler, 2018], review generation [Cho et al., 2018], and evidence extraction [Perez et al., 2019]. In particular, the work by Bai et al. [2022] places focus on improving model reliability and robustness by incorporating human feedback to gauge the helpfulness or harmfulness of its responses. However, while effective, the integration of human feedback comes with substantial costs. For example, Stiennon et al. [2020] achieved substantial enhancements over baseline methods but required the generation of summaries for 123,169 posts from the TL;DR dataset, a task performed by a large team of labelers from crowdsourcing platforms. This heavy-resource requirement is reflected in state-of-the-art work. Ouyang et al. [2022] emphasizes RLHF to improve alignment of the GPT-3 model across aspects such as toxicity, hallucinations, moral opinion, and overall quality. Here, the team enlisted the efforts of 40 labelers and worked with a dataset comprising over 100,000 examples labeled by humans.

Dueling Bandits The bandit literature has also explored the effectiveness of comparative feedback—for example, in the "dueling bandit" setting—while considering the cost of acquiring such information. This was first studied by Yue et al. [2012] in settings where comparative information is relatively easy to extract but absolute rewards (*i.e.*, direct queries) are ill-defined and have no absolute scale. Later, Bengs et al. [2021] surveyed methods used in the online learning setting, where the trade off with cost of information is most acute, including those used in the online contextual dueling bandit setting by Dudík et al. [2015]. These constraints motivate a kernelized approach that can incorporate the nonlinearities in the models used in practice.

Active Contextual Bandit Optimization When there exist distinct phases of learning and then deployment, an agent can often take steps for improved sample efficiency. For example, in a contextual bandit setting, Char et al. [2019] consider the problem where at test time the goal is to perform well on average across a context distribution, while during the learning phase the goal is to choose both contexts and actions for best performance at test-time. The authors proposed a multi-task version of Thompson sampling during the training phase, which yields provable regret bounds. We extend this setting from cardinal to ordinal rewards as is appropriate for comparative feedback.

In Li et al. [2023], the agent queries contexts where the value function is most uncertain and acts optimistically. Combined with least-squares value iteration, this method leads to provable polynomial-sample convergence in the worst-case error of the value function estimate in reinforcement learning in general, and as a corollary the setting from Char et al. [2019] as a special case. This sets the foundation that we will adapt to the comparative feedback setting.

In the realm of ,active contextual bandits that make use of kernels, previous research has explored various aspects, including robust objectives [Bogunovic et al., 2018], distributional robustness [Kirschner et al., 2020, Ramesh et al., 2023], multi-agent learning and mixed strategies [Sessa et al., 2019, 2020]. However, to our knowledge, none of the methods proposed in these prior studies can be directly employed in our specific dueling setting.

We also include related work on uncertainty estimation in large language models in Sec. C.8.

5.3 Problem Setting

In this paper, we consider a dueling variant of what we denote the active contextual bandit problem introduced in Char et al. [2019] that we refer to as ACDB for short. An instance of this problem is defined by a tuple $(\mathcal{X}, \mathcal{A}, f)$ where $\mathcal{X}$ denotes the context space, $\mathcal{A}$ denotes the action space and $f : \mathcal{X} \times \mathcal{A} \times \mathcal{A} \to [0, 1]$ is a preference function so that $f(x, a, a')$ denotes the probability that the action a is preferred to the action a' when the underlying context is x. We also define a domain $\mathcal{D} = \mathcal{X} \times \mathcal{A}$. We will design algorithms that operate under the following interaction protocol, which occurs for T time steps. During each time step $t \in [T]$, the agent selects a context $x_t \in \mathcal{X}$ and a pair of actions $a_t, a'_t \in \mathcal{A}$ and observes a binary random variable $w_t \sim \mathrm{Bern}(f(x_t, a_t, a'_t))$ which equals one if a_t is preferred to a'_t and zero otherwise.

We assume that the preference function takes the following form,

$$f(x, a, a') = \rho\left(r(x, a) - r(x, a')\right), \tag{5.1}$$

where $\rho : \mathbb{R} \to [0, 1]$ is the *link function* and $r : \mathcal{D} \to \mathbb{R}$ is the *unknown* reward function. Common link functions include the logistic function, which leads to the Bradley-Terry-Luce (BTL) model [Bradley and Terry, 1952] as well as the Gaussian CDF [Thurstone, 1927]. We also place some additional assumptions on the reward function for our theoretical analysis in the kernelized setting.

Our objective within this protocol is to design algorithms that are able to quickly identify policies with a small suboptimality gap. We define the suboptimality gap of a learner's policy $\pi : \mathcal{X} \to \mathcal{A}$ as

$$\mathrm{SubOpt}(\pi) = \sup_{x \in \mathcal{X}} \left(\sup_{a \in \mathcal{A}} r(x, a) - r(x, \pi(x)) \right). \tag{5.2}$$

We remark that this notion of suboptimality (considered in Char et al. [2019] and Li et al. [2023]) is stronger than usual notions that look at the expected suboptimality of the final policy when the contexts are sampled from some known distribution. In contrast, the form of suboptimality we consider here looks at the worst-case context for each policy. For the kernelized and LLM settings we address below, we will make explicit the instantiation of this problem setting.

5.4 Active Exploration in the Kernelized Setting

In this section, we describe our first contribution—a theoretically principled algorithm for the ACDB problem—and provide formal guarantees on its performance. In order to provide such guarantees, we must first instantiate our general problem setup by making assumptions on the preference function f (from (5.1)). In particular, we need to specify a class of functions that contain the true unknown reward function. This choice is subtle as we need to balance the trade-off between the expressiveness of our function class with theoretical tractability. Motivated by its theoretical popularity and empirical success, we choose this function class to be a Reproducing Kernel Hilbert Space. While this choice of function class is common in the literature, we take a slight departure from the standard assumptions in order to more appropriately accommodate our problem setting.

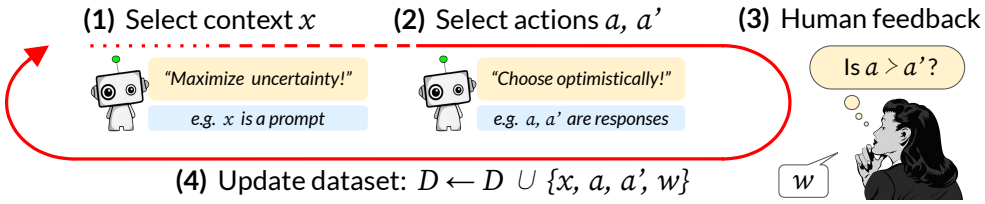

Figure 5.1: Illustration of the active contextual dueling bandit setting, and its application to sample-efficient RLHF in large language models.

The Contextual Borda Function Before going over our assumptions, we first introduce the *contextual Borda function* f_r, which is core to our algorithm. The contextual Borda function generalizes the Borda function introduced in Xu et al. [2020] for dueling-choice optimization which is defined as the probability that a particular action a will be preferred over a random action a' uniformly sampled from the action space. We generalize this definition to the contextual setting as follows, given as $f_r : \mathcal{D} \to [0, 1]$ where $f_r(x, a) = \mathbb{E}_{a' \sim U(\mathcal{A})}[f(x, a, a')]$, where $U(\mathcal{A})$ is the uniform measure over the action space. It is clear from the definition that f_r and r will have the same maximizers.

We now discuss the assumptions we make. Our first assumption restricts the reward and contextual Borda functions to be 'smooth' in an underlying Reproducing Kernel Hilbert Space (RKHS).

Assumption 2. *Let $\kappa : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ denote a positive semi-definite kernel and let $\mathcal{H}_\kappa$ denote its associated RKHS. We assume that $\|r\|_\kappa, \|f_r\|_\kappa \leq B$, where B is a known constant.*

Note that this assumption is stronger than the standard assumption, which only requires that r has a bounded RKHS norm. It is difficult to bound the norm of f_r given a bound on the norm of r due to the generality of our setting, which allows for different link functions. We investigate this issue numerically in Appendix C.5 where we find that the norm of the Borda function is almost always smaller than the norm of the reward function for samples drawn from the distribution of basis functions used for experiments in Section 5.4.3.

Our second assumption relates the reward function to the contextual Borda function.

Assumption 3. *Let $f_r^*(x) = \max_a f_r(x, a)$ and $r^*(x) = \max_a r(x, a)$. There exists a constant L_1 such that for every $x \in \mathcal{X}$, $a \in \mathcal{A}$ we have $\frac{1}{L_1}(r^*(x) - r(x, a)) \leq f_r^*(x) - f_r(x, a)$.*

This assumption implies that differences in r will cause a similar magnitude of difference in f_r. In fact, when the link function $\rho(\cdot)$ is Lipschitz continuous, it is sufficient for its Lipschitz constant to be at least $1/L_1$ for this condition to hold. We note that this assumption holds for the two most commonly used link functions, the logistic function [Bradley and Terry, 1952] and the Gaussian CDF [Thurstone, 1927].

5.4.1 Methods

At a high level, our approach reduces the dueling feedback problem to contextual optimization over a single action via the *contextual Borda function* introduced above. Once reduced appropriately, we apply techniques adapted from recent work on active exploration in reinforcement learning to construct a sampling rule and policy selection rule which allows us to output a policy with provably low sub-optimality. Broadly, our sampling rule samples contexts at which there is maximum uncertainty over the Borda 'value function' and then compares the optimistic action with an action sampled uniformly from the action set.

Estimating the Contextual Borda Function By design, we can estimate the contextual Borda function using preference data $\{x_t, a_t, a'_t, w_t\}$ by selecting x_t, a_t in an arbitrary fashion and sampling a'_t uniformly at random. For low dimensional settings, our algorithm first estimates the contextual Borda function using standard kernelized ridge regression (KRR) [Rasmussen and Williams, 2008]—we refer the reader to Appendix C.3 for an explicit description of this regression procedure. In Section 5.5, we explore modifications of our methods for higher-dimensional settings, such as in the case of LLMs. One key feature of KRR is that it provides both an estimate of the contextual Borda function after t observations, $\mu_t(x, a)$, as well as uncertainty quantification of the predictions. Indeed, under Assumptions 2 and 3 we can show that $|f_r(x, a) - \mu_t(x, a)| \leq \beta\sigma_t(x, a)$ for an appropriately chosen β and $\sigma_t(x, a)$ (see Lemma 5).

Selecting Contexts and Actions Our sampling rule builds on top of the one established in Li et al. [2023]—put simply, the rule is to sample the state with the maximum uncertainty over the value function and then act optimistically. We now present our algorithm which shows how to extend these ideas to the dueling setting via the contextual Borda function f_r.

For now, we assume that there is a known bonus term $\beta_t^{(r)}$ for all t. We can then define upper and lower confidence bounds $\overline{f_r^t}(x, a) = \mu_t(x, a) + \beta_t^{(r)}\sigma_t(x, a)$ and $\underline{f_r^t}(x, a) = \mu_t(x, a) - \beta_t^{(r)}\sigma_t(x, a)$. Our rule is to select a context

$$x_t \in \operatorname{argmax}_{x\in\mathcal{X}} \left(\max_{a\in\mathcal{A}} \overline{f_r^t}(x, a) - \max_{a\in\mathcal{A}} \underline{f_r^t}(x, a) \right). \tag{5.3}$$

Here, we are choosing a context that maximizes the difference between the optimistic 'value function' and the pessimistic 'value function' (both of which require a maximization over actions to compute). We then optimistically choose an action

$$a_t \in \operatorname{argmax}_{a\in\mathcal{A}} \overline{f_r^t}(x_t, a). \tag{5.4}$$

After repeating this process T times, we output a pessimistic policy against the tightest lower bound we can find, which is the maximizer of all our lower bounds through the optimization process. Put formally, we return $\hat{\pi}_N : \mathcal{X} \to \mathcal{A}$ such that

$$\hat{\pi}_N(x) \in \operatorname{argmax}_{a\in\mathcal{A}} \max_{t\leq T} \underline{f_r^t}(x, a). \tag{5.5}$$

From these pieces we construct the full active exploration algorithm, AE-Borda, which we present in Algorithm 6.

5.4.2 Analysis

Before proceeding with our algorithm's formal guarantees, we first introduce the *maximal-information gain* which plays an important role in our results. The maximum information gain over t rounds, denoted Φ_t, is defined as

$$\Phi_t = \max_{A\subset\mathcal{X}\times\mathcal{A}:|A|=t} I(r_A + \epsilon_A; r_A), \tag{5.6}$$

where $r_A = [r(x)]_{x\in A}$, $\epsilon_A \sim N(0, \eta^2 I)$, and $I(X; Y) = H(X) - H(X|Y)$ is the mutual information. With this definition, we are now ready to state our result.

Theorem 4. *Suppose we run Algorithm 6 with*

$$\beta_t^{(r)} = 2B + \sqrt{2\Phi_t + 1 + \log\left(\frac{2}{\delta}\right)}, \tag{5.7}$$

Algorithm 6 AE-Borda

Input: kernel function $\kappa(\cdot,\cdot)$, exploration parameters $\beta_t^{(r)}$, number of initial data n_0
Let $D_{n_0} = \{x_i, a_i, a_i', w_i\}_{i=1}^{n_0}$ for x_i, a_i, a_i' drawn uniformly at random.
for $t = n_0 + 1, \ldots, T$ **do**
    Compute $\mu_t(\cdot,\cdot)$, $\sigma_t(\cdot,\cdot)$ using KRR.
    Choose x_t according to (5.3).
    Choose a_t according to (5.4), draw $a_t' \sim U(\mathcal{A})$, and draw $w_t \sim \text{Bern}(f(x_t, a_t, a_t'))$.
    Let $D_t = D_{t-1} \cup \{(x_t, a_t, a_t', w_t)\}$.
end for
Output a final policy $\hat{\pi}_N$ according to (5.5).

then, with probability at least $1-\delta$*, we have that*

$$\text{SubOpt}(\hat{\pi}_T) \leq O\left(\frac{L_1}{\sqrt{T}}\left(B + \Phi_T\sqrt{\log\frac{1}{\delta}}\right)\right). \tag{5.8}$$

Proof Sketch. At a high-level, the proof of this result is as follows. First, we use standard results on KRR to show that our choice of $\beta^{(r)}$ guarantees that our confidence bands contain $f^r(x,a)$ with high probability simultaneously for all t and $x, a \in \mathcal{X} \times \mathcal{A}$. Next, we use Assumption 3 to show that the suboptimality of the pessimistic policy induced by our estimated contextual Borda function is small whenever we are able to estimate the contextual Borda function well. Finally, we conclude the proof by showing that our sampling rule indeed allows us to estimate the contextual Borda function well. The full proof can be found in Appendix 4.

Concrete Performance Bounds. To more concretely understand the performance of our algorithm, we instantiate our results for two commonly studied kernels: the linear and squared-exponential. For both of these settings, the scaling of the information gain is well known (see for example Srinivas et al. [2009]). In the linear setting, we have that $\Phi_T = O(d\log T)$ leading to a bound of $O\left(\frac{L_1}{\sqrt{T}}\left(d\log T\right)\right)$. For squared exponential kernels we have $\Phi_T = O\left(\log(T)^{d+1}\right)$ leading to a suboptimality bound of $O\left(\frac{L_1}{\sqrt{T}}\left(\log(T)^{d+1}\right)\right)$.

When compared to existing results for dueling bandits [Xu et al., 2020] as well as standard bandits [Chowdhury and Gopalan, 2017], we see that our suboptimality bounds match, thus showing that our algorithm is able to achieve the same performance under a stronger performance metric.

5.4.3 Experiments in the Kernelized setting

In order to assess the validity of our theory we have conducted synthetic experiments that allow us to come as close as possible to the theoretical setting and empirically confirm our results. To do so, we implemented the regression using the BernoulliGP model provided by GPyTorch [Gardner et al., 2018]. We use a Matérn kernel with automatic relevance detection with hyperparameters fit via maximum a posteriori optimized by the Adam algorithm [Kingma and Ba, 2014]. We tested on distributions of synthetic reward functions generated by sampling a random linear combination of Random Fourier Features [Rahimi and Recht, 2007] derived from a squared exponential kernel. For each sampled reward function r, we used the Bradley-Terry model where $p(w = 1 \mid a, a', x) = \frac{1}{1+\exp(r(x,a')-r(x,a))}$ to generate comparison data. For each trial we uniformly

sampled $n_0 = 25$ datapoints and then selected data to observe until $T = 500$ total datapoints had been collected according to one of three methods:

- **AE-Borda**: our method, as described in Section 5.4.1.
- **Uniform-Borda**: uniform sampling of both contexts and actions.
- **UCB-Borda**: uniform sampling of contexts, along with UCB actions as in AE-Borda.

This last method reduces to the method presented in Xu et al. [2020] when naively generalized to the contextual setting. All methods have the same test-time behavior of executing the action found by optimizing the pessimistic Borda function estimate for the test context. By optimizing the ground-truth reward function we were able to approximate the optimal policy and therefore estimate the regret of our policy against it. We give an example of the progression of our method for 1D context and 1D actions in Figure 5.3 as well as a comparison against Uniform-Borda and UCB-Borda in Figure 5.2. One can see that AE-Borda performs best both on median regret and on the maximum regret, which was the metric of interest in our theoretical analysis.

It is clear in Figure 5.3 that the method is quickly able to concentrate samples in regions that could plausibly be the optimum and it is similarly clear that the peaks in the acquisition function over contexts are sensible given the mean and uncertainty estimates of f_r. We give a set of results showing the progression of AE-Borda in Section C.6.

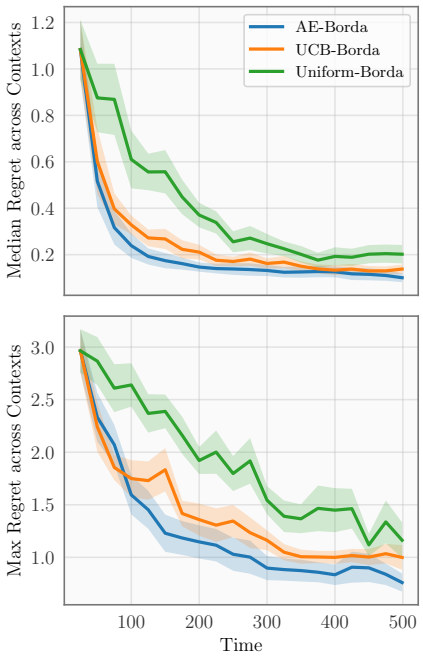

Figure 5.2: Performance of all methods across 10 random functions r with 1D Context and 1D action. The top plot shows the median regret across contexts and the bottom shows the maximum. Error bands show one standard error.

5.5 Scaling Active Exploration to Large Language Models

In order to adapt our method to the case where $\mathcal{X}$ and $\mathcal{A}$ are both large spaces of sequences as is common in natural language processing, we must address a few limitations of the AE-Bordamethod presented in Section 5.4.1:

- The contextual Borda function f_r as defined above is unsuitable for an action space that is extremely large and where most actions are obviously bad (a uniformly sampled sequence of tokens is trivially distinguishable from natural language).
- Neural network training proceeds in batches and it would be highly inefficient to label and train on a single example at a time.
- The uncertainty estimation tools in sequence modeling are more limited than those for explicitly kernelized models, especially due to the memory constraints in training LLMs.

We address these issues through a handful of modifications to our method as we specialize it to the LLM case. Though these modifications mean that we lose the theoretical guarantees in the previous section, we assert that given the rates of convergence associated with kernelized approximations of neural net architectures, we are not giving up strong guarantees in this setting. In particular, we modify the selection rule given in (5.3) to avoid having to use the Borda function, we naïvely do

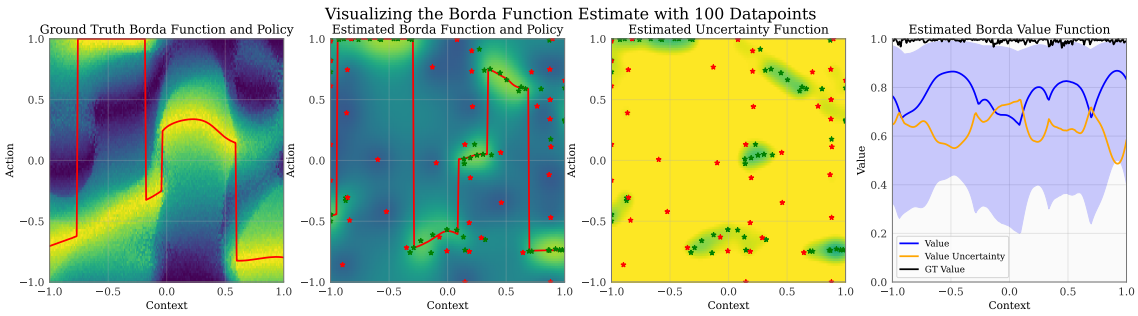

Figure 5.3: From left: the ground truth contextual Borda function f_r (the red line is the optimal policy), the mean of our posterior estimate of f_r (the red line is the best policy estimate), the uncertainty function σ_t, and the value function $\max_a f_r^t$. In the middle two plots, red dots are queries where $w_t = 0$ and green are where $w_t = 1$. We plot the value function with confidence intervals in blue on right as well as the value function uncertainty from (5.3) in orange. For a full version of this Figure, see Fig. C.6.

batched subset selection for our training minibatches, and we estimate the uncertainty of our policy using dropout for uncertainty estimation [Gal and Ghahramani, 2016]. In this section, we build atop the foundation presented in Rafailov et al. [2023], which avoids training a separate reward model; this is primarily due to the fact that we prefer to select datapoints based on the estimated uncertainty of the model used for decision making rather than any proxy.

Direct Preference Optimization Direct Preference Optimization (DPO) [Rafailov et al., 2023] avoids training a separate reward model based on preferences by instead training the policy directly on pairwise comparison using a loss that optimizes an equivalent objective despite functionally behaving like classification. As with PPO [Schulman et al., 2017], this loss depends on a reference policy, which we take to be the policy derived from the supervised fine-tuning step, π_{SFT}. The loss is defined as $\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{SFT}}) = -\mathbb{E}_{(x,a,a',w)\sim\mathcal{D}}\left[\log \rho\left(\gamma(2w-1)\left(\log \frac{\pi_\theta(a|x)}{\pi_{\text{SFT}}(a|x)} - \log \frac{\pi_\theta(a'|x)}{\pi_{\text{SFT}}(a'|x)}\right)\right)\right]$. The derivation in Rafailov et al. [2023] also shows that optimizing this objective is equivalent to training a PPO policy with reward function

$$r(x,a) = \gamma \log \frac{\pi_r(a \mid x)}{\pi_{\text{SFT}}(a \mid x)} + \gamma \log Z(x), \tag{5.9}$$

where γ is the hyperparameter of PPO scaling the KL penalty, $Z(x)$ is a partition function, and π_r is the policy which optimizes the PPO objective.

An Acquisition Function for DPO We observe as in the original paper that π_r is precisely the probability distribution which DPO is estimating. Therefore, the uncertainty estimates for our DPO policy are uncertainty estimates for π_r and we can use them to give an approximate confidence interval for r ($\overline{r}$ and $\underline{r}$). Concretely, we need to address the autoregressive nature of x and a in our case. We will assume a consists of ordered tokens t_i and that $\log \pi(a \mid x) = \sum_{t_i \in a} \log \pi(t_i \mid x, t_1, \ldots, t_{i-1})$. In our method, we employ dropout for uncertainty quantification. Specifically, the m dropout masks d_j are integrated into the function $\pi(t_i \mid x, t_1, \ldots, t_{i-1}, d_j)$. During inference, we perform autoregressive Monte Carlo sampling with dropout enabled, resulting in an ensemble of predictions with a mean $\mu(t_i \mid x, t_1, \ldots, t_{i-1}) = \frac{1}{m} \sum_{j\in[m]} \log \pi(t_i \mid x, t_1, \ldots, t_{i-1}, d_j)$. The standard deviation $\sigma(t_i \mid x, t_1, \ldots, t_{i-1}) = \sqrt{\frac{1}{m-1} \sum_{j\in[m]} \left(\log \pi(t_i \mid x, t_1, \ldots, t_{i-1}, d_j)\right)^2}$ across this ensemble serves as an approximation for the model's epistemic uncertainty. This technique allows us to

Algorithm 7 AE-DPO

Input: Reference policy π_{SFT}, exploration parameter β, policy constraint weight γ, batch size b, number of iterations N
for $t = n_0 + 1, \ldots, N$ **do**
  Draw an unlabeled batch $B_u = \{(x_i, a_i, a_i')\} \sim D$.
  Evaluate $\alpha(x_i)$ and let B_l be a batch of the top-b elements of B_u by α value.
  Collect labels r_i and add them to B_l.
  Update the policy π_θ (initialized from the ref.) using a gradient step against $\mathcal{L}_{\text{DPO}}$ using B_l.
end for
Output π_θ

capture uncertainty in a computation and memory efficient manner without compromising model performance. Given these estimates, we can compute our upper and lower bounds as follows:

$$\overline{r}(x, a) = \sum_{t_i \in a} \mu(t_i \mid x, t_1, \ldots, t_{i-1}) + \beta\sigma(t_i \mid x, t_1, \ldots, t_{i-1}) - \log \pi_{\text{SFT}}(a \mid x), \tag{5.10}$$

$$\underline{r}(x, a) = \sum_{t_i \in a} \mu(t_i \mid x, t_1, \ldots, t_{i-1}) - \beta\sigma(t_i \mid x, t_1, \ldots, t_{i-1}) - \log \pi_{\text{SFT}}(a \mid x), \tag{5.11}$$

for an uncertainty parameter $\beta > 0$. In the previous section, we chose contexts according to (5.3). Here, we define an acquisition function using a similar quantity:

$$\alpha(x) = \max_{a \in \mathcal{A}(x)} \overline{r}(x, a) - \max_{a \in \mathcal{A}(x)} \underline{r}(x, a). \tag{5.12}$$

In this equation, $\alpha(x)$ is the uncertainty of the state-value function according to x. In choosing the states where the potential for error in the value achieved is largest, the agent can learn to behave well in those places. This criterion is similar to that in Li et al. [2023] and provides similar guarantees to ours for max-regret in the active contextual bandit setting. In situations like ours where we are using fixed offline datasets, we set $\mathcal{A}(x)$ in (5.12) to the set of available responses for a particular action; otherwise, we use $\mathcal{A}(x) = \mathcal{A}$.

An algorithm for active RLHF From here, we use the acquisition function in (5.12) in order to choose points that are maximally informative. We must do this in batches in order to respect the constraints of training large models. We address this in the naive fashion, pulling a larger batch of some size, evaluating α and then choosing the top-b elements in order to address this criterion. We refer to our full procedure as AE-DPO, and give a description in Algorithm 7.

5.5.1 Experiments using LLMs

In order to evaluate whether our method is able to improve the selection of datapoints in RLHF, we conduct a set of experiments in which we train LLMs on three datasets using one of four methods. The goal of our empirical study is to see whether improving the data selection strategy causes the downstream policy to perform better on a given training task. In order to isolate the effect of the data selection method, we vary the selection method while largely holding our model and training procedure consistent. In all the experiments in this section, we compare four methods: **DPOAE**, the method we presented in Section 5.5; **USDPO**, which chooses x that maximize variance of the log probabilities of completions; **DPO**, the method from Rafailov et al. [2023], selecting batches uniformly at random; and **SFT**, which continues supervised learning with uniformly selected batches. In our training pipeline, we first train a baseline model with a Llama-7B [Touvron et al., 2023]

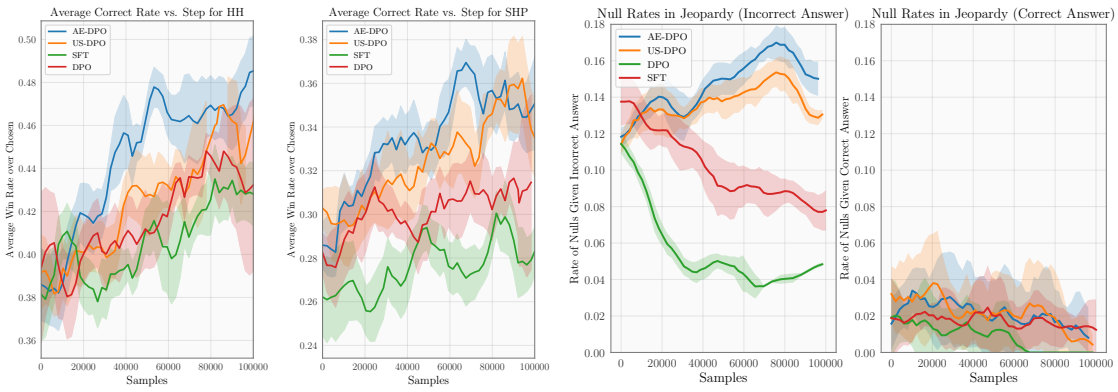

Figure 5.4: From left: smoothed win rates against preferred choices in dataset of samples generated from each policy at end of RLHF training runs across the final four evaluations, and all seeds, on the HH (first) and SHP (second) datasets. In the latter two plots, we force each policy to generate a (non-null) answer, and then, conditional on the answer being correct (fourth) or incorrect (third), plot the rate at which each policy abstains.

architecture using supervised fine-tuning (SFT) on a 40% split of data. We add a dropout layer before the penultimate linear layer for our uncertainty estimation mechanism and fine tune with dropout active. Next, we train using each of the four methods for 30000 samples, evaluating every 2048 samples—each time using our initial SFT trained model as a starting point. We give additional information on our experimental procedures in Section C.10.

We evaluate these methods on three different datasets. The first two, the Anthropic Helpful-Harmless (HH) dataset [Bai et al., 2022] and the Stanford Human Preferences (SHP) dataset [Ethayarajh et al., 2022], are taken from the literature. HH contains examples of two kinds: situations where an assistant needs to be helpful to a user asking a reasonable question and situations where an assistant should prioritize being harmless as the user is requesting a harmful action. All completions in HH are machine-generated. SHP is a dataset of Reddit posts with comments in 18 different categories and therefore consists of a broader range of human-generated text, but doesn't have the inherent tradeoff of HH. We evaluate policies trained on both of these by checking the rate at which the policy produces answers which are preferred to the chosen completion for the prompt in the dataset.

For the completions generated from the HH and SHP prompts, we use GPT-3.5 [Brown et al., 2020b] to generate winners amongst comparisons between the preferred choices given in the dataset. We give the prompts we use for evaluation in Section C.9. In Figure 5.2, we see that for the completions in the later part of our training run, AE-DPO performs best among the methods and outperforms US-DPO as well as the other baselines that sample uniformly. We believe this to be due to our acquisition function α, which accounts for the structure of the decision making problem in choosing which point to query. We do find our results to be noisy—due to the computational expense of these trials (which we elaborate on in Section C.11), we were not able to run each experimental baseline for a large number of seeds to further reduce uncertainty in our results.

We also introduce a new Jeopardy! dataset that includes a preference structure that captures some of the structure of the game while addressing the question of LLM hallucination. We augment a dataset consisting of 217,000 Jeopardy! questions and answers from HuggingFace [Wolf et al., 2023] with a plausible incorrect answer, using GPT-3.5. As in the game show, where points are deducted for an incorrect response, we enforce during training that a correct answer is preferred to an abstention (the empty string) and both of these should be preferred to the incorrect answer. We

found that our models do not learn to provide correct answers at a higher rate through a small amount of DPO training or additional SFT beyond what is required for them to answer the questions. This is unsurprising as trivia is intended not to generalize easily; in other words, it's difficult to imagine learning that the third US president was Jefferson given training examples of the first two. Instead, we evaluate policies for this dataset on the rate at which they abstain for questions ("null rate") where they counterfactually would have been correct vs where they would have been incorrect. Ideally, the policy learned would *always* abstain where it would have been incorrect and *never* abstain where it would have been correct. Naturally, this is an important goal in the alignment of LLMs and we hope to provide a straightforward benchmark for this effort. We include an additional exhibit where we use the factual nature of this dataset to begin to evaluate the dropout-based uncertainty estimation techniques we use in appendix C.12.1.

For the Jeopardy! dataset, we checked the probability of an empty generation and whether it was the most likely token. We generated a nonempty sequence in order to see whether the generated answer was correct, including as a counterfactual in the cases where the method would have abstained. We plot this in Figure 5.4, where we see that the AE-DPO method is the only method that learns to abstain from answering questions (modestly) more often when the model would have given the incorrect answer. We also find that the standard DPO method quickly learns not to abstain. No methods abstain more than a couple percent of the time in the case where they would have been correct. We also plot the results for correctness in Section C.12, which shows that no model substantially learns new factual information.

6 | Exploration and Sample-Efficient RL: Takeaways

In this part, we discussed work done over the past few years exploring the question of how to collect data that will be maximally valuable for learning good policies. Each of chapter 2, chapter 3, chapter 4, and chapter 5 gives a method for prospectively evaluating data prior to collecting it and then leverages that evaluation to make a good choice of what data to collect. These methods are applicable to different instances of the problem and have different strengths and weaknesses. They also are justified by somewhat different epistemic grounding. We conclude this part by discussing these methods in juxtaposition to one another and speculating as to where this field will head.

We began this part with the information-based methods in chapter 2. There, we define a heuristic acquisition function EIG_{τ^*} that estimates the expected information gain about the optimal trajectory τ^*. This acquisition function is computable in model-based RL settings where the dynamics model is a Gaussian process. Partly due to this restriction on the class of model and partly due to the superior data efficiency of model-based methods generally, these methods are the most sample-efficient of those discussed here. This method also relies on a model-predictive control strategy at deployment time (and during exploration, for the online TIP variant), so is less suitable for applications where a low latency is desired.

Next, we presented a strategy for exploration in Bayesian and meta-RL problems based on the EHIG presented in Neiswanger et al. [2022]. This method is most applicable to metalearning settings where you don't mind if your agent attains bad returns during exploration as long as the test-time policy is good. This is primarily accomplished by masking the returns of the policy during exploration and instead relying on inference of the expected returns during test time to drive exploration. We are able to give some simple theoretical arguments that justify the effect of this change in the tabular setting, but the extension to continuous problems is a heuristic that seems to work well in our theoretical setting.

In chapter 4, we illustrated the AE-LSVI method, which provably identifies a near-optimal policy in MDPs and contextual bandits in the active setting. It relies on assumptions that the value function is bounded in Reproducing Kernel Hilbert Space (RKHS) norm to give polynomial guarantees on sample complexity. Since the bounds given in this section are of a worst-case nature, this method is more suitable when it is not clear where in the state or context space the agent will be initialized. This method also relies on a kernel assumption and a GP estimate (this time for the value function), so doesn't give good guarantees for the neural network models used in high-dimensional spaces.

Finally, in chapter 5 we extended the AE-LSVI idea to bandit settings where comparative feedback is available. To do this, we first constructed an equivalent of the value function for dueling bandits. Next, we extended the arguments from the previous section to this case and showed that we

were able to attain similar guarantees even when the feedback was from a dueling problem. In order to do so, we needed some mild assumptions on the form of the link function. To our knowledge, this is the first efficient algorithm proposed for the offline contextual dueling bandit setting. As this setting is critical in the alignment of LLMs we extended our algorithm to the case where the function approximators are large neural network models. Though we abdicate our theoretical guarantees to do so, we find empirically that these methods perform well in practice.

The works presented here along with the wide literature on exploration give a solid foundation consisting of a variety of strategies that are suitable to different kinds of problems and come with differing guarantees. I believe that in the future, exploration and the prospective evaluation of decision making data will remain critical as AI systems become more sophisticated and integrated. Special interest will likely be paid to the meta-RL exploration techniques and those for comparative feedback. Though it is likely that future AI decision making systems have general world knowledge, there will always be a need to quickly align them to the job to be done. Separately from this, there will likely always be situations (such as in plasma control) where a policy must be found *de novo* as transfer or meta-learning may not be possible. In all these cases, I hope that this work serves as an inspiration and a tool to practitioners solving important problems.

Part II

Sample-Efficient Dynamics Modeling through Approximate Physical Knowledge

7 | Neural Dynamical Systems

The use of function approximators for dynamical system modeling has become increasingly common. This has proven quite effective when a substantial amount of real data is available relative to the complexity of the model being learned [Chua et al., 2018, Janner et al., 2019a, Chen et al., 1990]. These learned models are used for downstream applications such as model-based reinforcement learning [Nagabandi et al., 2018, Ross and Bagnell, 2012] or model-predictive control (MPC) [Wang and Ba, 2019]. As an alternative to data-hungry machine learning methods, there is also a long history of fitting models to a system using techniques from system identification, some of which include prior knowledge about the system drawn from human understanding [Nelles, 2013, Ljung et al., 2009, Sohlberg and Jacobsen, 2008]. These models, especially in the gray-box setting, are typically data-efficient and often contain interpretable model parameters. However, they are not well suited for the situation where the given prior knowledge is approximate or incomplete in nature. They also do not generally adapt to the situation where trajectories are drawn from a variety of parameter settings at test time. This is an especially crucial point as many systems of interest exhibit path-dependent dynamics which we aim to recover on the fly.

In total, system identification methods are sample efficient but inflexible given changing parameter settings and incomplete or approximate knowledge. Conversely, deep learning methods are more flexible at the cost of many more samples. In work already completed [Mehta et al., 2021], we make progress towards solving both of these problems by biasing the model class towards our physical model of dynamics. Physical models of dynamics are often given in the form of systems of ordinary differential equations (ODEs), which are ubiquitous and may have free parameters that specialize them to a given physical system. We develop a model that uses neural networks to predict the free parameters of an ODE system from the previous timesteps as well as residual terms added to each component of the system. To train this model, we integrate over the ODE and backpropagate gradients from the prediction error. This particular combination of prior knowledge and deep learning components is effective in quickly learning the dynamics and allows us to adjust system behavior in response to a wide variety of dynamic parameter settings. Even when the dynamical system is partially understood and only a subset of the ODEs are known, we find that our method still enjoys these benefits. We apply our algorithm to learning models in three synthetic settings: a generic model of ballistics, the Lorenz system [Lorenz, 1963], and a generalized cartpole problem, which we use for control as well. We also learn a high-level model of plasma dynamics for a fusion tokamak from real data. We discuss the related work in section 7.1 and provide a precise specification of the problem setting in section 7.2.

7.1 Related Work

There is a long tradition of forecasting physical dynamics with machine learning. Cressie and Wikle [2015] lays out ideas from classical statistics for predicting spatiotemporal data. In the deep learning world, recurrent neural networks and long short-term memory networks have been used for a long time to address sequential problems Hochreiter and Schmidhuber [1997]. However, these models struggle with continuous-time data and do not allow for the introduction of physical priors. Recently, there has been work in learning provably stable dynamics models by constraining the neural network to have a stable and jointly learned Lyapunov function Manek and Kolter [2020]. We see opportunities to use these techniques in follow up work.

Neural Ordinary Differential Equations As most numerical ODE solvers are algorithms involving differentiable operations, it has always been possible in principle to backpropagate through the steps of these solvers dating back to at least Runge [1895]. However, since each step of the solver involves calling the derivative function, naïve backpropagation incurs an $O(n)$ memory cost in the number of ODE solution steps, where n is the number of derivative calls made by the solver. Chen et al. [2018] demonstrated that by computing gradients through the adjoint sensitivity method, the memory complexity of backpropagating through a family of ODE solvers can be reduced to $O(1)$ for a fixed network, as opposed to the naive $O(n)$. However, this work only used generic neural networks as the derivative function and did not consider dynamics. They also provide a PyTorch package which we have built off of in our work.

There has been some work using neural ordinary differential equations to solve physical problems. Portwood et al. [2019] used a fully-connected neural ODE with an RNN encoder and decoder to model Navier-Stokes problems. Rudy et al. [2019] used a neural network integrated with a Runge-Kutta method for noise reduction and irregularly sampled data. There has also been work learning the structure of dynamical systems, first with a convolutional-deconvolutional warping scheme inspired by the solutions to advection-diffusion PDEs [de Bezenac et al., 2018], then with a Neural ODE which was forced to respect boundary conditions and a partial observation mechanism [Ayed et al., 2019]. There was also work on constraining a neural network model to respect separable conservative Hamiltonian dynamics, but like all of the aforementioned methods, this does not incorporate information about the dynamics of a particular system [Chen et al., 2019]. In general, none of these methods incorporate prior knowledge as explicitly as including appropriate equations in the statistical model. Furthermore, many of these methods focus on a specific problem, whereas we give a way to apply specific knowledge about a variety of problems.

An interesting approach that builds on a previous version of this paper [Mehta et al., 2021] is developed in Yin et al. [2021], where it is shown that iteratively solving a Lagrangian formulation of the problem for a fixed set of parameters and varying Lagrange multiplier results in predictions which maximally use the physical model. Our problem setting is a generalization of theirs in that we allow the parameters to vary among rollouts and predict them on the fly.

A further generalization is Rackauckas et al. [2020], which treats the whole problem of differential equations with missing components as a framework for modeling a very wide range of settings. We give a more prescriptive architecture for our class of problems and extend the analysis to a setting with real data as well as showing an application of our model for control.

Machine Learning for Nuclear Fusion: As far back as 1995, van Milligen et al. [1995] showed that by approximating the differential operator with a (single-layer, in their case) neural network,

one could fit simple cases of the Grad-Shafranov equation for magnetohydrodynamic equilibria. Recently, work has shown that plasma dynamics are amenable to neural network prediction. In particular, Kates-Harbeck et al. [2019] used a convolutional and LSTM-based architecture to predict possible plasma disruptions (when a plasma instability grows large and causes a loss of plasma containment and pressure). There has also been work in the field of plasma control: a neural network model of the neutral beam injection for the DIII-D tokamak has been deployed in order to diagnose the effect of controls on shots conducted at the reactor [Boyer et al., 2019b]. Additionally, Boyer et al. [2019a] used classic control techniques and a simpler model of the dynamics to develop a controller that allows characteristics of the tokamak plasma to be held at desired levels.

7.2 Problem Setting

Here we deal in dynamical systems $\dot{s} = f_\phi(s, a, t)$ with some parameters ϕ, a conventional model for system identification problems. The objective is to predict future states given past states, past and future controls, and prior knowledge of the form of f. We denote $s(\phi, t, \mathbf{a}, s_0) = s_0 + \int_0^t f_\phi(s, a, t)dt$ as the state obtained by integrating our dynamical system around f to time t. We consider in this section a more general setting and address the problem of prediction and control over a *class of dynamical systems*, which we define as the set $\{\dot{s} = f_\phi(s, a, t) \mid \phi \in \Phi\}$, where Φ is the space of parameters for the dynamical system (e.g. spring constant or terminal velocity). We can generate a trajectory from a class by sampling a $\phi \sim P(\Phi)$ for some distribution P and choosing initial conditions and controls. In real data, we can view nature as choosing, but not disclosing, ϕ. For a particular example j, we sample $\phi \sim P(\Phi)$ and $s_0 \sim p_0$ and are given controls $\mathbf{a}$ indexed as $a(t)$ and input data $\{s(\phi, t_i, \mathbf{a}, s_0)\}_{i=0}^{T}$ during training. At test time, we give a shorter, prefix time series $\{s(\phi, t_i, \mathbf{a}, s_0)\}_{i=0}^{T'}$ but assume access to future controls. Then the prediction objective for a class of systems for N examples for timesteps $\{t_i\}_{T'+1}^{T}$ is

$$\hat{s} = \text{argmin}_{\hat{s}} \mathop{\mathbb{E}}_{\substack{s_0 \sim p_0) \\ \phi \sim P(\Phi)}} \left[\sum_{i=T'+1}^{T} ||s(\phi, t_i, \mathbf{a}, s_0) - \hat{s}_{t_i}||_2^2 \right].$$

This objective differs from the traditional one in that implicitly, identifying ϕ for each trajectory needs to be done from the problem data in order to be able to predict the data generated by f_ϕ. Similarly, the control problem is

$$\mathbf{a} = \min_{\mathbf{a}} \mathbb{E}_{\phi \sim P(\Phi), s_0 \sim p_0} \left[\int_0^t c(a(t), s(t))dt \right],$$
$$\text{s.t. } s(t) = s_0 + \int_0^t f_\phi(s, a, t)\, dt$$

for some cost functional c. We will primarily explore the prediction problem in this setting, but as secondary considerations, we explore robustness to noise, the ability to handle irregularly spaced input data, and the ability to recover the parameters ϕ which generated the original trajectories. We will also consider the control problem in a simple setting.

7.3 Methods

We build up the description of our proposed method by first describing the two methods that inspire it: gray box system identification through optimization [Ljung et al., 2009], and using a Neural ODE

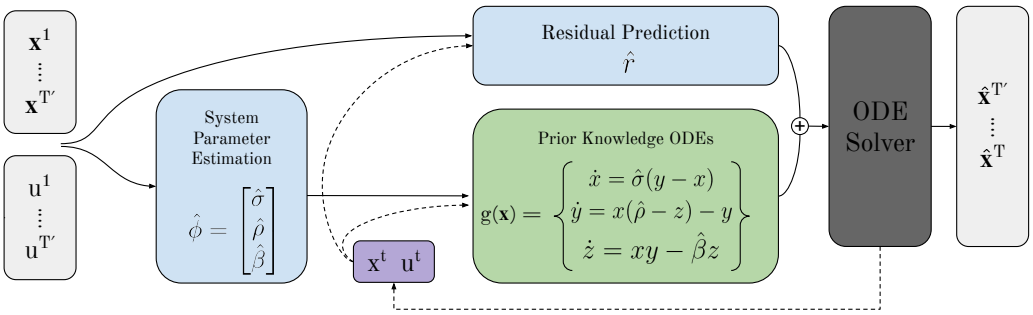

Figure 7.1: **An example Neural Dynamical System.** Here, blue boxes are fully connected neural networks, gray boxes are problem data and output, the green box is the prior knowledge dynamical system, the purple box is data output by ODE solver to query derivatives, and the black box is an ODE solver. The ODEs and system parameters are problem dependent, but here we consider the Lorenz system (defined in Example 1) as an example. Our notation for x is unfortunately overloaded by our method and the Lorenz system—the x from our method is bolded in the figure.

[Chen et al., 2018] to predict future states in a dynamical system.

To apply grey box optimization [Ljung et al., 2009] to a dynamical system $\dot{x} = f_\phi(x, u, t)$ for problem data $\{x_{t_i}\}_{t=0}^{T'}$, we would use nonlinear least squares [Coleman and Li, 1996] to find

$$\hat{\phi} = \text{argmin}_{\hat{\phi}} \sum_i \left|\left| \int_0^{t_i} f_\phi(s, a(t), t)\, dt - \int_0^{t_i} f_{\hat{\phi}}(s, a(t), t) dt \right|\right|. \tag{7.1}$$

This makes good use of the prior knowledge component of our system but is prone to compounding errors through integration and does not leverage data that may have come from alternate system parameters.

A data driven approach would be to minimize the same objective with a fully connected neural ODE [Chen et al., 2018] h_θ in place of f. However, we find that this procedure requires large amounts of training data and doesn't leverage any prior knowledge we might have, though it is flexible to classes of dynamical systems.

We define a Neural Dynamical System (NDS) by taking the advantages of both these methods in the setting where we know the full and correct ODEs and then show how to generalize it to situations where only some ODEs are known or they are approximate. Specifically, a *Neural Dynamical System* (NDS) is a class of dynamical systems where a neural network predicts some part of $f_\phi(s, a, t)$, usually parameters ϕ or a term which is added to f.

NDS with Full System Dynamics Consider a class of dynamical systems as defined in Section 7.2 where $s \in \mathbb{R}^n$, $a \in \mathbb{R}^m$, $\phi \in \mathbb{R}^{d_p}$, $d_h, d_c \in \mathbb{N}$ and let θ, ϑ, τ be trainable neural network weights. Let $h_\theta(s_{t_{1:T'}}, a_{t_{1:T}})$ be a neural net mapping state history and control sequence to the d_p parameters of the system $\hat{\phi}$ and an embedding $b_h \in \mathbb{R}^{d_h}$. Also let $c_\vartheta(s_t, a_t)$ be a similar network taking a single state and control that outputs an embedding $b_c \in \mathbb{R}^{d_c}$. Finally, let $d_\tau(b_h, b_c)$ be a network which takes the two output embeddings from the previous network and outputs residual terms $\hat{r}$. Intuitively, we would like to use the observed history to estimate our system parameters, and some combination of the observed history and current observation to estimate residuals, which influences the design of

our model, the *neural dynamical system* (a visualization of which is shown in Figure 7.1), written

$$\begin{aligned} \dot{s} &= \underbrace{g_{\hat{\phi}}(s_t, a_t, t)}_{\text{Prior knowledge}} + \hat{r} \quad \hat{\phi}, b_h = \underbrace{h_\theta(s_{t_{1:T'}}, a_{t_{1:T}})}_{\text{History encoder}} \\ b_c &= \underbrace{c_\vartheta(s_t, a_t)}_{\text{Context encoder}} \qquad \hat{r} = \underbrace{d_\tau(b_h, b_c)}_{\text{Residual prediction}} \end{aligned} \tag{7.2}$$

where g are domain-specific ODEs which are the input 'domain knowledge' about the system being modeled. Note that if the prior knowledge g is identically zero, this method reduces to the Neural ODE predictions we discussed at the beginning of this section. We also study an ablated model, NDS0, which lacks the residual component $\hat{r}$ and context encoder network d_τ. We note here that the context encoder is intended to potentially correct for model misspecification and noise but in the noiseless case with a model which is perfect, it may not be necessary. We explore this throughout Section 7.4.

Example 1: Lorenz system. To illustrate the full construction, we operate on the example of the the Lorenz system: a chaotic dynamical system originally defined to model atmospheric processes [Lorenz, 1963]. The system has 3-dimensional state (which we'll denote by x, y, z), 3 parameters, ρ, σ, and β, and no control input. The system is given by

$$\dot{x} = \sigma(y - x) \quad \dot{y} = x(\rho - z) - y \quad \dot{z} = xy - \beta z. \tag{7.3}$$

For a given instantiation of the Lorenz system, we have values of $\phi = [\beta, \sigma, \rho]$ that are constant across the trajectory. So, we can instantiate h_θ which outputs $\hat{\phi} = [\hat{\beta}, \hat{\sigma}, \hat{\rho}]$. We use the DOPRI5 method [Dormand and Prince, 1980] to integrate the full neural dynamical system in Equation 7.2, with g given by the system in Equation 7.3 using the adjoint method of Chen et al. [2018]. We use the state $x_{T'}$ as the initial condition for this integration. This gives a sequence $\{\hat{s}_t\}_{t=T'}^{T}$, which we evaluate and supervise with a loss of the form

$$\mathcal{L}_{\theta,\vartheta,\tau}(\{\hat{s}_{t_i}\}_{i=T'+1}^{T}, \{s_{t_i}\}_{t=T'+1}^{T}) = \sum_{t=T'+1}^{T} ||s_{t_i} - \hat{s}_{t_i}||_2^2. \tag{7.4}$$

Because of the way we generate our input data, this is equivalent to Equation 7.2. We assume in our setting with full dynamics that the true dynamics lie in the function class established in Equation 7.2. By the method in Chen et al. [2018] we can backpropagate gradients through this loss into the parameters of our NDS. Then algorithms in the SGD family will converge to a local minimum of our loss function.

NDS with Partial System Dynamics Suppose we only had prior knowledge about some of the components of our system and none about others. We can easily accommodate this incomplete information by simply 'zeroing out' the function. This looks like

$$g_\phi(s, a, t) = \left[\begin{cases} g_\phi^{(i)}(s, a, t) & \text{if } g_\phi^{(i)} \text{ is known,} \\ 0 & \text{else.} \end{cases} \right] \tag{7.5}$$

substituted into equation 7.2. In this setup, the residual term essentially makes the unknown dimensions unstructured Neural ODEs, which still can model time series well [Portwood et al., 2019].

NDS with Approximate System Dynamics For Neural Dynamical Systems to be useful, they must handle situations where the known model is approximate. This is transparently handled by our formulation of Neural Dynamical Systems: the parameters of the approximate model $\hat{\phi}$ are predicted by $h_\theta(s_{1:T'}, a_{1:T'})$ and the residuals $\hat{r}$ are predicted by $d_\tau(b_h, b_c)$. This is the same as in the case where we have the correct dynamics, but we remove the assumption of a perfect model.

Example 2: Nuclear Fusion System. In this paper, we apply this technique to plasma dynamics in a tokamak. In a tokamak, two quantities of interest are the stored energy of the plasma, which we denote E and its rotational frequency, ω. The neutral beams and microwave heating allow us to add power (P) and torque (T) to the plasma. The plasma also dissipates energy and rotational momentum via transport across the boundary of the plasma, radiative cooling, and other mechanisms. While the detailed evolution of these quantities is described by turbulent transport equations, for the purposes of control and design studies, physicists often use reduced, volume-averaged models. The simple linear model (up to variable parameters) used for control development in Boyer et al. [2019a] is used in this work.

$$\dot{E} = P - \frac{E}{\tau_e} \quad \dot{\omega} = \frac{T}{n_i m_i R_0} - \frac{\omega}{\tau_m} \tag{7.6}$$

Here, n_i is ion density, m_i is ion mass, and R_0 is the tokamak major radius. We use the constant known values for these. τ_e and τ_m are the confinement times of the plasma energy and angular momentum, which we treat as variable parameters (because they are!). These are predicted by the neural network in our model. We again use the model in Equation 7.2 to give us a neural dynamical system which can learn the real dynamics starting from this approximation in Section 7.4.2.

7.4 Experiments

In the following experiments, we aim to show that our methods improve predictions of physical systems by including prior dynamical knowledge. These improvements hold even as we vary the configurations between structured and fluid settings. We show that our models learn from less data and are more accurate, that they handle irregularly spaced data well, and that they learn the appropriate parameters of the prior knowledge systems even when they only ever see trajectories.

We use L2 error as our evaluation measure for predictive accuracy as given by Equation 7.4, though in cases where the absolute errors aren't very interpretable, we normalize for ease of comparison. We also evaluate our model's ability to predict the system parameters by computing the L2 error, i.e. $\sum_{i=1}^{n} ||\hat{\phi}_i - \phi_i||_2^2$. For synthetic examples, we consider the Lorenz system in (7.3) and a simple Ballistic system modeling projectile motion under a variety of drag conditions. We learn over trajectories $\{(s_{t_i}, a_{t_i}, t_i)\}_{i=1}^{T}$ where the s_{t_i} are generated by numerically integrating $\dot{s}_\phi(s, a, t)$ using scipy's odeint function [Virtanen et al., 2019], with s_0 and ϕ uniformly sampled from $\mathcal{S}$ and Φ, and a_{t_i} given. Note that ϕ remains fixed throughout a single trajectory. Details on the ranges of initial conditions and parameters sampled are in the appendix. We evaluate the synthetic experiments on a test set of 20,000 trajectories that is fixed for a particular random seed generated in the same way as the training data. We use a timestep of 0.5 seconds for the synthetic trajectories. On the Ballistic system this allows us to see trajectories that do not reach their peak and those that start to fall. Since the Lyapunov exponent of the Lorenz system is less than 3, in 16 predicted timesteps we get both predictable and unpredictable data [Frøyland and Alfsen, 1984]. We believe it is important to look at the progress of the system across this threshold to understand whether the NDS model is robust to chaotic dynamics — since the Lorenz system used for structure is itself chaotic, we want to make sure that the system does not blow up over longer timescales.

We note that ReLU activations were chosen for all feedforward and recurrent architectures, while in the Neural-ODE-based architectures, we follow the recommendations of Chen et al. [2018] and use the Softplus. The sizes and depths of the baselines were chosen after moderate hyperparameter search. We can view the Partial NDS and NODE as ablations of the Full NDS model which remove some and all of the prior knowledge, respectively. We discuss the training details in section D.1.1 comparison methods in section D.1.2.

7.4.1 Synthetic Experiments

We first present results on a pair of synthetic physical systems where the data is generated in a noiseless and regularly spaced setting.

Sample Complexity and Overall Accuracy In order to test sample complexity in learning or fitting, we generated data for a full training dataset of 100,000 trajectories. We then fit our models on different fractions of the training dataset: $1, 0.25, 0.05, 0.01$. We repeated this process with 5 different random seeds and computed the $L2$ error of the model over a the various dataset splits seen by the model in Table 7.1. The error regions are the standard error of the errors over the various seeds. We also see that with small amounts of data, the NDS models greatly outperform the Neural ODE, but with the full dataset, their performances get closer. This makes sense as the Neural ODE is likely able to infer the structure of the system with large amounts of data. Also, the Fully Connected Neural ODE outperforms the other baselines, which we posit is due to the fact that it implicitly represents that this system as a continuous time dynamical process and should change in a continuous fashion. From a sample-complexity perspective it makes sense that the better initialization of NDS should matter most when data is limited. A table of the full results of all experiments can be seen in Table 7.1.

We notice that the NDS0 slightly outperforms the NDS with higher variance on these systems. Since it has a perfect model of the system, the residual components aren't necessary for the model to perform well, however, there is no way the network can 'correct' for a bad estimate.

Curiously, we see on the ballistic system that the partial NDS slightly outperforms the full NDS in the small data setting, but they converge to similar performance with slightly more data. A potential explanation for this is that errors propagate through the dynamical model when the parameters are wrong, while the partial systems naturally dampen errors since, for example, $\dot{z}$ only depends on the other components through a neural network. Concretely this might look like a full NDS predicting the wrong Rayleigh number σ which might give errors to y which would then propagate to x and y. Conversely, this wouldn't happen as easily in a partial NDS because there are neural networks intermediating the components of the system. We also see APHYNITY (which fits a min-error parameter and is otherwise very similar to NDS) performs worse than NDS in this setting.

Parameter Learning without Explicit Supervision For experiments in Figure 2, we stored the parameter estimates $\hat{\phi}$ for the NDS and gray box models and compared them to the true values to see how they perform in identification rather than prediction. None of these models were ever supervised with the true parameters. We see in Figure 7.2 that the NDS is better able to estimate the parameter values than the gray-box method for both systems tested. We believe this is because our method is able to leverage many trajectories to infer the parameters whereas the gray-box method only uses the single trajectory.

System	Lorenz				Ballistic			
Samples	100,000	25,000	5,000	1,000	100,000	25,000	5,000	1,000
FC	1.06 ± 0.002	1.39 ± 0.003	1.694 ± 0.002	4.692 ± 0.6	7.8 ± 1.4	8.3 ± 1.6	9.2 ± 2.5	13.4 ± 3.4
FC NODE	1.205 ± 0.01	1.233 ± 0.01	1.27 ± 0.01	1.917 ± 0.12	2.53 ± 0.2	2.6 ± 0.4	4.8 ± 0.7	9.7 ± 1.3
Full NDS	$\mathbf{1.004 \pm 0.06}$	$\mathbf{1.087 \pm 0.06}$	1.14 ± 0.05	$1.42 \pm .05$	1.2 ± 0.1	1.4 ± 0.2	1.5 ± 0.2	4.23 ± 0.3
Partial NDS	1.036 ± 0.03	$\mathbf{1.064 \pm 0.1}$	$\mathbf{1.12 \pm 0.06}$	$\mathbf{1.39 \pm 0.04}$	$\mathbf{1.05 \pm 0.1}$	$\mathbf{1 \pm 0.03}$	$\mathbf{1.48 \pm 0.02}$	$\mathbf{1.9 \pm 0.15}$
NDS0	$\mathbf{1 \pm 0.03}$	$\mathbf{1.075 \pm 0.1}$	$\mathbf{1.13 \pm 0.11}$	$\mathbf{1.36 \pm 0.17}$	$\mathbf{1.2 \pm 0.06}$	$\mathbf{1.35 \pm 0.14}$	$\mathbf{1.45 \pm 0.18}$	4.3 ± 0.6
APHYNITY	1.08 ± 0.03	1.17 ± 0.05	1.23 ± 0.04	1.61 ± 0.07	1.7 ± 0.1	1.75 ± 0.13	1.94 ± 0.18	6.2 ± 0.4
LSTM	4.98 ± 0.01	5.98 ± 0.3	5.99 ± 0.3	6.13 ± 0.4	8.5 ± 1.6	9.2 ± 1.5	10.1 ± 2.1	14.9 ± 1.9
SR	2.3 ± 0.6	n/a	n/a	n/a	3.5 ± 0.3	n/a	n/a	n/a
GBO	2.8 ± 0.4	n/a	n/a	n/a	2.94 ± 0.3	n/a	n/a	n/a

Table 7.1: Sample Complexity Results as discussed in Section 7.4.1. Here, the values are normalized by the smallest reported value for comparison purposes.

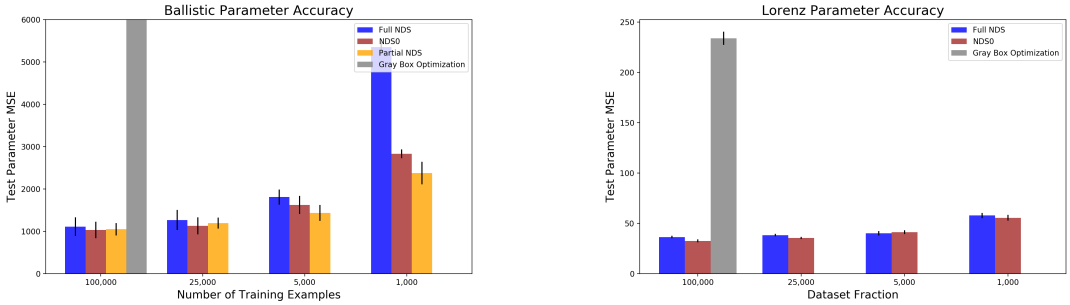

Figure 7.2: $L2$ **distance between** ϕ **and** $\hat{\phi}$**.** As the NDS are trained under the usual $L2$ supervision, the parameters $\hat{\phi}$ of the system approach the correct values.

7.4.2 Fusion Experiments

We explored the concept of approximate system dynamics in a simplified fusion system. We predict the state of the tokamak as summarized by its stored energy and rotational frequency given the time series of control input in the form of injected power and torque. As mentioned in Section 7.3, we have a simplified physical model given by Equation 7.6 that approximately gives the dynamics of these quantities and how they relate to one another through time. Though there is a lot of remaining work to apply this model in a real experiment, approaches merging theoretical models with data to make useful predictions can be embedded into useful controller designs and improve the state of fusion.

Our full dataset consisted of 17,686 trajectories, which we randomly partitioned into 1000 as a test set and 16,686 as a training set.[1] The data are measured from the DIII-D tokamak via magnetic reconstruction [Ferron et al., 1998] and charge-exchange recombination spectroscopy [Haskey et al., 2018]. Similar to our synthetic experiments, we cut each trajectory into overlapping 48 timestep sections and train on 32 timesteps to predict 16 timesteps. We compare with the same models as in the previous section, but using our Fusion Neural Dynamical System as described in Equation 7.2 with g given by Equation 7.6. As we discussed above, the dynamics in this equation are approximate. To illustrate this, we have included the accuracy of the naive dynamics with no learning on our data with fixed confinement times $\tau_e = \tau_m = 0.1$s as the Nominal Fusion Model in Table 7.2. We use a

[1]Data is loaded and partially processed within the OMFIT framework [Meneghini et al., 2015]. We used the "SIGNAL_PROCESSING" module which has recently been developed for this task and is publicly available on the "profile_prediction_data_processing" branch of the OMFIT source code. Details of the preprocessing are in the Appendix.

Model	$L2$ Error on the Fusion Test Set
FC	4.02 ± 0.27
FC NODE	1.71 ± 0.11
Nominal Fusion Model	2.89
NDS with Approximate Dynamics	$\mathbf{1 \pm 0.06}$
NDS0	1.85 ± 0.09
APHYNITY	1.21 ± 0.15
LSTM	5.23 ± 0.43
SR	5.26 ± 0.35
GBO	2.98

Table 7.2: The performance of our comparison models on the nuclear fusion problem, as discussed in Section 7.4.2. We again normalize by the smallest value for ease of comparison.

larger fully connected network with 6 layers with 512 hidden nodes to attempt to capture the added complexity of the problem.

Sample Complexity and Overall Accuracy When comparing our NDS models, the machine learning baselines, the system ID baselines, and a nominal model from Boyer et al. [2019b], we see that the Fusion NDS model performs best by a large margin. Although the fully connected neural ODE performs competitively, it fails to reach the same performance. We speculate that the dynamical model helps with generalization whereas the fully connected network may overfit the training data and fail to reach good performance on the test set. Here the NDS0 is unable to perform well compared to the NDS, as the approximate dynamics mean that the model error is somewhat catastrophic for predictions. We see however that the NDS0 outperforms the Nominal Physics Model as it is able to estimate the parameters for each example rather than fixing values of the parameters for the whole dataset. Similarly, APHYNITY outperforms the nominal model but underperforms the NDS0, suggesting that the online parameter estimation component is necessary for good performance.

We see these results as highly encouraging and will continue exploring uses of NDS in fusion applications.

7.4.3 Control Experiment

We also explored the use of these models for control purposes using model-predictive control [Camacho and Alba, 2013]. For this purpose, we modified the Cartpole problem from Brockman et al. [2016] so that there are a variety of parameter values for the weight of the cart and pole as well as pole length. Typically, a 'solved' cartpole environment would imply a consistent performance of 200 from a control algorithm. However, there are three factors that make this problem more difficult. First, in order to allow each algorithm to identify the system in order to make appropriate control decisions, we begin each rollout with 8 random actions. The control never fails at this point but would certainly fail soon after if continued. Second, the randomly sampled parameters per rollout make the actual control problem more difficult as the environment responds less consistently to control. For example, MPC (as discussed in Section 1.1.1) using the typical Cartpole environment as a model results in rewards of approximately 37. Third, all training data for these algorithms uses random actions with no exploration, which has been seen to degrade the performance of most

	MSE of Model		MPC Returns	
# Train	5K	1K	5K	1K
FC	0.031 ± 0.009	0.058 ± 0.018	52 ± 3	41 ± 4
FC NODE	0.028 ± 0.011	0.049 ± 0.013	55 ± 4	46 ± 3
LSTM	0.081 ± 0.023	0.092 ± 0.025	23 ± 6	25 ± 8
Full NDS	$\mathbf{0.020 \pm 0.006}$	$\mathbf{0.029 \pm 0.007}$	$\mathbf{72 \pm 4}$	$\mathbf{60 \pm 3}$
Partial NDS	$\mathbf{0.022 \pm 0.009}$	$\mathbf{0.033 \pm 0.011}$	$\mathbf{69 \pm 8}$	55 ± 6
NDS0	$\mathbf{0.023 \pm 0.013}$	$\mathbf{0.028 \pm 0.014}$	$\mathbf{71 \pm 11}$	$\mathbf{57 \pm 8}$
SR	0.037 ± 0.023	0.041 ± 0.015	65 ± 4	56 ± 4
GBO	0.046 ± 0.019	n/a	49 ± 5	n/a

Table 7.3: Modeling and Control on the EvilCartpole system.

model-based RL or control algorithms [Mozer and Bachrach, 1990].

We then trained dynamics models on this 'EvilCartpole' enviroment for each of our comparison algorithms on datasets of trajectories on the environment with random actions. At that point, we rolled out trajectories on our EvilCartpole environment using MPC with control sequences and random shooting with 1,000 samples and a horizon of 10 timesteps. The uncertainties are standard errors over 5 separately trained models.

As shown in Table 7.3, the NDS algorithms outperform all baselines on the cartpole task for both the modeling and control objectives. We see that all algorithms degrade in performance as the amount of data is limited. We notice however that with larger amounts of data (we performed other experiments with 25,000 and 100,000 samples) the Fully Connected and Neural ODE models perform as well as the NDS models. We hypothesize that this is due to the fact that the cartpole dynamics are ultimately not that complicated and with sufficient data unstructured machine learning algorithms can learn the appropriate dynamics to reach a modestly performing controller as well as NDS.

7.5 Discussion and Future Work

In conclusion, we give a framework that merges theoretical dynamical system models with deep learning by backpropagating through a numerical ODE solver. This framework succeeds even when there is a partial or approximate model of the system. We show there is an empirical reduction in sample complexity and increase in accuracy on two synthetic systems and on a real nuclear fusion dataset. In the years since this paper, the physically informed modeling of dynamical systems has progressed substantially [Karniadakis et al., 2021, Yin et al., 2021]. We hope that this field continues to improve our understanding of how to synthesize the advantages of first-principles modeling and of data-driven estimation of complex systems.

Part III

Applications of learning in Plasma Control

8 | Plasma Control in Tokamaks

As we mentioned in the introduction, nuclear fusion has extremely attractive properties due to the energy density of fusion reactions. In this chapter we'll begin by giving an overview of the tokamak control problem as I understand it. Next in chapter 9 we discuss an application of BO-style open loop control design to the design of the plasma current rampdown. We conclude this part in chapter 10 by discussing some of the other experiments we attempted as well as giving some broader commentary on applying machine learning to fusion.

We will begin here by motivating the plasma control problem by grounding it in quantities a power plant operator might care about. Then we discuss the available sensors and actuators, the concrete process of running a shot, and key limits and constraints of the system. The goal of this section is that a reader with a background at the level of a typical computer science graduate student should walk away with an idea of the 'state of the game board' in fusion and an idea of what problems might be meaningful to work on. Much of this information is based on work in Freidberg [2007], Smith and Cowley [2010], and Walker et al. [2020].

Global energy usage continues to grow rapidly [Jackson et al., 2018] into the present day. As use of energy correlates with the level of human development [Yumashev et al., 2020], it is imperative that we find sustainable ways to grow our civilizational ability to harness and direct energy to a wide variety of ends. Of the technologies currently being contemplated for providing us with energy at scale, nuclear fusion has a variety of attractive properties:

- Fusion has an extremely high energy density per gram of fuel and should scale to far larger energy budgets than humanity currently desires.
- By itself, fusion uses no heavy radioactive elements that are needed for the proliferation of nuclear weapons.
- Fusion is a 'firm' energy generation technology and in principle could be depended on regardless of the weather or circumstances on the grid.
- The radioactive waste produced by a fusion plant would be relatively low-level and easy to store.
- Fusion plants are passively stable; in the absence of active control input, the reaction will subside and there is zero risk of a chain reaction.

This is also an exciting time in fusion. The ITER project [Holtkamp et al., 2007], a collaboration of most major world governments to build a large tokamak in the south of France, is nearing the end of its construction phase and aims for first plasma this decade and net energy gain from fusion in the mid-2030s. Advances in plasma physics, computing, and material science have driven research forward at the existing devices. There has been a substantial growth in private investment, with a cumulative $6.21B invested in private companies attempting to bring fusion power to market [Fusion Industry Association, 2023]. With this said, there are still substantial challenges remaining

before humanity is able to power substantial amounts of its activities with terrestrial fusion reactors.

8.1 Achieving Net Energy from Fusion

Fusion reactions produce energy because the strong nuclear force leads to an attraction between nucleons (protons and neutrons) at very short distances that leads to more nuclear binding energy for larger nuclei (up to iron) than the equivalent nucleons in a pair of smaller nuclei. The majority of fusion efforts today focus on the D-T reaction between deuterium (hydrogen with a neutron) and tritium (hydrogen with two neutrons). The D-T reaction proceeds as below:

$$\underbrace{\mathrm{D} + \mathrm{T} \rightarrow {}^{4}\mathrm{He} + \mathrm{n} + 17.6\,\mathrm{MeV}}_{\text{D-T Fusion}} \tag{8.1}$$

However, it has thus far been extremely difficult to achieve the conditions required for fusion. The primary difficulty in achieving fusion is that the protons in one nucleus repel the protons in another nucleus via the Coulomb force, which acts at longer distances than the strong nuclear force. In order to overcome this repulsion, the nuclei must be heated to very high temperatures exceeding the ionization energy of the atoms they are part of. Therefore, as part of the process of heating, the gas is ionized and becomes a plasma. Though a single particle pair will have some relative kinetic energy and therefore some probability of a reaction upon collision, a volume containing particles will have a distribution of energies and it is therefore convenient to discuss the volume averaged reactivity of a plasma, which we write $\langle \sigma v \rangle$. This quantity depends only on the species of nuclei reacting and the the temperature. Then the fusion power per unit volume of plasma is $\mathcal{P}_{\text{Fusion}} = n_1 n_2 \langle \sigma v \rangle \delta W$, where n_1 and n_2 are the densities of the 2 species reacting (in m^{-3}) and δW is the energy released by a single reaction (in the case of D-T it is 17.6 MeV as in eq. (8.1)). Practically, after some steps of derivation and after substituting known constants for deuterium and tritium, we have that

$$\mathcal{P}_{\text{Fusion}} \sim 0.08 P^2 \frac{\mathrm{MW}}{\mathrm{m}^3}, \tag{8.2}$$

where P is the plasma pressure in atmospheres [Smith and Cowley, 2010].

As the temperatures required to achieve this (around $100\,\mathrm{M}^{\circ}\mathrm{C}$) are too high for any material, the plasma must be confined by magnetic fields. The most developed design for doing so is a twisted toroidal shape known as a tokamak. In a tokamak, a plasma current in the toroidal direction is driven by a central solenoid (CS) via transformer action. Alongside this solenoid, toroidal and poloidal field coils are used to produce a magnetic field from the sum of these sources such that in a toroidal volume all field lines eventually close on themselves. These can be seen in fig. 8.3. Since all particles in the plasma are charged, the plasma organizes itself along these lines and is thereby confined in the interior of the vessel.

Actuators The heat in the plasma comes from 4 sources:

- Ohmic heating: heat from the fact that the plasma is resistive and has current running through it.
- Neutral beam injection (NBI): one of the major actuators available to tokamak controllers, the neutral beams fire beams of high-energy deuterium ions and electrons into the plasma as a heating source, a fueling source, and a source of torque.

- Electron cyclotron heating (ECH): microwaves are fired into the plasma and absorbed by the electrons with the same resonance frequency. This allows heating to be precisely localized to within a centimeter or so.
- Fusion reactions as in eq. (8.1) (in future devices).

Besides the currents in each coil, NBI, and ECH, controllers have the ability to control the density by puffing gas or firing frozen pellets into the reactor. Together, the currents in all coils, the heating powers, and the density injections comprise the set of actuators $\mathcal{A}$ available on a tokamak. The key question of tokamak control is what to do with them. To date, most reinforcement learning work Char et al. [2023], Abbate et al. [2021b], Seo et al. [2022] has focused on controlling the neutral beams and shape parameters. For both of these actuators and each of the others, there are substantial and varying limitations on usage due to safety and hardware constraints. For NBI, these limitations include that intermediate beam powers are achieved by modulating the beam on and off within constraints on the rate of switching, that beams often fail within a particular shot, and that for various plasma configurations there are limitations on beam power due to safety considerations [Logan et al., 2023]. There are often limits on the availability of gyrotrons to provide ECH. Similarly all coils come with current limits and the overall system operates within power constraints.

Observations As a controller to decide what actions to take, it relies on state information reconstructed from a variety of diagnostics. The state space includes 1-dimensional discretized *profiles* of the pressure, temperature, current density, number density, and rotation of the plasma. In addition, we recieve information on the shape and position of the last closed flux surface, the strength of the toroidal magnetic field, the inductance, various magnetic measurements, and a wide range of other physical phenomena. This information comes in at varying frequencies, with direct measurements of physical phenomena (such as the current) arriving at a much higher frequency than reconstructed values such as the 1D profiles.

There are a large number of diagnostics on the DIII-Ddevice, as depicted in fig. 8.2. These include various RF scattering diagnostics, coils that sense changes in the external magnetic field, diagnostics that vary the neutral beams to measure the radiation response, a motional Stark effect diagnostic to measure current density, and many others. These give observations of the plasma that can be used along with known physics to infer the internal quantities of the plasma.

In order to estimate the state (profiles, plasma boundary, etc) from the measurements, the magnetohydrodynamic equilibrium of the plasma is reconstructed from the sensor measurements above [Lao et al., 2005]. This is accomplished via an iterative linearization and solution of the Grad-Shafranov equations subject to the constraints imposed by observed measurements. These EFITs are computed at a rate of 20 kHz during shots and then at a rate of afterwards at a higher fidelity. We typically use the outputs of these EFIT calculations for state variables for controllers; however, one more recent thrust of work aims to directly model the future diagnostic measurements from the history observed.

Timeline of a Shot In this section, we'll follow Walker et al. [2020] through the timeline of a typical shot at DIII-D as an example. A typical day at DIII-D is devoted to one or two experiments. Experiments are allocated on long-term planning cycles based on a competitive research proposal process to scientists. The schedule is globally optimized in order to account for the various configurations needed to accommodate the research program. For example, the beams at DIII-D are capable of rotating variously in the toroidal and poloidal planes, heating system availability varies widely, and operations using other species are sometimes conducted.

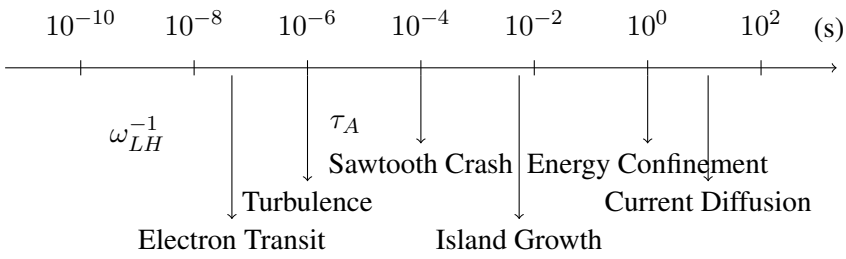

Figure 8.1: Characteristic Timescales of Phenomena in a Tokamak

Once the tokamak is configured for operation in an experiment (a process which includes a reorganization of the power supply systems of the tokamak to ensure appropriate electricity is available to support the planned settings and manual adjustment of a high-current patch panel), shots commence on a 10-15 minute cadence over the course of 4 hours. In that time, the operators orient themselves to incoming data, debug software and hardware issues, consult with each other and external resources, and ultimately adjust the settings of the plasma control system (PCS) in order to pursue their experimental goals. The cognitive load on operators in the control room is high; later in chapter 10, we discuss other work that takes first steps towards machine-learning driven tools to assist operators in this environment.

Prior to the beginning of a shot, the toroidal field coil is brought to a constant level and deuterium gas is puffed into the vessel. At $t = 0$, the current in the CS is rapidly driven down in order to induce current in the torus and ionize D atoms via a generated electric field. At this point, the CS drives the plasma current through transformer action in a linear ramp up to some flat-top level over about 1s. The poloidal field coils concurrently control the plasma position and shape. The plasma heats due to Ohm's law. Additional heating is provided by NBI and ECH over up to 5 seconds of flat top where the system is driven to and stabilized at experimental conditions. Finally, the current is brought down by the CS and the shot concludes. The three periods described here are commonly referred to as the *ramp-up*, *flat-top*, and *ramp-down* phases of the shot.

In this section, we gave a high level description of the timeline of a shot. However, as fig. 8.1 shows, the relevant phenomena for plasma control operate on a wide range of timescales; controllers must be designed that operate in harmony over this wide range.

Objectives in Plasma Control At an extremely high level, magnetic confinement of plasma depends on a magnetic force containing the plasma pressure. This 'magnetic pressure' is approximately proportional to B^2 for a magnetic field of B Tesla. One measure of the efficiency of such confinement is the ratio between the plasma pressure and this magnetic pressure, which we denote $\beta = \frac{P}{B^2/2\mu_0}$ for a plasma pressure P in atmospheres. Substituting this definition into eq. (8.2) gives that $\mathcal{P}_{\text{Fusion}} \propto \beta^2 B^4$. Since the magnetic field is essentially fixed once the tokamak is built, we often choose to optimize β as our key measure of performance of a plasma, subject to several operating limits derived from data, physical principles, and engineering design choices.

Limits on Plasma Operating Conditions There are a number of limits observed or derived on the state of the plasma and therefore of controller designs. These include:

- the **Troyon limit** [Troyon et al., 1984], which tells us that in a conductive plasma the maximum β is approximately $\frac{I_p}{aB}$, where I_p is the plasma current in MA< a is the minor radius in meters and B the magnetic field in Tesla.

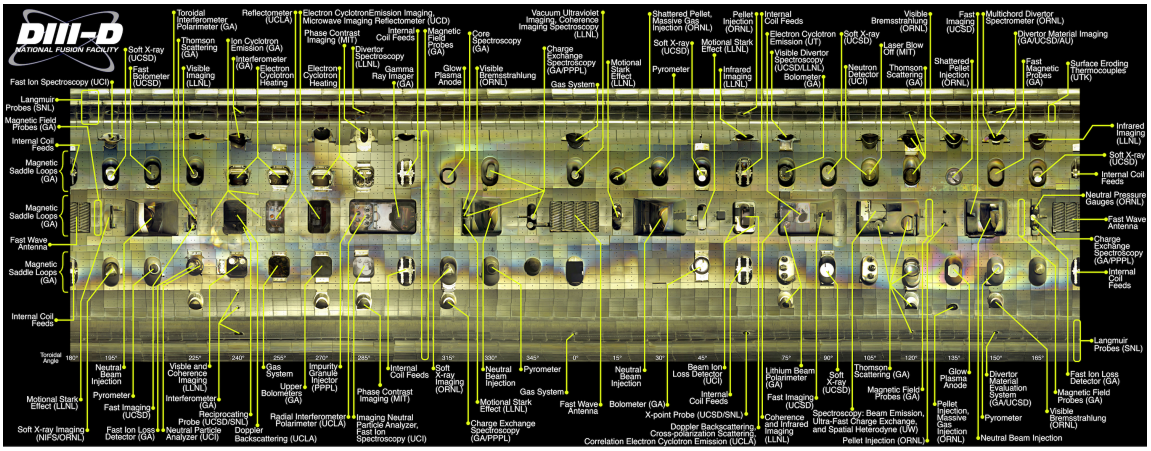

Figure 8.2: A panorama of the DIII-D Tokamak with many diagnostic systems labeled.

- the **Greenwald limit** [Greenwald et al., 1988], an empirically determined limit on the density of a tokamak plasma. It says that the maximum density $n_G = \frac{I_p}{\pi a^2}$ when measured in $10^{20} m^{-3}$. We often speak of the Greenwald fraction $f_{GW} = \frac{n}{n_G}$ as shorthand to understand how close the density is to this limit. The Greenwald limit is 'soft' and can be exceeded under certain operating conditions.
- the **safety factor limits**, which limit how high we can drive current. At a high level, field lines in a tokamak take a helical path around the major axis with the number of poloidal revolutions per toroidal one inversely proportional to the plasma current enclosed by the poloidal orbit. This ratio is known as the safety factor and denoted as the q-profile. Many of the instabilities that plague tokamaks are due to low values of q. In particular, values below 1 and integral numbers are particularly problematic.
- Flux consumption **limits on steady-state operation**: typically, the plasma current is driven by the change in current in the CS of the tokamak via Faraday's law. However, there is an inevitable limit on the flux that can be produced via a change in current in a solenoid. If the operating conditions of the tokamak require consistent current drive from the solenoid, they must inevitably allow for operation to pause so that the solenoid can be recharged. This is inconvenient for operating power plants, and one goal of plasma control is to find operating regimes that don't rely on this kind of current drive at steady state and instead achieve the required current from other sources.
- **Materials limits**, most notably the heat flux tolerable by the divertor materials. As particles are vented from the confinement area to an area of the tokamak called the *divertor*, it will be exposed to radiation fluxes of the order of $10 MW m^{-2}$ [Smith and Cowley, 2010]. Jointly searching for control strategies to ameliorate this and for materials that are durably capable of tolerating these conditions will be one of the key questions to be answered in future fusion research.

Disruptions Disruptions, which are sudden and unplanned terminations of plasma confinement, are the primary adverse event seen in tokamak operations. They can be detrimental to the structure and operations of a tokamak and operators take care to avoid them. They are typically caused by exceeding one of the first three limits above, a radiative collapse of the plasma due to impurities, or a

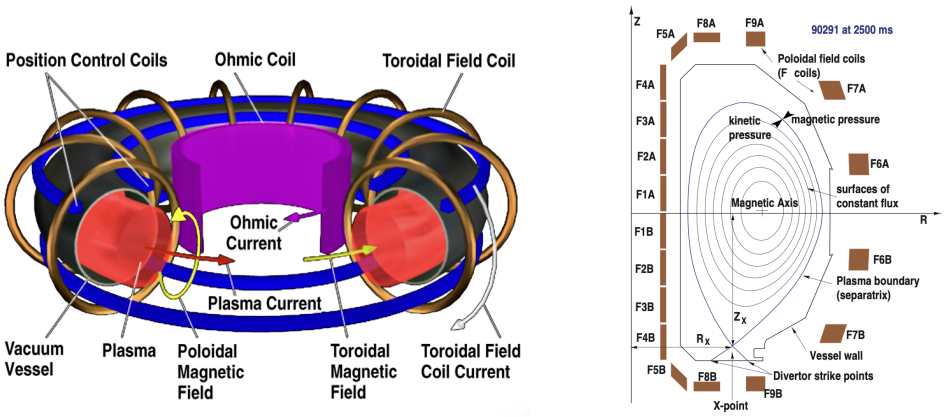

Figure 8.3: Left: a schematic drawing of a tokamak and its major magnetic components. Right: a typical plasma shape at equilibrium. The various boxes labeled with Fs are magnetic coils tasked with attaining the desired plasma shape.

loss of vertical position control (as we'll discuss later). One further instability that must be addressed by future fusion reactors is the neoclassical tearing mode (NTM), a "3D helical deformation of the plasma around certain flux surfaces where the magnetic field closes on itself in a rational number of turns around the torus," [Walker et al., 2020] which can be addressed through effective control of electron-cyclotron heating systems or avoided entirely through a choice of operating point. For ITER and future fusion reactors geared towards power generation, disruptions pose a significant concern. At ITER, the engineering constraints allow for a maximum of 3,000 major disruptions (equating to 10% of the anticipated full-performance pulses) and 300 vertical displacement events (VDEs) [Sannazzaro et al., 2009]. Therefore, identifying operating regimes and control strategies that reduce disruption risk is vital in order to remain within these limits. The physics of disruptions is poorly understood — most progress of physics during disruptions has been made by solving the extended MHD equations with codes like NIMROD Sovinec et al. [2004]. But this is expensive and still does not achieve quantitative accuracy. What's more, as outlined by De Vries et al. [2011] and Kates-Harbeck et al. [2019], disruption *prediction* is a harder task still due to the disparate disruption causes, so empirical models are the state of the art and are far from perfect Fu et al. [2020b].

8.2 Key Tokamak Control Problems

8.2.1 Shape, Power, and Current Control

Perhaps the simplest control problems on the tokamak are of current, position, and shape. As the closed-form dynamics of magnetic flux and plasmas are easily understood through the assumption that the plasma is toroidally axisymmetric, linear controllers which attain the desired plasma shape by varying the currents through the various coils (as seen in Figure 8.3) are available. The magnetic controls are typically determined by a linearization of the Grad-Shafranov equation [Walker et al., 2020]; this field is fairly mature [Pironti and Walker, 2005, Ariola et al., 2008] As we'll discuss in section 8.3, recent work shows that it is also possible to train policies via reinforcement learning to solve this problem. In many other recent applications of machine learning [Char et al., 2019, Abbate et al., 2021b, Seo et al., 2021], parameters of the target shape set for the low-level controller are

used as virtual actuators by the higher-level learning based control algorithm rather than directly controlling coil currents.

8.2.2 3D Control

There are additional coils on a tokamak designed to address the fact that the magnetic fields and plasma quantities are not perfectly axisymmetric. In particular, fabrication errors lead to small deviations in the magnetic field that can be corrected by application of 3D magnetic flux [Reiman and Monticello, 1991] and even in the absence of these, global 3D instabilities known as Resistive Wall Modes can form and must be corrected by 3D magnetic control [Ariola et al., 2014]. To my knowledge, there has not been reinforcement learning research addressing this problem, but it could potentially be a fruitful application especially as ML work on the core problem of turbulence progresses [Zhang and Duraisamy, 2015].

8.2.3 Kinetic Control of Plasma

The regulation of key properties of the plasma such as current, temperature, density, rotation, and pressure is foundational to the eventual success of the fusion project. Improvements in confinement and power density are driven by advanced modes of operation determined by these factors. The dynamics of density, temperature, and rotation are determined by the solutions to transport equations as given in John [2005]. These quantities are challenging to control as the actuation capabilities are more limited. The story is similar for the control of the current and rotation profiles, where actuators for both torque and current drive are sparse. Two additional observations add to the challenge of kinetic control: the profiles are themselves nonlinearly coupled and often share the same actuators that may need to be applied to multiple objectives in order to regulate the plasma. This suggests a role for integrated controllers; reinforcement learning is a prime candidate for the development of such controllers though it also brings difficulties.

8.3 Prior Work

Over the past 5 years, work has begun applying learning-based control to tokamaks. One early work [Boyer et al., 2018] applies neural network modeling in order to accurately predict the deposition profile of the neutral beams of the NSTX-U tokamaks in anticipation of control applications. Kates-Harbeck et al. [2019] trained a very large Fusion Recurrent Neural network to predict instabilities and disruptions in plasma so that they can be mitigated online. This method showed impressive generalization to tokamaks that hadn't been seen before. However, the work did not contain a major control component itself. In a similar spirit Fu et al. [2020b] successfully trained a tree-based predictor to notice when disruptions or NTMs were likely to happen in the near future. This work used a very simple control strategy of decreasing the power when these predictions were made and was able to avoid some tearing modes and disruptions. The method in Fu et al. [2020b] is able to avoid some tearing modes and disruptions, but it is not clear how to extend this method to a more general control strategy. There are several other works [Parsons, 2017, Rea et al., 2019, Boyer et al., 2021] which apply machine learning to predict disruptions and then attempt control strategies to avoid them.

One early work by Baltz et al. [2017] begins a line of work applying black-box optimization to plasma control by using their optometrist algorithm to optimize the parameters of a field-reversed configuration (a tokamak alternative). Another earlier work by Char et al. [2019] developed an

algorithm for contextual optimization of myopic controls for maximizing β_N while maintaining stability on the TRANSP simulator [Breslau et al., 2018]. In another foundational work [Abbate et al., 2021b], DIII-Ddata was processed and used to train a dynamics model predicting profiles over time. This model was used for a finite-set controller which drove temperature profiles to a desired state.

One recent work by Degrave et al. [2022] was able to use a simulator of the magnetic control problem (what currents to run through electromagnetic coils in order to achieve a desired plasma shape) with randomized dynamic parameters as an environment in which to train a reinforcement learning algorithm, MPO [Abdolmaleki et al., 2018], which was able to achieve relatively good magnetic control in experiments on the TGV tokamak. Works like Seo et al. [2021, 2022], Char et al. [2023] learn a policy from previously logged data collected by the experiments that have been run on KSTAR and DIII-D, respectively. These latter works fall under the "offline reinforcement learning" setting Levine et al. [2020]; there are inherent challenges associated with this setting such as the fact that the training data was generated by some other policy (fusion scientists running experiments) rather than the agent. As tokamak time is exceedingly limited, it is in general impossible to train controllers in an online fashion (on the machine, with opportunity for the agent to learn from its previous experiences) for fusion applications.

9 | Automated Experimental Design of Safe Rampdowns via Probabilistic Machine Learning

9.1 Introduction

The termination phase of a shot is an essential part of tokamak operations for all machines present and future. In this phase, the plasma current is decreased as much as possible while attempting to avoid disruptions until confinement is eventually lost. Currently, a large fraction of the disruptions that occur during machine operations occur during this termination phase. For the current generation of machines, disruptions are usually tolerable because they cause little damage. For ITER and future fusion reactors geared towards power generation, disruptions pose a significant concern. At ITER, the specified total number of major disruptions assumed for the design of ITER components is 3000 (equating to 10% of the anticipated full-performance pulses) Sannazzaro et al. [2009]. Special emphasis is placed on avoiding vertical displacement events (VDEs). In an analysis of future tokamak power plants Maris et al. [2023], it was found that "the disruption handling requirements for achieving $<$ \$100/MWh LCOE are extreme". Therefore, identifying operating regimes and control strategies that reduce disruption risks is vital in order to remain within these limits. The physics of disruptions is poorly understood — most progress of physics during disruptions has been made by solving the extended MHD equations with codes like NIMROD Sovinec et al. [2004] in order to explain observed phenomena. What's more, as outlined by De Vries et al. [2011] and Kates-Harbeck et al. [2019], disruption *prediction* is a harder task still due to the disparate disruption causes, so empirical models are the state of the art and are far from perfect Fu et al. [2020b].

In the termination (or rampdown) phase, a number of concurrent changes must be made to the plasma state: the plasma current must be decreased to zero, the auxiliary heating power must be removed (which will precipitate the H to L-mode back transition and a consequent decrease in kinetic energy if it has not already occured), and density must decay. While these changes are occurring, a number of operating and stability limits of the plasma must be respected in order to avoid disruption and consequent damage to the device. Of paramount importance is the maintenance of vertical stability (VS), which is related to the inductance ℓ_i, β_p, and the elongation κ as discussed in de Vries et al. [2017]. The VS limit differs across machines, and at DIII-D benefits from the presence of poloidal field coils located close to the plasma that can respond quickly. The Greenwald density fraction $f_{GW} = \frac{n\pi a^2}{I_p}$ Greenwald et al. [1988] is another key quantity of interest during the rampdown and can cause disruptions when it exceeds 1 as I_p decreases unless density concurrently decreases. The Greenwald limit is an idealized quantity and work such as Giacomin et al. [2022] continues to sharpen our understanding of density limits on different devices under

various conditions. Another potential concern during rampdown is the presence of a variety of MHD instabilities in the plasma, which can cause the plasma to disrupt if they grow too large. Finally, there is a possibility that the plasma undergoes a radiative collapse, where there is no longer sufficient kinetic energy to maintain stability. One possible cause of this is excessive radiation due to impurity accumulation. In this work we explore the optimization of rampdowns using a Bayesian optimization strategy. Before further explaining the contributions of this paper, we first give some background on Bayesian optimization in the context of machine learning.

9.1.1 Related Work

Rampdown Optimization Prior works de Vries et al. [2017], Teplukhina et al. [2017] have addressed the termination phase through optimization and study via models derived from first principles. In this work, we aim in contrast to address this problem through a machine-learning based method for rampdown trajectory design. In de Vries et al. [2017], a large scale analysis of the components of stability of rampdowns was conducted across many of the world's tokamaks; here, the goal was to describe and analyze the key physical phenomena that determine whether a rampdown is successful or disrupts prematurely. The authors identify the relationships between the change in elongation, decrease in power, and decrease in current that underlie the control developments in this work. Another study was conducted in Teplukhina et al. [2017], where numerical optimization was conducted over rampdown trajectories using the RAPTOR simulator. The plasma was successfully ramped down using the design given from the optimization solution on both the TCV and ASDEX Upgrade tokamaks. The key differences between that work and this one are that here there is a data-based approach to rampdown design rather than a simulation-based and there was a large-scale experimental campaign at DIII-D in contrast to study of a pair of shots on different machines. In Barr et al. [2021], the authors develop an emergency shut-down procedure which involved transitioning to a limited topology in order to maintain control down to a safe current level. Our work addresses the nominal rampdown in a similar spirit, but we use a machine-learning based methodology to design the trajectory. Finally, in Fu et al. [2020b], the authors propose a machine-learning-based controller which predicts disruptivity. During the rampdown phase, future disruptivity is predicted in real time. If at any point the disruptivity prediction exceeds a threshold, an Off-Normal Response is triggered that begins a fast rampdown of the plasma current in order to disrupt at a safer level. This feedback control mechanism is complementary to this work—we focus on the design of feedforward trajectories which avoid disruptions in advance while this method reacts in a closed-loop fashion.

Bayesian Optimization There is a large literature focusing on optimization of black box functions Frazier [2018]. The usual assumption in Bayesian Optimization (BO) is that the function of interest $f : \mathcal{A} \to \mathbb{R}$ for some action space $\mathcal{A}$ is drawn from a Gaussian process prior or is bounded in the relevant norm. A typical procedure for BO is to iteratively estimate the function of interest using all observations, use the estimate to compute an *acquisition function* $\alpha : \mathcal{A} \to \mathbb{R}$ that prospectively evaluates the benefit of observing a new datapoint $(a, f(a))$ at some point in the domain, finding the maximizer $a_t = \text{argmax}_{a \in \mathcal{A}} \alpha(a)$, and querying the black box function $f(a_t)$. This process is repeated until the budget for queries is exhausted. Acquisition functions such as upper confidence bound optimization Srinivas et al. [2009], Thompson sampling Russo et al. [2018], and expected improvement Jones et al. [1998] have been developed and analyzed in the preceding decades. These methods have been applied to optimize functions observed in real systems including in the feedforward control of robots Tesch et al. [2013], hyperparameter tuning of machine learning models Kandasamy et al. [2020], and even the design of recipes for chocolate chip cookies Solnik

et al. [2017]. This work uses approximate Bayesian optimization to find a feedforward rampdown trajectory that avoids disruptions.

Many works address a generalization of this setting known as *contextual Bayesian optimization* Char et al. [2019] wherein the domains of f and α are augmented with context x in some context space $\mathcal{X}$ and the goal is to find a policy $\pi : \mathcal{X} \to \mathcal{A}$ that finds optimal actions for each context. This work does not address the contextual setting though we believe it a promising direction for future work.

Learning-based Control in Fusion Typically in learning-based control, an *agent* interacts with an environment by alternately executing actions and receiving observations. The agent then adjusts its decision-making *policy* based on the information it has collected in a way which aims to optimize some objective. In this work we use 'agent' to refer to a generic decision making entity (as in Lu et al. [2023]) though it is most often used to refer to the reinforcement learning setting. In the broader machine learning world, the greatest successes of learning-based control methods have inevitably come when the agent has the ability to collect a large number of samples using its current policy in the ground-truth environment. This has happened most notably in games like Atari Mnih et al. [2015], Go Silver et al. [2016], and Chess Silver et al. [2017] but also plays out in domains where there is a fast and accurate simulator of the system. In Degrave et al. [2022], reinforcement learning was applied to a simulator of current and shape dynamics in order to find a policy which was successfully able to achieve a desired shape on the TCV tokamak. However, shape control is typically achieved using hand-designed and more interpretable controllers Walker et al. [2020] for the same reason that the simulation was accurate: the underlying dynamics of shape control are relatively easy to model. For kinetic control or other more challenging problems, the direct sim-to-real transfer approach may be more difficult. To address this, works like Seo et al. [2021]Char et al. [2023] learn a policy from dynamics models trained on previously logged data collected by the experiments that have been run on KSTAR and DIII-D, respectively. These works fall under the so-called "offline reinforcement learning" setting Levine et al. [2020]; there are inherent challenges associated with this setting such as the fact that the training data was generated by some other policy (fusion scientists running experiments) rather than the agent. Another direction that has been explored is that of finite-set control with a learned model as in Abbate et al. [2021b], where the controller predicted future states for a finite set of actuator settings and chose the one which was predicted to be closest to the target temperature profile. Yet another learning-based control method was deployed in Fu et al. [2020b], where a decrease in injected power was precipitated by a learned disruptivity model. In each of these works, a model was fit to a static dataset and used to make control decisions for the tokamak. As tokamak time is exceedingly limited, it is often infeasible to train controllers in an online fashion (on the machine, with opportunity for the agent to learn from its previous experiences) for fusion applications.

9.1.2 Contributions

In this work, we took advantage of a rare exception to the preceding statement: during the 2022 operations on the DIII-D tokamak we undertook a (relatively) large-scale study of online data-driven rampdown designs. This was made possible by our 'piggyback' experimental design in which we were able to vary the parameters of rampdowns at the end of shots for which the primary experimental data was to be collected during the flat-top phase. After choosing a parameterization for a feedforward control trajectory and a cost function for the desired rampdown behavior, we projected historical DIII-D data onto our action space and trained probabilistic models that predicted

the cost incurred by the rampdown from the action chosen. We first executed the optimal action according to the model several times. Then, we began choosing actions according to a handful of data acquisition functions taken from the Bayesian Optimization literature in order to efficiently explore the design space of rampdowns. After running a few dozen trials in this way, we executed the optimal action according to an updated model several times.

With the caveat that we could not control the initial conditions of the rampdowns in our tests, we found that when compared against the other shots at DIII-D (either those from the same experiments or the broader dataset) our rampdowns were significantly better at reaching low currents prior to disruption with a mean current at disruption 2.5x lower than the DIII-D average. The rampdown designs improved over the course of our experimentation as a general trend and that the final optimum outperformed the initial optimum found, showing that the exploration was helpful in improving our estimate of the optimal rampdown. Our methodology is fairly general and could in principle be used for other feedforward trajectory design problems in plasma control in the future.

In Section 2, we present the rampdown optimization problem setting and discuss the assumptions made in order to simplify our procedure. Section 3 describes the methods used, including data processing, machine learning methods, and our experimental protocol. Section 4 presents the results of our initial modeling exercise as well as a quantitative discussion of our experimental results. In Section 5, we analyze pairs of shots from the test and control set in order to understand what might be driving the observed differences in performance. In Section 6, we conclude by discussing the work in a broader context and give an idea of future directions.

9.2 Method

At a high level, our approach is simple: given an action space $\mathcal{A}$ and a cost function C we aim to find the action $a = \text{argmin}_{a \in \mathcal{A}} C(a)$ by making queries to C using various actions a. In order to do so, we need to find actions which efficiently search the space of possibilities and take into account the values of C obtained by executing various actions. At the ith trial, the action a_i is chosen by approximately maximizing some criterion $\alpha_i(a)$ which we refer to as an *acquisition function* that can be derived from a probabilistic estimate $\hat{C}_i$ of the cost function C. We execute a_i for 1 or more experiments. Then the we run a script to ingest the additional data from the tokamak, process it so that it can be use to fit another machine learning model $\hat{C}_{i+1}$, and generate a new action by optimizing the acquisition function α_{i+1}. By iterating this process for n iterations we aim to discover an action $\hat{a}^* = \text{argmin}_{a \in \mathcal{A}} \hat{C}_n(a)$ that is the "best guess" for the best action design. Though this work focuses on the application of this general loop to the rampdown design problem, it is in principle applicable to a much wider set of problems. There is a diagram of the overall loop in Figure 9.1.

First in Section 9.2.1, we describe the choices for $\mathcal{A}$ and C. Next, in Section 9.2.2, we describe how we acquire and process the data in order to input it into the machine learning model. Then, in Section 9.2.3 we discuss the machine learning methods used to estimate C and the acquisition functions used in the paper. Finally, in Section 9.2.4, we describe the protocol for executing actions on the DIII-D tokamak in a series of piggyback experiments spanning most of 2022.

9.2.1 Problem Setting

We address the rampdown problem as a Bayesian optimization problem where the action space $\mathcal{A}$ is the space of designs for rampdowns and the cost function aims to capture the damage that might be done by a particular rampdown. In making this choice, we explicitly ignore the problem

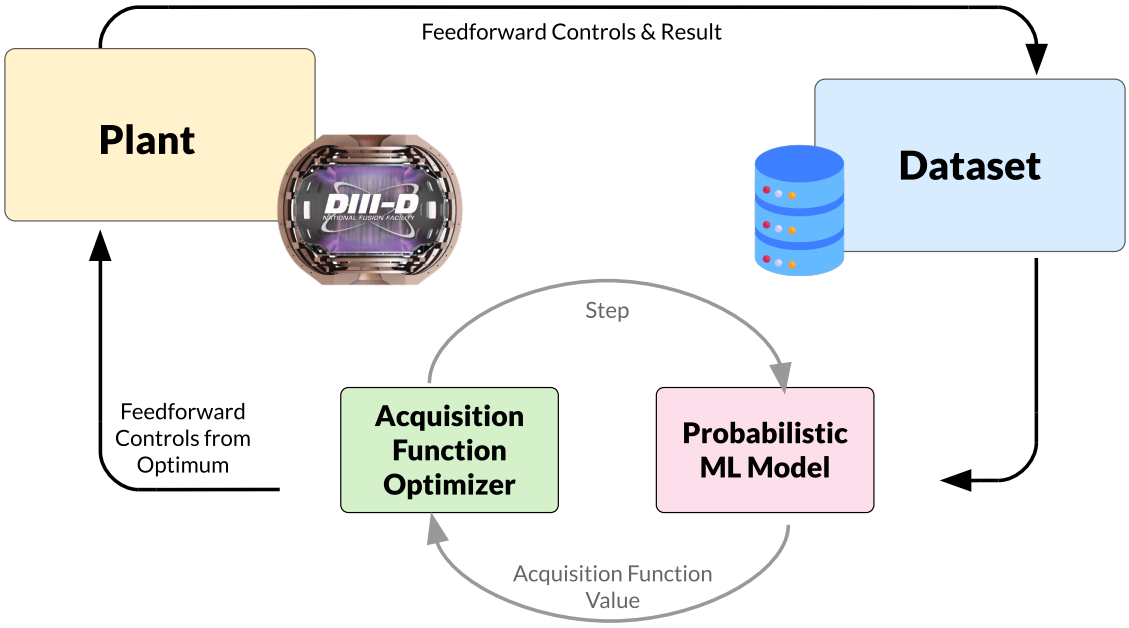

Figure 9.1: Diagram of overall method. Here, the process of executing the actions that optimize the acquisition function and observing their results is shown.

of *context*–that is, we mostly ignore the state of the plasma at the time the rampdown is initiated and search for a specific rampdown design rather than a function mapping the plasma state to a rampdown design. This choice was made for simplicity of modeling and optimization, but we also note that it is not generally possible to know exactly the state of the plasma at the end of the flat-top phase prior to the shot. It also was made in light of the fact that we would be running these experiments in a piggyback capacity after other experiments at DIII-D and therefore we would have very little control or even information about what the state of the plasma would be at the end of the flat-top phase. We also are only addressing *feedforward* control and explicitly leaving the feedback control to existing systems. This choice allows the flexibility to explicitly change the controller behavior based on new information without recompiling the plasma control system. We give a diagram of the overall loop in Figure 9.1.

In order to specify the optimization problem, we must define an action space and an objective function.

Action Space Stemming from prior work Barr et al. [2021], we decided that it was most practical to vary three actuators:

- power injected from the neutral beams (pinj)
- current (Ip)
- elongation of the plasma shape (κ)

As a rampdown design requires all of these to be varied in time, we needed a way to parameterize the trajectory of each of these actuators over time starting from the programmed beginning of the rampdown. As we aimed to conduct piggyback experiments, we explicitly do not consider any changes to shot programming prior to the beginning of the rampdown phase. After considering

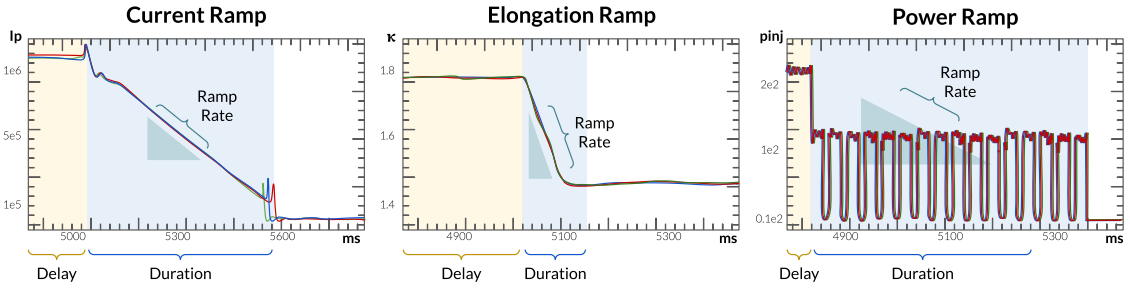

Figure 9.2: Depiction of an example action in our piecewise linear parameterization for current, elongation and injected power. This example is a stylized drawing of shot 188823.

a handful of methods, we decided on a *piecewise linear* (PWL) function for each, which we represented with three parameters: delay (t_0), rate (r), and duration (Δt), which we collectively call θ. We depict these in Figure 9.2. This parameterization also leaves as a free variable the initial value x_0 of each trajectory as these may vary depending on the design of the shot at rampdown. When we execute a PWL action parameterized by θ in a piggyback, we use the value of x_0 from the beginning of the rampdown in the nominal shot trajectory in order to concretely generate the feedforward rampdown trajectory. Then for a particular signal $x_\theta(t)$ starting at time $t = 0$ with initial value x_0, delay t_0, rate r and duration Δt the value is given by

$$x_\theta(t) = \begin{cases} x_0 & t \leq t_0 \\ x_0 - r(t - t_0) & t_0 < t \leq t_0 + \Delta t \\ x_0 - r\Delta t & t_0 + \Delta t < t. \end{cases} \tag{9.1}$$

Since we need a PWL representation of the action for each of our three actuators, our action representation $a \in \mathcal{A}$ has a total of nine parameters $a = [\theta_{\text{pinj}}, \theta_{\text{Ip}}, \theta_\kappa]$ that represents the 3D time series $[x_{\theta_{\text{pinj}}}, x_{\theta_{\text{Ip}}}, x_{\theta_\kappa}]$

Objective Function The primary goal of a rampdown design is that it is safe against disruptions. In particular, disruptions at high levels of current are of concern to operations at tokamaks as well as the uncontrolled release of the various forms of electromagnetic and thermal energy stored in the plasma. In all applications of machine learning, the design of the objective is especially critical. One of the fundamental choices is whether to *shape* the objective function in order to encourage behavior that is thought to lead to outcomes consistent with the ultimate goal. In other words, one might add an instrumental goal to the objective function in the hopes of encouraging behavior that leads to the ultimate goal. This is often done in AI research contexts Hu et al. [2020] by e.g. adding a reward for advancing toward a target state even though the objective is actually only to attain it. In this work, we used an objective function with some *reward shaping* with the ultimate goal of reducing the current at disruption time. Our cost function for an action a as described above is

$$C(a) = \left(\frac{I_p^{t_{cq}}}{10^6 a_{\text{minor}}^{t_{cq}} B} \right)^2 + 10^{-6} W_{\text{MHD}}^{t_{cq}} + |q_{95}^0 - \min_{t>0} q_{95}^t| \tag{9.2}$$

where t_{cq} is the time of disruption (as marked by the time at which the current quench occurs), $x_{\theta_{\text{Ip}}}(t_{cq})$ is the plasma current at the time of disruption, $a_{\text{minor}}^{t_{cq}}$ is the minor radius at the time of disruption, B is the magnetic field, $W_{\text{MHD}}^{t_{cq}}$ is the magnetohydrodynamic energy of the plasma at the time of disruption, q_{95}^{0} is the safety factor 95% of the way to the edge at the beginning of rampdown, and q_{95}^{t} is the same safety factor at times during the rampdown.

The first two terms of the objective are the electrical and magnetohydrodynamic energies of the plasma. These relate to the objective of controlling the plasma to as low of an energy content as possible before a disruption. The third term penalizes any rampdown where the safety factor, q_{95}, drops below its initial value over the course of the rampdown. This term is an example of the reward shaping mentioned earlier: as q_{95} is a key determinant of the stability of the plasma Chen et al. [1984], we encourage our agent to keep it from decreasing. Additionally, this reward is roughly unit-scale, which simplifies the modeling process.

9.2.2 Offline then Online Data Processing

In order to achieve an initial offline estimate of the objective function C, we fit a model to historical data from DIII-D. The data processing consisted of 3 steps: collection, preprocessing, and featurization. In the collection phase, we pulled data about historical DIII-D shots numbered 120000-188814, which run from 2004 to April 2022. from the MDSPlus database as in Abbate et al. [2021b] in 50ms windows. In particular, we collected the information about the action space: target plasma current, injected power from the neutral beams, and elongation as well as for the cost function: safety factor near the edge (q_{95}), minor radius, and MHD energy as computed by EFIT01. For the cost function we also collected the time of disruption (as marked by the current quench) and the plasma current at that time. In order to do so, we used a technique developed in Barr et al. [2021]. The time of current quench is determined by searching for dIp/dt passing a very high threshold (-14MA/s) for reduction in plasma current after which the plasma current never recovers. The beginning of this very fast final Ip drop is the time which the current quench begins, and is used for the current quench time in this work. It then double checks that this drop was not programmed as part of the desired trajectory of the shot. In some cases the plasma recovers after a very large transient event. The code ignores this phenomenon and skips forward to the final (actual) disrupting event.

Finally, we also collected the times at which the rampdowns were programmed to start in order to know at what time the agent could have begun to modify the controls.

In order to make sure that the data were usable for machine learning, it was preprocessed into a form that made it suitable for featurization. Much of this involved removing shots which were unsuitable for use in this study. We removed shots for which any of the following occurred:

- Shot disrupted prior to originally scheduled rampdown or within 50ms of rampdown beginning.
- Shot data in relevant fields contained at least 4 consecutive NaNs (lasting 200ms).
- PWL action projection could not achieve sufficient accuracy (see below).

This left us with 1,173 shots in the original offline dataset from which to perform regression. This data cleaning procedure was conservative and led to shots being excluded which otherwise could have been used. This is important in the experimental section where certain experimental shots did not pass data filtering checks and are therefore excluded as in Figure 9.5. As the experiments progressed all subsequent shots were appended to the dataset including but not limited to those where the rampdown was designed by the model.

Projecting Historical Data to the PWL action space For each shot retrieved from the DIII-D database, actuator data is represented as a time series $\{u_i\}_{i=0}^k$ with u_t the scalar values of the actuator every 50ms. For shots which disrupted during the rampdown the lengths for the action representation were padded as if they had gone to their intended conclusion and disrupted at 50kA. In order to use PWL representation for the action space, it was necessary to find some PWL parameters that approximately corresponded to the time series retrieved for each actuator in each shot. Thus we solved the following optimization problem with the curve fitting library taken from scipy Virtanen et al. [2020], which uses the Trust Region Reflective algorithm Branch et al. [1999]:

$$\begin{aligned} &\text{argmin}_\theta \, ||x_\theta(t) - u_t||_2^2 \\ &\text{subject to } x_0 = u_0. \end{aligned}$$

This optimization problem is a projection of the time series of each actuator onto the space of piecewise linear functions. As these coefficients found were to be used as features for the subsequent machine learning estimation, we discarded the shots for which the projection induced substantial error. Concretely, these were shots for which $\frac{||x_\theta(t)-u_t||_2^2}{||u_t||_2^2} > 0.1$. This was the case for at least one signal in 34% of the data. The dataset $D_n = \{(a_i, c_i)\}_{i\in[n]}$ consisted of the observed actions after all preprocessing as well as the computed costs.

After performing this optimization and filtering, we fit a machine learning model for C and choose an action as described below.

9.2.3 Machine Learning Methods

As discussed in Section 9.1.1, there has been a huge amount of work done in the machine learning community addressing the Bayesian optimization setting. The standard approaches involve uncertainty-aware regression to learn the function C from observations $(a, C(a))$. The specific types of uncertainty required are determined by the acquisition function $\alpha_i(a)$ being used.

Uncertainty-Aware Regression Techniques There is a substantial literature of uncertainty-aware regression techniques Abdar et al. [2021]Psaros et al. [2023]. This work relied on three representations of predictive uncertainty: epistemic uncertainty, aleatoric uncertainty, and posterior sampling. Epistemic uncertainty is the uncertainty in predictions that can be reduced by making observations and performing inference. Aleatoric uncertainty is the irreducible uncertainty in a prediction, often because the system is itself stochastic. Due to the many unobserved features of the tokamak at rampdown time, there is substantial uncertainty that could be reduced given perfect observations but is not captured in the data presented to our model. For modeling purposes, this is treated this as irreducible given our assumptions. Posterior sampling is a bit different – given some approximate prior belief over cost functions $P(C)$, and a set of observations D_n, we can update our beliefs to an approximate posterior $P(C \mid D_n)$ and sample from it. This process can be interpreted as choosing from the set of functions that are consistent with the observations D_n given prior knowledge. Many of the tools developed in BO deal with settings where the black-box function can be fit well by a Gaussian process regression with some kernel. Even when using empirical techniques like maximum marginal likelihood kernel fitting, we were unable to find a kernel that gave reasonable predictive accuracy on our data.

Instead, we used both multilayer perceptrons (MLPs) (following Paria et al. [2022]) implemented in JAX Bradbury et al. [2018] and gradient boosted trees (GBTs) taken from the Catboost package Dorogush et al. [2018] alongside probabilistic variations and ensembles composed of these units.

Acquisition Function	Uncertainty Estimate Required	Functional Form of α
Optimum	None	$-\hat{C}(a)$
Thompson Sampling	Posterior Sampling	$-f(a), f \sim P(C \mid D)$
LCB	Epistemic Uncertainty	$-\hat{C}(a) + \beta\sigma_e(a)$
UCB-LCB	Epistemic & Aleatoric Uncertainty	$-\hat{C}(a) + \beta_1\sigma_e(a) - \beta_2\sigma_a(a)$

Table 9.1: Data acquisition functions and the corresponding uncertainty estimates required.

In their standard forms, neither MLPs nor GBTs estimate uncertainty. However, this can be easily solved for each by having the model output the mean (which we write $\hat{C}(a)$) and standard deviation $\hat{\sigma}(a)$ of the response variable and training them via maximum likelihood Malinin et al. [2021]. We follow Chua et al. [2018] in using $\hat{\sigma}(a)$ for an estimate of the aleatoric uncertainty $\sigma_a(a)$, while the standard deviation of the mean predictions of ensemble members $\hat{C}_i(a)$ can be interpreted as an estimate of the epistemic uncertainty $\sigma_e(a)$. One also can sample a single ensemble member $\hat{C}_i(a)$ as an estimate of a function sampled from the posterior. At every iteration, the model wa s trained using all observations that were part of the dataset. Ensembles consisted of 10 members trained on bootstrapped data sampled with replacement from the training set.

To address the lack of a clear hypothesis over the objective function C, we employed a rotational strategy with different function approximators for each acquisition function. As discussed below, this involved periodically switching between various approximators to align with the requirements of each acquisition function. Concretely, we alternated between MLPs and GBTs and then choose the probabilistic and / or ensemble variant that would provide the uncertainty estimate required by the acquisition function being used (see Table 9.1 for these).

Acquisition Functions In order to acquire data that will facilitate black-box optimization, we rely on probabilistic estimates of the cost function given by Besides choosing the optimum of the estimated cost function, we used Thompson sampling Russo et al. [2018], lower confidence bounds Srinivas et al. [2009], and upper / lower confidence bounds as acquisition functions. Thompson sampling relies on the fact that choosing the optimum of a function sampled from the posterior is equivalent from sampling from the posterior over optima. Lower confidence bounds use an optimistic decision rule to choose points at which the model is either overconfident or correct in its optimism, thereby ruling out parts of the design space which could potentially be good. Upper / lower confidence bounds additionally include a penalty for areas of the design space that are estimated to be highly noisy. As shown in Table 9.1, each of these acquisition functions requires a particular type of uncertainty estimate.

Each of these acquisition functions has shown state-of-the-art performance on some Bayesian optimization problems. Given the uncertainty about which function would best suit this application, we adopted a strategy inspired by Head et al. [2021]: cycling through these functions for each subsequent trial. This approach involves selecting a model which provides the appropriate uncertainty estimate for the specific acquisition function in use at any given trial.

9.2.4 Piggyback Experiments

Throughout the course of 2022, we conducted a campaign of piggyback experiments executing various rampdown designs after the conclusion of the flat top phase. For experiments for which

the session leader (SL) was amenable to our work, the mainline experiment could tolerate some disruptions, and the authors were available, we collected data based on executing actions chosen according to an acquisition criterion with maximally up-to-date data. We also made sure to collect data which used the default rampdown in order to be able to see a useful control set. At DIII-D the nominal rampdown evolves over time as session leaders modify it, but the default strategy has been a decrease in current at 1 MA/s and a complete shutdown of beam power very close to the start of rampdown. The shape is changed to a low-elongation and limited plasma shape around 100ms into the rampdown. There are often small changes in the vicinity of this design. Within an experiment, we would first allow the SL to run with no modifications on our end until they were able to achieve a plasma that lasted until the programmed time of rampdown. For several experiments, this proved difficult and we were unable to run. Once several trials successfully reached the rampdown phase we programmed in an action generated by optimizing an acquisition function and executed it several times. Once it had executed several times, we ran our scripts for ingesting and preprocessing additional data, generated another action, and executed it as well. This process proceeded across several run days in 2022.

9.3 Experiments

9.3.1 Initial Modeling Results

The first step was to fit an estimate of our cost function $\hat{C}$ to the offline dataset D. We initially considered models ranging among linear regression, k nearest neighbors, Gaussian processes, GBTs, and traditional MLPs. Hyperparameter tuning and model selection was conducted using 5-fold cross validation on a training set consisting of 80% of the training data. GBTs and MLPs performed best on cross validation. The MLP with learning rate of 3×10^{-4} worked well as did CatBoost Dorogush et al. [2018] with out-of-the-box settings. GBTs achieved 74% explained variance on the test set and the MLPs were slightly worse at 72%. As can be seen in Figure 9.3, the optimum found in the initial model calls for a moderately aggressive current rampdown (close to 1.8 MA/s) along with modest change in elongation and an aggressive decrease in beam power (80MW/s, an immediate shutdown). The optimum is not as low-cost as the lowest-cost elements due to the fact that the model does not predict extreme values as well. An analysis of the feature importances showed that the ramp rate and duration for current were the most important features used to explain the cost function, a result in line with expectations.

9.3.2 Real-World Performance of Online Bayesian Optimization

After our offline modeling we ran 41 piggyback shots with 16 different synthesized actions across 8 different run days at DIII-D in 2022. The actions were synthesized by optimizing acquisition functions defined over the GBT and MLP models as described in Section 9.2.3. We aimed to answer two questions: (1) did our model synthesize better rampdowns than the default at DIII-D? and (2) could our exploration strategies cause the rampdown designs to improve over time through our trials? In both cases, the answer was yes, with some complexity in the answer to (2).

To address the first question, we determined 2 potential control sets: all rampdowns on DIII-D after 2015 (Wide Control) and all rampdowns taken from the same miniproposal (a document which references a particular experimental allocation in the DIII-D procedures) as our test shots (Same-MP Control). After removing all shots with missing data, the wide control set was 11,047 shots, the Same-MP control set was 25 shots and the test set was 29 shots. The latter two covered a wide range

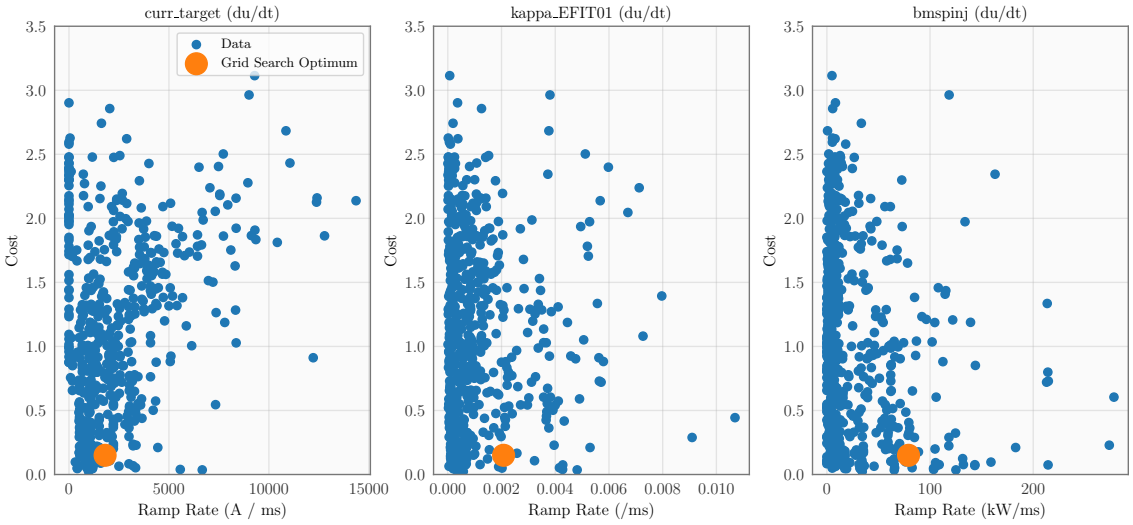

Figure 9.3: Optimization of a learned model using offline data only. This figure depicts a GBT model mapping θ to the cost function to all high-quality examples available prior to our experimental campaign and optimized it via grid search over θ. The plots show the historical observations and the optimum found for 3 components of θ.

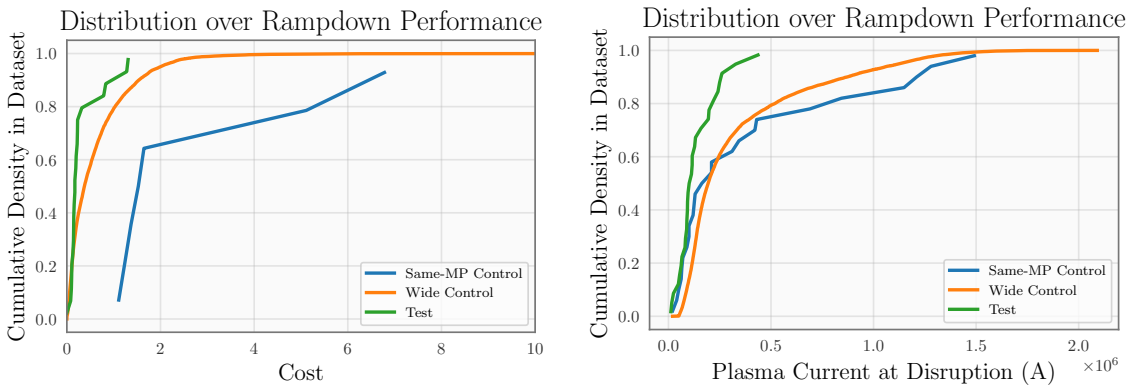

Figure 9.4: Performance of comparison sets of rampdowns on cost and current at disruption. These are empirical cumulative distribution functions, so e.g. the median of the observed samples will be the value on the horizontal axis where the curve crosses 0.5 on the vertical axis.

p-value / Effect Size	Same-MP Control	Wide Control
Current at Disruption	0.027 / -2.77	1.7×10^{-6} / -10.97
Cost	9.9×10^{-5} / -2.72	0.018 / -2.79

Table 9.2: Statistical Tests of Rampdown performance. We used the Mann-Whitney U-test on the disruption currents and costs observed in our experiments to compute the p values shown here. We report the modified Cohen's d for effect sizes.

of plasma conditions, with flat top current ranging between 0.6-1.6 MA in the test set and 0.55-1.8 MA in the Same-MP control set and β_N ranging between 0.6-2.4 in the test set and 0.4-2.2 in the control set. The observed currents at disruption as well as the computed costs for these three sets are shown in Figure 9.4. It is clear by inspection that the shots in the Same-MP control disrupted at a similar distribution of plasma currents to the broader baseline of DIII-D shots. However, in the test shots, our method was able to perform significantly better than was observed in either control set. The mean current at disruption in the test set, 134 kA, was 2.5x smaller than the mean current at disruption in the control set (336 kA) and 2.9x smaller than that of the Same-MP control (389 kA).

After the conclusion of our experimental campaign, we performed statistical tests to assess the possibility that our results were the result of random chance. Ideally, we would have chosen a significance threshold for our statistical results prior to beginning our experimental campaign. However, we did not do so and can only comment on results after the fact. Based on the p-values observed in Table 9.2, which give the probability that differences of at least this magnitude could be observed between samples generated from the same process, it is highly unlikely that our results were generated by random variation. We also compute the modified Cohen's d, a measure of the effect size taken by normalizing the difference between means of two samples by the standard deviations. Typically, 2 denotes a large effect size Sawilowsky [2009] and each of our comparisons attains that threshold.

In particular, it was more clear for the Same-MP control that the cost attained by our method was an improvement compared to the current, while the reverse was true for the wide control. This might have been due to the fact that the Same-MP control contained more recent shots where the rampdown has been optimized more for current but not the cost function, leading to improved performance on the former metric but not the latter relative to the wider control set. It is again important to note that these results come with the caveat that we could not control much of the experimental process due to the piggyback experiment design.

Figure 9.5 shows the performance of the test actions over time. The results are mixed. There is no clear trend in the series of costs (blue) and in fact the last datapoints that are not missing from our time series have fairly high costs compared to the others (some investigation showed us that this was due to a drop in q_{95} immediately at the beginning of the rampdowns). However, the currents at disruption at the end of each rampdown (red) show a clear downward trend as the experiments proceeded. This trend suggests that over time the model learned actions that were more reliably able to bring the shot down to a very low current prior to disruption. One important caveat to note is that these are modest sample sizes and we do not have perfectly controlled conditions so it is possible that our results were the result of statistical fluctuations or unobserved factors.

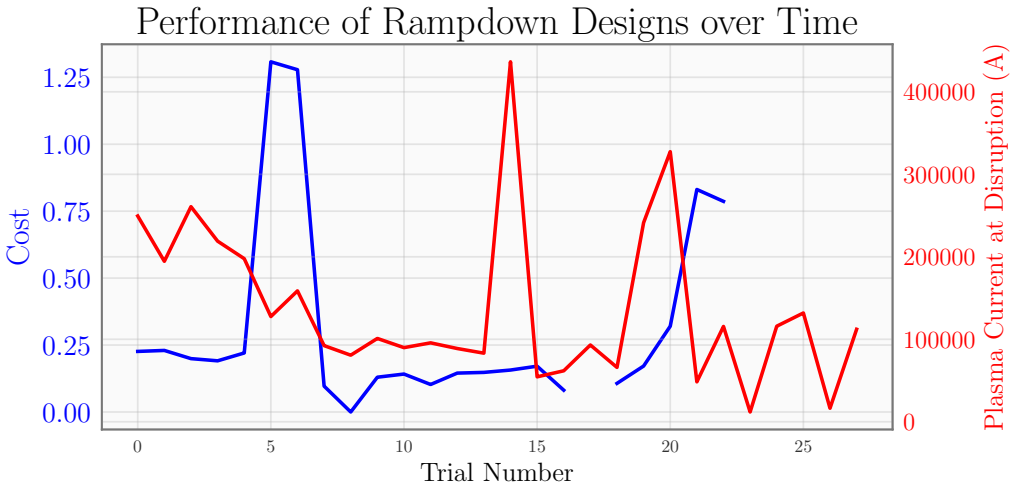

Figure 9.5: Costs and disruption current observed in test group experiments as trials were conducted.

9.4 Analysis

The previous section gave quantitative results of the rampdown experiments. This section presents qualitative results and gesture at how the model may have modified the rampdowns in order to avoid disrupting at high current.

9.4.1 Analysis of Selected Shots

Figure 9.6 shows 3 examples of shots from the test and same-MP control sets in our study that were taken from the same experiment. The left example depicts a pair of shots (a control shot, 192252, and a test shot, 192244) where the control shot disrupted at around 700 kA. Though they started from similar initial conditions, the test shot had a slightly higher ramp rate on Ip and power than the control shot. The control shot experienced a $n = 2$ mode that locks around 5200ms and led to an early disruption. The test shot was able to last longer without such issues until the decreased current caused the Greenwald fraction to increase and cause a disruption at a much safer current level. Notably, the test shot 192244 is the shot with the highest current at disruption in our test set and thus gives a mild failure case of our method.

The second example shows three shots: the control shot 191544 and the test shots 191547 and 191548. It seems likely that 191544 disrupted due to an already-existing $n = 1$ mode that was not present in either test shot 191547 or 191548. This example highlights the difficulties in this piggyback experimental setup: as we're not controlling the conditions of the flat top we cannot reproduce the conditions at the beginning of rampdown and reproduce the challenges encountered. Though these two examples here of very similar rampdowns with slightly slower ramps show good results it is impossible to know whether they would have survived if given the initial conditions of 191544.

The final example shows a combination of exogenous and endogenous factors causing the plasma to safely ramp down. The control shot (192024) disrupts almost immediately upon beginning the rampdown due to what appears to be a locked $n = 1$ mode. The time of the rampdown was slightly moved up by the flat-top operators, and our method output a more gradual IP ramp rate. Together these changes were sufficient to cause the shot to ramp down smoothly to minimal plasma current.

9.4.2 Action Selection across Experimental Campaign

We also inspected the actions output by the acquisition function optimization in order to understand whether there were any patterns observable from the data. Figure 9.7 shows some components of the action space of particular interest: the rates of change suggested by the model for power, current, and elongation. From this, it is clear that the model quickly gives up on varying the elongation of the model. This could be because DIII-D has vertical control coils that are very capable and close to the plasma and therefore the model doesn't find a decrease in elongation necessary to maintain vertical stability. A similar study on a different device with less vertical stability control (ITER, for example) might therefore lead to a rampdown design that relies more heavily on elongation than ours. In many shots, there is an unactuated decrease in elongation as current drops (see Figure 9.6 for examples) and perhaps the model was able to infer the necessary relationships between the control requests and the eventual outcomes. The model explores a range of aggressive I_p ramp rates prior to converging to a healthy but less aggressive ramp rate around 1.2 MA/s. The model also widely explores various settings for the rate of change of power. The model seems to be somewhat confounded by the range of initial power settings at the start of rampdowns, which are part of the context for each shot that the model does not recieve. If the context had been included in the model, we would expect to see a more convergent process for injected power. This is a three-dimensional slice of a nine-dimensional action space so there are six additional axes of exploration not shown here. In the future we hope to extend these methods to include context and in fact to potentially move to closed-loop control in order to respond to developments in real time. We'd also potentially attempt to control density in order to address an unaddressed cause of disruptions (the Greenwald limit) from the current setup.

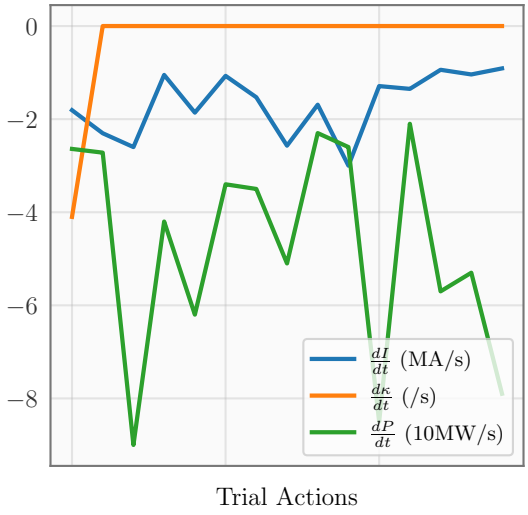

Figure 9.7: Rates of change chosen by models over time for current, elongation, and NBI power.

9.5 Discussion

We conclude with a discussion of piggyback experiments using machine learning more generally. As ML methods are data-hungry and tokamak time is scarce, it is necessary to either use offline data or to find applications like this one where nonstandard opportunities are available to run experiments. We aimed to keep the experimental and machine learning protocol as constant as possible as experiments progressed. As discussed in Section 9.3, reward shaping actually made experiments and analysis more difficult to run. We encourage practitioners trying similar setups in the future to keep things as simple as possible in all aspects, including the data used in decision making and reward computation, the codebase for ingesting data and updating the models, and in the success criteria. It was crucial to keep a fixed experimental protocol on our end in the face of the variation inherent in tokamak operations. ML-driven control of tokamaks is a promising direction

and we hope that our study design is instructive alongside our results. In settings where protocols for rampdowns are not available, as with new machines that don't have these procedures, these methods might prove particularly valuable in exploring possible trajectory designs.

In the previous section, we discussed the 'convergence' of certain components of the actions chosen to some reference values. There is a large literature Srinivas et al. [2009], Chowdhury and Gopalan [2017], Frazier [2018], Russo et al. [2018] discussing the rates of convergence of the various data acquisition methods used here in formal settings. Many of them can be tuned in order to achieve a desired confidence bound or even to work backward from a known horizon of experimentation. Throughout our study, we were highly uncertain about the number of trials we would be able to execute due to scheduling and machine operation questions that are familiar to all tokamak experimentalists. This made the question of how to tune our acquisition strategy slightly more difficult.

This work aimed to find a rampdown trajectory that improved the existing one at DIII-D by trial and error using strategies from black-box optimization. In experiments over the course of 2022 we conducted trial rampdowns as piggybacks and updated our model with new observations as they came in. Our rampdowns were able to bring the plasma current down to an average value 2.5x smaller than is typical at DIII-D and, based on our statistical tests, our results are unlikely to be due to chance. However, as we could not control the plasma state at the beginning of rampdown it is difficult to decouple disruptions due to physical phenomena present at this time from disruptions due to poor control.

One exciting direction for future research is in attempting a similar experimental campaign without the benefit of the existing DIII-D database but perhaps with the use of first-principles driven simulators such as those being used for the development of ITER Kessel et al. [2007] and SPARC Rodriguez-Fernandez et al. [2020]. Although the specific rampdown design we developed in this work is unlikely to transfer to those devices, it is possible that a procedure like this one could be used to support the commissioning of rampdowns at each. The efficient and robust commissioning of rampdowns for these devices will be an important part of their successful operation within disruption constraints. Simulating those conditions could help us understand the effectiveness of these methods in a more realistic setting. With all this considered, we see ample opportunities for additional work of this nature, both in continuing to optimize and better understand rampdowns on DIII-D and at other machines and in applying active machine learning methods to other control tasks in fusion research.

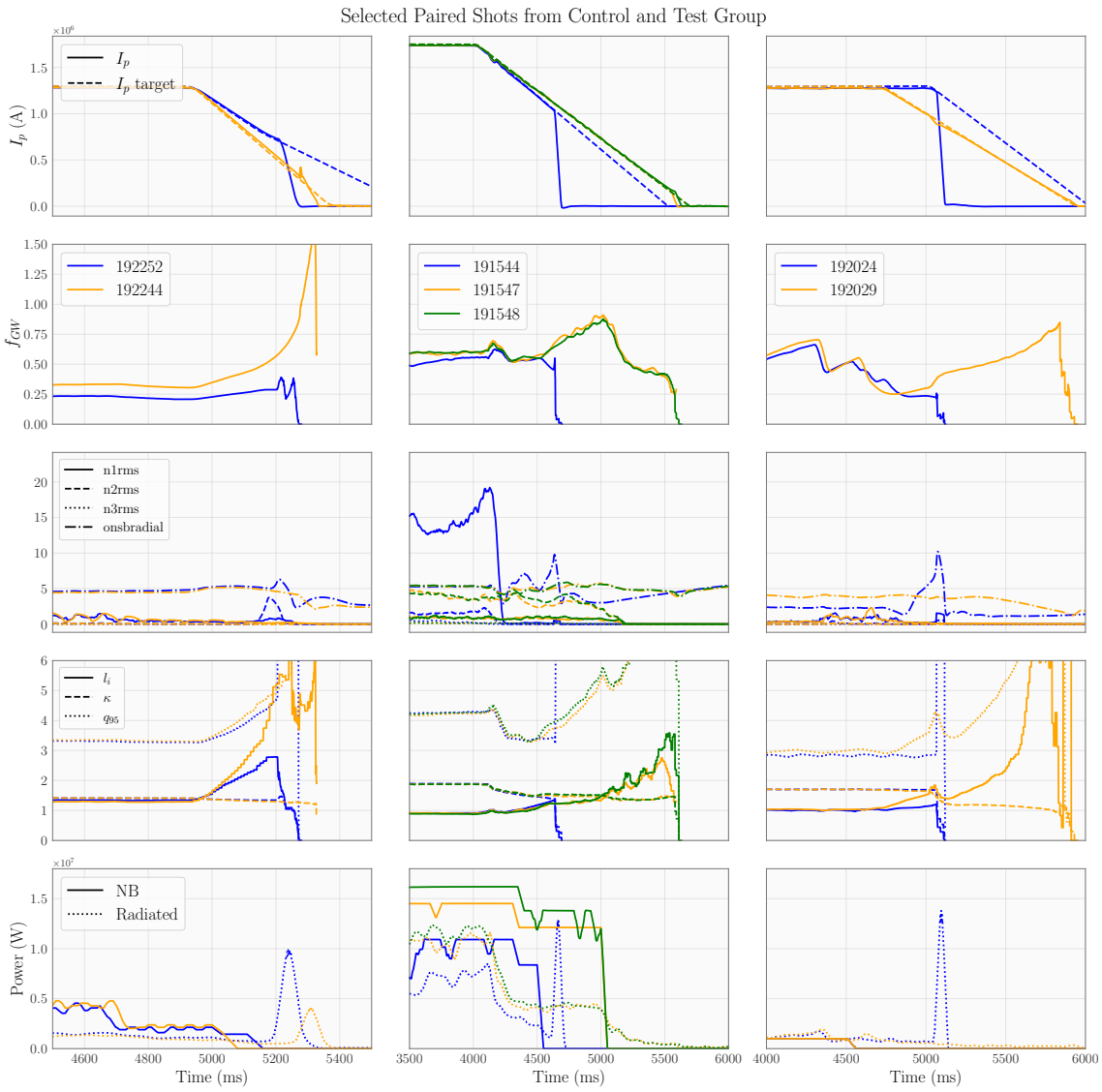

Figure 9.6: Selected paired shots from the test and control sets. Orange is test group and blue/green are control. Shot numbers are given in the second row.

10 | Future prospects for applications of AI to Fusion

Over the past few years, we have seen initial applications of learning-based control to robotics as discussed in section 8.3. Though a number of works have achieved results worth noting, in general plasma control has not seen substantial utilization of these learning-based methods by fusion practitioners and plasma physicists. In this part, I'll give context on some of the underlying reasons for this through a critical view of my own experiences in the hope that future researchers will benefit from frank discussion of the challenges involved. In contrast to chapter 8, here I'll focus on the practical aspects of fusion research. As an illustrative example, I'll give some color on an experiment I ran in 2023 on DIII-D which was largely unsuccessful. Next, I'll discuss what I think it will take to really 'move the needle' in fusion; what are the key problems that are keeping us from a future powered by magnetically confined fusion? Finally, I'll synthesize these perspectives and give some commentary on where AI (taken in a more general spirit) can be helpful to this process. Here, I'll discuss recent work we conducted as a first step to taking advantage of advances in NLP in order to aid tokamak operators.

10.1 Challenges in applying AI to fusion

In this section I'll give a description of the process of running an experiment at DIII-D and the constraints that this process and the general operation of the machine impose on the application of machine learning. Though things vary somewhat between devices, DIII-D serves as a good

Figure 10.1: The DIII-D tokamak (left) and one of its 4 neutral beams (right) as of July 2022.

example. At DIII-D, experiments are allocated on multi-year planning cycles. This is required due to a number of constraints on materials, maintenance, human capital, and travel. The experiments range from a half day to 2 days of tokamak time. Typically a 'day' of tokamak time is 8 hours of operations, where the tokamak fires for ~ 5 seconds and then takes about ten minutes until it is ready to fire again. Data from the previous shot arrives intermittently over the first couple minutes into a computer system [Stillerman et al., 1997] from which humans or scripts can pull it. A typical experiment builds atop a 'reference shot' taken from a prior experiment. This is useful as there are such a large number of options in tokamak configuration and trajectory design that could be explored that scientists usually start from a known-good initialization and make incremental changes. Often it takes several tries to reasonably reproduce the reference shot, a phenomenon that is obvious in light of fig. 10.1, which highlights how many components are part of this system. Each of these can fail; in fact, hardware failures happen during essentially every experiment.

In order to run a learned policy consisting of a machine learning model, the model must first be integrated into the PCS [Humphreys et al., 2005]. At DIII-D, the PCS is written in a subset of C. To solve the problem of compiling policies to this subset, Conlin et al. [2021] built the Keras2C framework, which compiles a subset of Keras down to C that can be included in the PCS. However, this process requires that the parameters of the network are given at compile time. Though it is in principle possible to fix the network structure and load parameters from a file at runtime, to my knowledge no experiments of this kind have been conducted on a tokamak. Experiments of this kind would allow for closed-loop controllers that learn from each trial.

The PCS is configured between shots by operators to the specification of the experiment's session leader. This primarily consists of operators adjusting waveforms corresponding to the settings of lower-level controllers that handle the neutral beams, the plasma shape, the current, and several other quantities as was done in chapter 9, as well as adjusting gains of lower-level controllers for e.g. the 3D coils. When running a machine learning experiment, substantial online debugging occurs as there are often differences between the software testing and deployment settings. Tokamak dynamics are also nonstationary: the wall conditions vary and affect plasma conditions and actuators and diagnostics often fail. As DIII-D has been operating for decades now, there is a substantial amount of logged data that can be used to train controllers via offline reinforcement learning methods. Typical works that attempt this [Char et al., 2023, Seo et al., 2022] find about 15,000 usable shots from the DIII-D database. However, the device has changed over the years as has the experimental program. So, both the dynamical system and the set of behavior policies have drifted over time and the latter may not correspond to the desired policy for a particular experimental goal. To summarize, the experimental setup for testing learned policies on tokamaks is a far cry from the cheap, fast, deterministic games where RL has been most successful.

10.2 Gating challenges to fusion power

It is often nonobvious to those new to fusion what challenges to fusion are 'gating', or in other words, what the key problems are to solve so that we can have net energy from fusion and eventually power plants. In this section, I'll give a set of problems that stand out to me as particularly important to the global research program. As we discussed in section 8.1, fusion power per unit volume scales with β and the magnetic field strength. The magnet coils in fusion plants will be superconducting electromagnets. Since superconductors typically come with limits on temperature, current, and field strength [Tinkham, 2004], any research that improves these would alter the design space of reactors we could build. Similarly for the materials that collect neutrons, provide structural stability, and

Figure 10.2: The control room at DIII-D.

handle high levels of heat flux: progress in materials science that leads to more performant materials with which to build the neutron blanket, the reactor structure, and the divertor will allow us to relax constraints on the reactor. As an example, the limits of the divertor dictate the goals of the divertor control system which will help prevent it from melting; as the divertor is made more robust, the difficulty of divertor control decreases.

There is a persistent search for new modes of tokamak operation; in fact, the H-mode which has been widely studied in the past decades of fusion research [Wagner et al., 1982] was discovered experimentally. Subsequently, other advanced modes of plasma confinement have been found. Efficiently searching the space of these scenarios and using simulation, data, and device resources effectively to find good plasma regimes is another key challenge in fusion.

Given a particular scenario, the key challenge in fusion is robustly achieving and maintaining the plasma at those conditions. Here, the most critical problems are in fault response and disruption avoidance [Hollmann et al., 2015]. Work by Boyer et al. [2021] and others has begun to address these issues, but substantial further study is needed. Additionally, integrated control systems which globally allocate actuation resources to objectives as needed will need to be developed for effective tokamak control.

10.3 How can AI make a real impact on this problem?

Today, the international community of fusion researchers continues to address the existing issues with fusion. As a machine learning researcher, I believe it is important to approach the problem space with humility and not assume ML is a hammer that we should hit it with but rather think about why an AI solution would be better than a human-designed one. Here, I'll discuss four key ideas that could help specify good areas to apply AI systems to fusion.

The simplest is in **higher-frequency problems**, such as the online actuation of 3D coils at megahertz frequencies to control Alfvén eigenmodes [Garcia-Munoz et al., 2019]. When data comes in at a much higher frequency it is more difficult to hand-design filtering and control techniques but

there is far more data with which to train ML systems. Addressing these with efficient architectures for sequence modeling and prediction may be an area where ML can provide a lift in performance.

Another key area where ML could be helpful is in **search problems**. The work described in chapter 9 was inspired by the revelation that researchers were manually searching through the space of rampdown parameters in order to find good ones. Machine learning is often useful in these areas due to its explicit representations of uncertainty as well as the principled strategies for searching for the optima of a probabilistic estimate. We found initial success here in the rampdown setting, but hope that this idea can be applied to further objectives in e.g. scenario design.

10.3.1 LLMs as operational copilots and research assistants

As mentioned in section 8.1, there is substantial cognitive load on operators of nuclear reactors due to the tight timeline of each shot, the large number of incoming data sources, and dispersed knowledge of the system and the relevant physics. There are rich sources of textual information about this system: shot log notes from operators, proposals for experiments, papers, textbooks, PCS source code, and conference proceedings. It is possible that an LLM with access to this information may prove helpful to operators in answering their questions.

In Mehta et al. [2023], we made initial steps towards this goal by implementing a retrieval-augmented generation system [Chen et al., 2017a, Lewis et al., 2020] that can answer questions about the tokamak and its operations grounded in the shot log notes at the DIII-D and C-MOD [Hutchinson et al., 1994] tokamaks. We made this system available through the Discord server DIII-D operators use to communicate during experiments and hope that our system can be useful to operators in the upcoming campaigns. However, there is substantial opportunity to extend our work.

While leveraging shot logs is a crucial first step for making a tokamak co-pilot, we would also like to note that there are other data sources that one could incorporate to enhance the co-pilot beyond this initial version. For example, we could also enable the source code in the Plasma Control System (PCS) to be queried. Systems such as GitHub Copilot [Chen et al., 2021] have already shown that pairing LLMs with code bases results in powerful tools. In our setting, access to source code could enable operators to quickly diagnose and debug software problems during experiment sessions should they arise.

We also could use more advanced schemes for combining LLM generation and information retrieval. These include having the LLM generate SQL for queries, a map-reduce style architecture for summarizing larger context, and self-introspection in order to determine whether an answer has been found. When deployed on the Discord, 3 out of 7 questions inquired about quantitative information that could have been answered by a reasonable SQL query.

Perhaps the richest source of information not being considered at the moment is the diagnostic data measured for each shot. This data set includes rich, high-fidelity information about the plasma (e.g. temperature, density, and rotation) Stillerman et al. [1997], and has already been leveraged by prior works to predict the evolution [Char et al., 2023][Seo et al., 2023] and stability [Rea et al., 2019][Fu et al., 2020b]. At the same time, previous works, such as CLIP [Radford et al., 2021], have achieved impressive results by learning a mapping between the latent encodings for different modalities of data. Building on both areas of works, we hope to incorporate diagnostic information into our current system. Not only do we expect this to provide additional information and lead to more intelligent answers, but doing so could potentially lead to valuable new use cases for our system such as automatic shot log generation or predicting the evolution of the plasma from text descriptions for scenario development. Any of these could prove valuable to operators and scientists under the extreme attention demands of a tokamak experimental session.

10.3.2 Actually Offline RL

As discussed above, offline RL has gained significant attention in recent years [Levine et al., 2020]. However, offline RL research has broadly been been focused [Kumar et al., 2020, Kostrikov et al., 2021] on the D4RL [Fu et al., 2020a] and RL-Unplugged [Gulcehre et al., 2020] datasets. These datasets consist of logged transition data for training and MDPs for test-time evaluation. Inevitably, researchers will evaluate their method on the MDP and then tweak it for better performance. This is simply not possible in fusion, where you may only get a handful of trials in one afternoon a year. Though offline RL is in some sense the 'correct' tool to use to learn controllers for fusion systems, the field as currently conducted is not addressing the problem its full difficulty. I hope to see held-out dynamical systems and 'contests' where policies are submitted for evaluation in order to begin to address the correct problem and one day perhaps learn policies from logged fusion data.

11 | Conclusion

In this thesis, we embarked on a journey through the landscape of sample-efficient reinforcement learning, the integration of prior knowledge in the identification of dynamical systems, and the novel application of these techniques to plasma control for nuclear fusion. Our work represents progress on understanding how agents ought to prospectively evaluate data, a demonstration of the value of machine learning for search problems in fusion, and contributes to the synthesis of first principles and data-driven thinking.

The methodologies and insights presented in part I augment our ability to build agents which are able to reason about what they *know* and what they *need to know*. Through information theory, decision theory, and a frequentist confidence-bound framework, we were able to give several perspectives on how agents should value data. Furthermore, by extending one method to the comparative feedback setting, we were able to apply these idea to the alignment of LLMs, a problem of maximal salience to our field today.

In part II, we demonstrated the power of leveraging existing knowledge about systems to optimize data usage. This approach significantly reduces the data requirements for identifying and controlling complex systems. The line of work that has grown in this area since this work was published has the potential to greatly improve our ability to model and control complex physical systems.

The application of these methodologies to plasma control in nuclear fusion, as detailed in part III, exemplifies the practical impact of our research. Our successful experiments using Bayesian Optimization for plasma current ramp-down highlight the potential of machine learning in enhancing the efficiency and safety of nuclear fusion processes.

I'd like to conclude by acknowledging that in both fields we address in this thesis we have a very long way to go. In both reinforcement learning and nuclear fusion we are in a critical and exciting period of progress and investment; it will take focused effort from smart and hardworking people to continue to move forward. I hope that this work gives useful tools and knowledge as we continue to press ahead.

Appendices

A | Appendix for chapter 2

A.1 Related Work

Transition Query Reinforcement Learning In the standard online RL setting, one assumes data $D = \{(s_i, a_i, s'_i)\}_{i\in[n]}$ must be collected in length-H trajectories (*rollouts*) where the initial state $s_0 \sim p_0$, 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 the process repeats. Kearns et al. [2002] introduced the setting where the agent collects data by sequentially sampling transitions 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. The agent sequentially acquires data (s_i, a_i, s'_i) in arbitrary order by querying a state action pair (s_i, a_i) from $\mathcal{S} \times \mathcal{A}$ and recieving a sample $s'_i \sim T(s, a)$ from the black-box transition function T [Kearns et al., 2002, Kakade, 2003, Azar et al., 2013]. The goal in both settings is to find a policy which optimizes the objective in Equation (2.4). This setting is relevant in a variety of real-world applications where there is a simulator of the transition model available. In particular, we 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].

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]. 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. The improvement shown in finite cases suggests that there could be similar reductions available in a continuous setting. To our knowledge there do not exist works specifically solving the TQRL setting for continuous MDPs. In this work, we give an algorithm specifically designed for this setting, which 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., 2017b], Thompson sampling (TS) [Osband et al., 2016a], 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. Although UCB, TS, and entropy bonuses all try to adapt the exploration strategy to the problem, they all tackle which action to take from a predetermined state and don't explicitly consider which states would be best to acquire data from.

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 and 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. However, these methods don't scale to continuous problems and don't provide a way to choose states to query. These methods were further extended with the development of knowledge gradient policies [Ryzhov et al., 2019, Ryzhov and Powell, 2011], which approximate the value function of the Bayes-adaptive MDP, and information-directed sampling (IDS) [Russo and 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. In a similar spirit, Arumugam and Van Roy [2021] provide a further generalization of IDS which can also be applied to RL. One recent work very close to ours is Lindner et al. [2021], which actively queries an expensive reward function (instead of dynamics as in this work) to learn a Bayesian model of reward. Another very relevant recent paper [Ball et al., 2020] gives an acquisition strategy in policy space that iteratively trains a data-collection policy in the model that trades off exploration against exploitation using methods from active learning. Achterhold and Stueckler [2021] use techniques from BOED to efficiently calibrate a Neural Process representation of a distribution of dynamics to a particular instance, but this calibration doesn't include information about the task. A tutorial on Bayesian RL 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], a density estimate [Bellemare et al., 2016], or predictive entropy [Buisson-Fenet et al., 2020]. However, these methods all assume that there is no reward function and are inefficient for the setting of this paper, 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 and Verdinelli, 1995]. The BAX framework gives a general procedure for sampling points which are informative about the future execution of an algorithm.

Control Problem	Pendulum	Cartpole	Lava Path	Reacher	Beta Tracking
Budget b	200	300	100	1500	300
# of points for optimization k	1000	1000	1000	1000	1000
# of posterior function samples n	15	15	15	15	15

Table A.1: BARL hyperparameters used for each control problem.

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 and 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 and Deisenroth, 2018], which gives a principled probabilistic model-predictive control algorithm for GPs.

A.2 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 A.1.

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 and 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.2.1 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 $\pi_{\hat{T}}$ to collect data as in the standard RL setting. We also compare against PILCO [Deisenroth and 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 which 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.

A.2.2 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 [Char et al., 2019, Mehta et al., 2021] where the controller must maintain a fixed normalized plasma pressure using as GT dynamics a model learned similarly to Abbate et al. [2021a]. The lava path is intended to test stability and exploration of algorithms. The goal is to reach a fixed goal state from a narrow uniform distribution over start states. As shown in Figure 2.1b, the state space contains a 'lava' region which gives large negative rewards for every timestep. When not 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 there is a narrow path through the lava, we want to explore a policy which crosses efficiently and safely. Agents who fail to find this solution will be forced to go around, incurring penalties.

A.2.3 Runtime Details

Based on these choices and the MPC hyperparameters below in Section A.3, each of these problems results in a varying runtime for the BARL algorithm. In Table A.2, 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.

A.3 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 1.4, which uses several 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

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 A.2: Runtime in seconds for the phases of the BARL algorithm on all problems when run on the author's 24-core CPU machines. 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.

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 A.3: Hyperparameters used for optimization in MPC procedure for control problems.

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 A.3.

A.3.1 Robustness of EIG_{τ^*} to a suboptimal controller

In order to compute EIG_{τ^*} 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 τ^* on posterior samples of the transition function.

In order to understand the sensitivity of EIG_{τ^*} 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 suboptimal policy generating samples of τ^*.

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.

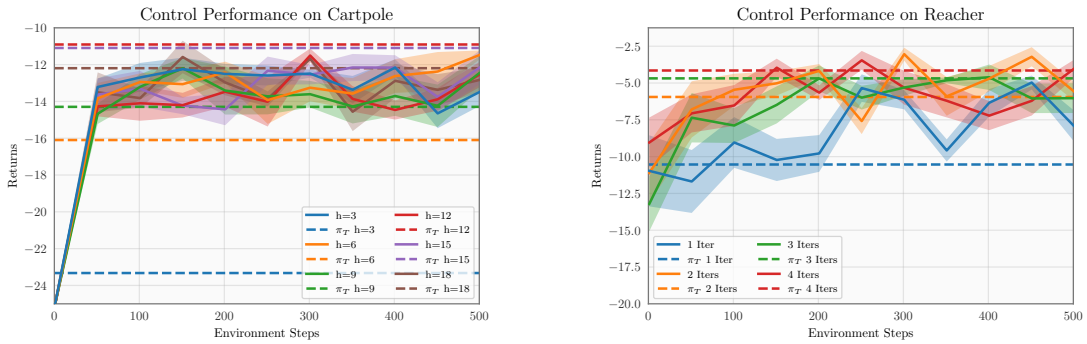

Figure A.1: Performance of BARL when MPC budget for posterior function samples is varied while MPC test time budget is held constant. The error regions are the standard error of the return seen across 5 trials of the policy. The dashed lines are the performances that MPC with the equivalent hyperparameters achieves if executed at test time given the ground truth dynamics.

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 A.1 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.

A.4 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 distribution over start states. As shown in Figure 2.1b, 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. [2021a]. 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.

A.5 Implementation Details for TIP

A.5.1 Derivation of Computational Cost

In this section, we derive the computational complexity of the TIP algorithm. For simplicity, we focus on a single TIP planning iteration as might be done at each replanning in closed-loop control or at the start of a trial in open-loop control. In order to keep the analysis general, we assume that the chosen planning algorithm requires p accesses to the model where h actions are sequentially executed, giving ph total queries per planner execution. We also assume that the numbers of inducing points and basis functions used in our GP posterior function sampling are constant as are the Monte Carlo hyperparameters m, n.

The TIP algorithm consists of the following major operations:

- Sample T_i' for $j \in [m]$: $O(nmN)$ total cost from sampling algorithm, where N is the dataset size.
- Sample τ_{ij}^* for $i \in [n], j \in [m]$: $phHnm$ total cost from running the planner (ph posterior function queries) H times for each sampled τ^*, where H is the MDP horizon.
- Compute Cholesky decomposition for each $D \cup \tau_{ij}^*$. This takes a total of $O(nm(N+H)^3)$ operations as the augmented dataset is of size $N+H$ and Cholesky decompositions are $O(d^3)$ in the matrix size d.
- Compute posterior covariance $\Sigma^{S'} \mid D \cup \tau_{ij}$ for all τ_{ij}. This involves several matrix operations but the most computationally intensive is solving h triangular systems of size $(N+H) \times (N+$

Control Problem	Pendulum	Cartpole	β Tracking	β + Rotation	Reacher
Sample τ^* mn times	24	31	7	25	130
Plan actions that minimize C_{τ^*}	16	15	15	50	295
Total for TIP Iteration	40	46	22	75	425
Evaluation for one episode	5-20	2-10	2-5	3 - 18	100-500

Table A.4: Runtime in seconds for the phases of the TIP algorithm on all problems when run on the authors' CPU machines. 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.

H), which each take $O((N+H)^2)$ time. So the total computation here is $O(pnmh(N+H)^2)$.

- Compute determinants of covariance matrices for each of p queries and nm augmented datasets $D \cup \tau_{ij}$. Each of these operations is over a matrix of size $h \times h$ and therefore costs $O(h^3)$. So the total cost is $O(pnmh^3)$.

Summing these costs gives $O(nmN+phHnm+nm(N+H)^3+pnmh(N+H)^2+pnmh^3)$. Clearly the third term dominates the first, the fourth dominates the second, and since $H > h$, the fourth dominates the fifth. So, the computational cost can be summarized as $O\left(nm\left((N+H)^3+ph(N+H)^2\right)\right)$.

A.5.2 Wall Times

Though TIP and oTIP are designed for applications where samples are expensive and computation is relatively inexpensive, we present in this section data on the running time of these methods. We ran all experiments on a shared research cluster available to us on large machines with hundreds of GB of memory and between 24 and 88 CPU cores. In general our implementation did not make use of more than 20 CPU cores concurrently. In Table A.4, we give the running time of the phases of the TIP algorithm. We note that the bulk of the computation in the planning procedure actually goes towards the just-in-time compilation of the JAX code that computes the cost function C_{τ^*} on sampled future trajectories. In order to allow for this compilation cost, we modified the iCEM algorithm from [Pinneri et al., 2020] to take fixed batch sizes as the compilation (e.g. for the β tracking problem) takes approximately 90% of the time required for planning. Unfortunately this compilation process must be repeated at every iteration due to the limitations of the JAX compiler. We believe that a similarly JIT-compiled implementation of the planning algorithm for sampling τ^* on posterior samples could lead to a substantial speedup and a more flexible compiler could do more still.

A.5.3 GP Model Details

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 and 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 β + Rotation and Reacher, for which we chose a random subset of $\cup_i \cup_j \tau^*_{ij}$ (the posterior samples of the optimal trajectory) since the space of samples is 10-dimensional and

Control Problem	Pendulum	Cartpole	β Tracking	β + Rotation	Reacher
Number of samples	25	30	25	50	100
Number of elites	3	6	3	8	15
Planning horizon	20	15	5	5	15
Number of iCEM iterations	3	5	3	5	5
Replanning Period	6	1	2	1	1

Table A.5: Hyperparameters used for optimization in MPC procedure for closed-loop control problems.

Control Problem	Nonlinear Gain 1	Nonlinear Gain 2	Lava Path
Number of samples	50	50	25
Number of elites	6	6	4
Planning horizon	10	10	20
Number of iCEM iterations	6	8	6

Table A.6: Hyperparameters used for optimization in MPC procedure for open-loop control problems.

uniform random sampling will not get good coverage of interesting regions of the state space.

A.5.4 Cost Function Details

We set $n = 15$ and $m = 1$ for our Monte Carlo estimate of the cost function for each problem.

A.5.5 Details on Planning Method

As mentioned in the main text, we use the iCEM method from Pinneri et al. [2020] with one major modification: a fixed sample batch size. This is in order to take advantage of the JIT compilation features of JAX and avoid recompiling code for each new batch size.

In Tables A.5 and A.6, we present the hyperparameters used for the planning algorithm across each problem. The same hyperparameters were used for the TIP, MPC, EIG_T, DIP, sDIP, and sTIP methods. 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.

A.6 Description of Comparison Methods

We compare against 14 different methods across open and closed-loop problems. Of these, 7 used the same model and planning algorithm (including hyperparameters) as TIP and oTIP. **DIP** and **oDIP** use the cost function $C(\tau) = -\mathbb{H}\left[T(S') \mid D\right]$ and **sDIP** (summed DIP) uses the cost function $C(\tau) = -\sum_{i=0}^{h} \mathbb{H}\left[T(s_i, a_i) \mid D\right]$. These are all pure exploration methods, but DIP and oDIP are more sophisticated in that they plan for future observations with a large amount of *joint* information as opposed to sTIP which sums the individual information expected at each timestep. oDIP is simply

the open loop variant of DIP. **EIG**$_T$ uses the same objective as sDIP but operates in the TQRL setting, querying points that approximately maximize the predictive entropy of the dynamics model. **BARL** similarly operates in the TQRL setting but uses the EIG_{τ^*} acquisition function from Mehta et al. [2022c]. We use the authors' implementation of that work for comparison. **MPC** uses C_g from (2.7) and plans to directly maximize expected rewards. This method can be seen as quite similar to Kamthe and Deisenroth [2018] and a close cousin of Deisenroth and Rasmussen [2011] in that it optimizes the same objective with a similar model. **oMPC** is simply the open loop variant of MPC.

Besides these methods which directly compare cost functions, we include 8 additional baselines from published work. **PETS** is a method given in Chua et al. [2018] which uses a similar cross-entropy based planner and a probabilistic ensemble of neural networks for an uncertainty-aware estimate of the dynamics. PETS also plans to minimize C_g. **HUCRL** [Curi et al., 2020] learns a policy via backpropagation through time using a hallucinated perturbation to the dynamics that maximizes discounted rewards subject to the one-step confidence interval of the dynamics. HUCRL also uses a probabilistic ensemble of neural networks. Using the same implementation we also tested Thompson Sampling (**TS**), which acts optimally according to a network drawn from the posterior over models, and **BPTT** which plans to minimize C_g using a neural network policy and backpropagation through time. BPTT can also be viewed as a cousin of PILCO [Deisenroth and Rasmussen, 2011] as it attempts to take stochastic gradients of the expected cost. We also compare against **SAC** [Haarnoja et al., 2018], **TD3** [Fujimoto et al., 2018], and **PPO** [Schulman et al., 2017]. SAC uses entropy bonuses to approximate Boltzmann exploration in an actor-critic framework. TD3 and PPO include various tricks for stable learning and add Ornstein-Uhlenbeck noise in order to explore.

For our FEEF implementation, we took hyperparameters from the most similar comparison environments in that paper and used them for our results. We tried several values for 'expl_weight' in order to se whether we were inadequately balancing exploration and exploitation. Ultimately we saw an 'expl_weight' of 0.1 was the best value.

We used the author's implementation of RHC. RHC makes strong assumptions on the form of the reward function by assuming that all problems are regulation problems where the goal is to drive the system to a given state and keep it there (with some cost for actuation). We were able to pass the targets for all of our problems (which may change between episodes) to the RHC controller. We did a light hyperparameter search tuning the number of random Fourier features used in the Bayesian linear model in this method. Ultimately we were disappointed in the performance of RHC when applied to our problems. We believe that this might be due to its undirected uncertainty sampling objective and relatively constrained model of environment dynamics.

A.7 Description of Control Problems

A.7.1 Plasma Control Problems

The plasma control problems are based on controlling a tokamak, a toroidally shaped device for confining a thermonuclear plasma using magnetic fields. Achieving net positive energy from fusion requires confining a plasma at high enough temperature and density long enough for hydrogen isotopes to collide and fuse. However, as the temperature and density are increased, a wide variety of instabilities can occur which degrade confinement, leading to a loss of energy. Full physics simulation of tokamak plasmas requires 10s-1000s of CPU hours to simulate a single trajectory, and often require hand tuning of different parameters to achieve accurate results. Following the work of

Abbate et al. [2021b], each of our plasma control problems used neural networks trained on data as the ground truth dynamics models. We used the MDSPlus tool [Stillerman et al., 1997] to fetch historical discharges from the DIII-D tokamak in San Diego [Fenstermacher et al., 2022]. In total, we trained our models on 1,479 historical discharges. The data was pre-processed following the procedure outlined in Abbate et al. [2021b]. We describe how each environment was constructed in more detail below.

β Tracking In this environment the goal is to adjust the total injected power (PINJ) of the neutral beams so that the normalized plasma pressure, β_N (defined as the ratio of thermal energy in the plasma to energy in the confining magnetic fields), reaches a target value of 2%. Reliably controlling plasmas to sustain high performance is a major goal of research efforts for fusion energy, so even this simple scenario is of interest. The ground-truth dynamics model takes in the current β_N and PINJ, the β_N and PINJ at some Δt time in the past, and the PINJ at some Δt time in the future (we assume that we have complete control over the values of PINJ at all times). Given these inputs, the model was trained to output what β_N will be Δt time into the future. In total, the state space is 4D and the action space is 1D. For this environment, we set $\Delta t = 200ms$, and we specify the reward function to be the negative absolute difference between the next β_N and the target $\beta_N = 2\%$.

β + Rotation Tracking This environment is a more complicated version of the β tracking environment in several ways. First of all, the controller now must simultaneously track both β_N and the core toroidal rotation of the plasma. To do so, the controller is also allowed to set the total torque injected (TINJ) of the neutral beams (DIII-D has eight neutral beam injectors at different positions around the tokamak, so it is generally possible to control both total power and total torque independently). Controlling both of these quantities simultaneously is of interest since rotation shear often results in better confinement and less chance of instabilities in the plasma [Bondeson and Ward, 1994, Groebner et al., 1990]. In addition, we assume a multi-task setting where the requested targets for β_N and rotation can be set every trajectory. Specifically, the β_N target is drawn from $U(1.5\%, 2.5\%)$ and the rotation target is drawn from $U(25, 125)$ krad/s every trajectory. These targets are appended to the state space.

The learned, ground-truth dynamics model is also more sophisticated here. In addition to the inputs and outputs used by the β tracking environment model, the inputs for this model also include rotation and TINJ at times t, $t - \Delta t$, and $t + \Delta t$ for TINJ only. This model receives additional information about the plasma (e.g. the shape of the plasma); however, we have assumed these inputs are fixed to reasonable values in order to avoid partial observability problems. In total, the state space of this problem is 10D (targets plus current and past observations for β_N, rotation, PINJ, and TINJ) and the action space is 2D (next PINJ and TINJ settings).

A.7.2 Robotics Problems

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.

Environment	TIP	sTIP	DIP	sDIP	MPC	PETS	SAC	TD3	PPO	FEEF	HUCRL	TS	BPTT	BARL	EIG_T
Pendulum	**21**	36	36	46	46	5.6k	7k	26k	14k	800	>50k	>50k	>50k	**21**	56
Cartpole	131	141	161	141	201	1.63k	32k	18k	>1M	>2.5k	>6k	>6k	>6k	**111**	121
β Tracking	**46**	76	276	131	76	330	12k	17k	39k	300	480	420	450	186	>1k
β + Rotation	**201**	>500	>500	>500	>500	400	30k	>50k	>50k	>2k	>5k	>5k	>5k	>500	>1k
Reacher	**251**	>400	>1k	>1k	751	700	23k	13k	>100k	>5k	6.6k	4.5k	3.7k	**251**	>1.5k

Table A.7: **Sample Complexity Comparison of All Methods:** Median number of samples across 5 seeds required to reach 'solved' performance, averaged across 5 trials. We determine 'solved' performance by running an MPC policy (similar to the one used for evaluation) on the ground truth dynamics to predict actions. We record $> n$ when the median run is unable to solve the problem by the end of training after collecting n datapoints. The methods in the rightmost section operate in the TQRL setting and therefore have more flexible access to the MDP dynamics for data collection.

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 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 along with the 2-dimensional control space.

A.8 Additional Results

Due to space constraints in the main paper, we omitted results for the methods sDIP and BPTT. The are included alongside the rest in Table A.7. They are outperformed across the board by TIP.

A.9 Additional Related Work

A.9.1 Bayesian Exploration Techniques

Given unlimited computation and an accurate prior, solving the Bayes-adaptive MDP [Ross et al., 2007] gives an optimal tradeoff between exploration and exploitation by explicitly accounting for the updated beliefs that would result from future observations and planning to find actions that result in high rewards as quickly as can be managed given the current posterior. However, this is computationally expensive even in small finite MDPs and totally intractable in continuous settings. Kolter and Ng [2009] and Guez et al. [2012] show that even approximating these techniques can result in substantial theoretical reductions in sample complexity compared to frequentist PAC-MDP bounds as in Kakade [2003]. Another line of work [Dearden et al., 1998, 1999] uses the myopic value of perfect information as a heuristic for similar Bayesian exploration in the tabular MDP setting. Further techniques for exploration include knowledge gradient policies [Ryzhov et al., 2019, Ryzhov and Powell, 2011], which approximate the value function of the Bayes-adaptive MDP and information-directed sampling (IDS) [Russo and 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 using neural networks in Nikolov et al. [2019]. Another

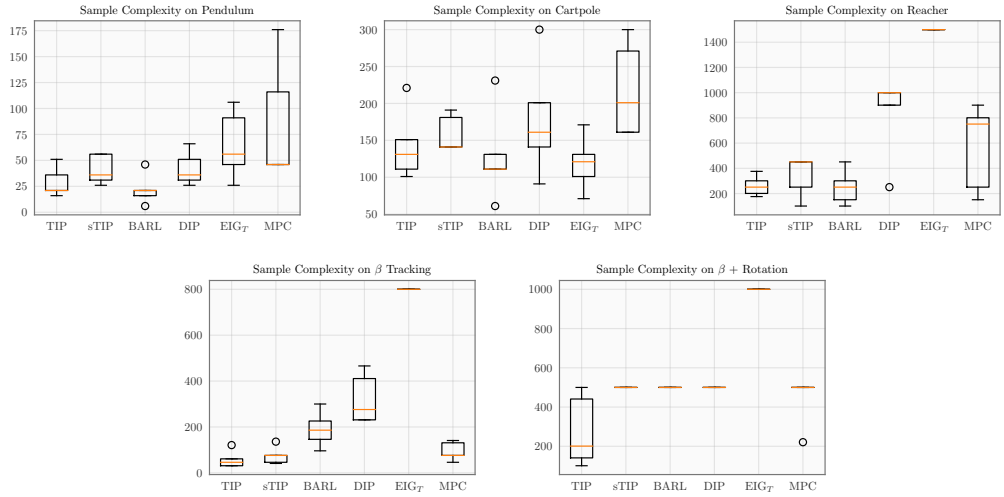

Figure A.2: Box plots showing sample complexity figures across the 5 random seeds run. Each of these show for a given training run how many samples were needed to achieve the performance of an MPC controller given ground truth dynamics averaged across test episodes. We imputed the maximum number of samples for agents that failed to ever solve the problem on a given run.

very relevant recent paper [Ball et al., 2020] gives an acquisition strategy in policy space that iteratively trains a data-collection policy in the model that trades off exploration against exploitation using methods from active learning. Achterhold and Stueckler [2021] use techniques from BOED to efficiently calibrate a Neural Process representation of a distribution of dynamics to a particular instance, but this calibration doesn't include information about the task. A tutorial on Bayesian RL methods can be found in Ghavamzadeh et al. [2016] for further reference.

A.9.2 Gaussian Processes (GPs) in Reinforcement Learning

There has been substantial prior work using GPs [Rasmussen and Williams, 2008] in reinforcement learning. Most well-known is PILCO [Deisenroth and Rasmussen, 2011], which computes approximate analytic gradients of policy parameters through the GP dynamics model while accounting for uncertainty. The original work is able to propagate the first 2 moments of the occupancy distribution through time using the GP dynamics and backpropagate gradients of the rewards to policy parameters. In Wilson et al. [2020], a method is developed for efficiently sampling functions from a GP posterior with high accuracy. One application show in their work is a method of using these samples to backpropagate gradients of rewards through time to policy paramters, which can be interpreted as a different sort of PILCO implementation. Most related to our eventual MPC-based method is [Kamthe and Deisenroth, 2018], which gives a principled probabilistic model-predictive control algorithm for GPs. We combine ideas from this paper, PETS [Chua et al., 2018], and the ability to sample posterior functions discussed above to give our eventual MPC component as discussed in Section 2.4.2.

B | Appendix for chapter 3

B.1 Proofs

B.1.1 Proof of Theorem 1

Proof. We will prove this theorem by construction. The intuition behind this construction is simple: We create 2 paths for the agent to take. The value maximizing path includes an unknown transition dynamic that can be learned upon a single traversal. However, because of the expected loss from that first traversal down this path, the Q-function favors an exploration policy that always takes the suboptimal path and the algorithm never learns the necessary variable to identify the optimal path during test time.

Fix some $H > 7$, $|\mathcal{S}| = N \geq 7$, $\epsilon > 0$. Consider a distribution over MDPs with states numbered 1-N and 2 actions as depicted in Figure B.1. The transition function is known for state 1, where action 1 transitions to state 2 and action 2 transitions to state 4. At state 2, either action transitions to state 3. At state 3 every action returns the agent to state 7. In state 4, there are two possible options, each with prior probability 0.5. Either action 1 will lead to state 5 and action 2 to state 6 or vice versa. At both states 5 and 6, any action will cause agent to visit state 7. States 7-(N-1) all transition from n to n+1 under any action and state N transitions to state 1 under any action. Given this design, we set the episode length to $h = |\mathcal{S}| - 2$.

The reward function of all MDPs is the same: 0 for all states besides 3 and 6, 1 for 6, and c (a parameter we'll discuss later) for 3.

At a high level, the design of the MDP is to make sure that π_x^b chooses action 1 at state 1 instead of exploring the risky option of action 2 that would lead to improved test-time performance. To ensure this, it is sufficient to show that $Q^{\pi_x^b}(1,1) > Q^{\pi_x^b}(1,2)$. We know that for a horizon H there will be $\lfloor \frac{H}{|\mathcal{S}|-2} \rfloor$ episodes completed (with the reward obtained at the last timestep). Once an action is attempted at state 4, the agent can infer based on the transition which of the 2 MDPs it is operating in and make the correct choice and attain return 1 in the episode. The first time an action is attempted in state 4, the expected value of the action is 0.5. If state 3 is visited by taking action 1 at state 1, the return of an episode will be c.

From these facts, we can see that $Q^{\pi_x^b}(1,1) = c\lfloor \frac{H}{|\mathcal{S}|-2} \rfloor$ and $Q^{\pi_x^b}(1,2) = \lfloor \frac{H}{|\mathcal{S}|-2} \rfloor - 0.5$. So, for $Q^{\pi_x^b}(1,1) > Q^{\pi_x^b}(1,2)$ to hold, c must be greater than $\frac{\lfloor \frac{H}{|\mathcal{S}|-2} \rfloor - 0.5}{\lfloor \frac{H}{|\mathcal{S}|-2} \rfloor}$.

An agent that behaves optimally in the MDP according to our objective will take action 2 at state 1 on the first try as it results in the maximum test time performance. The test-time policy of an agent that never takes action 2 at state 1 will take action 1 at state 1 as $c > 0.5$. So the simple regret of such an agent will be $1 - c$.

We choose a value $c = \frac{\lfloor \frac{H}{|\mathcal{S}|-2} \rfloor - 0.5}{\lfloor \frac{H}{|\mathcal{S}|-2} \rfloor} + \frac{\epsilon}{2}$. Then the regret will be

$$1 - c = \frac{1}{2\lfloor \frac{H}{|\mathcal{S}|-2} \rfloor} - \frac{\epsilon}{2}.$$

□

B.1.2 Proof of Theorem 2

This proof is fairly straightforward. We show by induction over the number of remaining timesteps that at any point π_h will choose the action that will on average lead to the best performance on the objective J. We can accomplish this via arguing that the Q^{π_h} function will be maximized by this action.

Proof. First consider the case where there is a single timestep remaining in exploration, the agent is at state s and maintains an updated and accurate belief b_τ. Here the state-action value function $Q_t^{\pi_h}$ (here, the subscript is for timesteps remaining in exploration) can be written as

$$Q_1^{\pi_h}(s, b_\tau, a) = \mathbb{E}_{T,R\sim b_\tau, s'\sim T(s,a), r=R(s,a,s')} \left[\mathrm{BR}(b_{[s_t,a_t,s'_t,r_t]||\tau}) \right].$$

π_h by definition will choose the action $\mathrm{argmax}_a Q_1^{\pi_h}(s, b_\tau, a)$ in this situation. Here, a is precisely the action that will lead on expectation to the best performance measured by J as it is the action such that on average over the belief it will lead to a transition that will allow the belief to be updated in the way that leads to the best Bayes return. Therefore π_h is optimal for single-step exploration under J.

Next assume for the sake of induction that for all $s \in \mathcal{S}, b \in \mathcal{B}, H = N - 1$, executing π_h with initial belief b starting from state s for H steps will lead to an optimal value of J and that π_h is executing starting from state s and accurate and updated belief b_τ with N steps of exploration remaining. In this situation,

$$Q_N^{\pi_h}(s, b_\tau, a) = \mathbb{E}_{T,R\sim b_\tau, s'\sim T(s,a), r=R(s,a,s')} \left[Q_{N-1}^{\pi_h}(s', b_{\{(s_t,a_t,s'_t,r_t)\}\cup\tau}, \pi_h(s', b_{\{(s_t,a_t,s'_t,r_t)\}\cup\tau})) \right].$$

Since by the inductive hypothesis π_h is optimal for $H = N - 1$, we know that $Q_{N-1}^{\pi_h}$ will be the maximal attainable value for any particular state-belief-action triple with $N - 1$ steps of exploration remaining. As $Q_N^{\pi_h}$ gives the expected observed transition given the belief for a particular action, we would expect that on average the action that maximizes $Q_N^{\pi_h}$ to lead to the largest eventual Bayes Return when π_h is executed afterwards. Since π_h with N steps remaining will choose the action given s, b_τ that maximizes $Q_N^{\pi_h}$, it must be optimal when acting with N steps remaining.

Therefore, by induction, this must hold for all N and we can conclude that π_h is optimal or in other words that $J(\pi_h, \hat{\pi}_N) = \max_{\pi_x} J(\pi_x, \hat{\pi}_N)$. □

B.2 Exact Experiments

Implementation Details We implemented the exact EHIG-MDP and BAMDP methods in Julia and executed them for up to 5 timesteps of exploration. We executed all experiments using

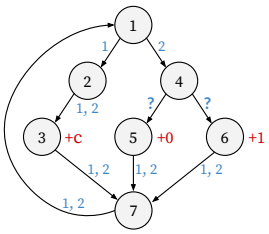

Figure B.1: A depiction of the MDP distribution used in the construction for the proof of Theorem 1. Edges with both numbers on them mean both actions lead along that edge and the edges with question marks after state 4 mean that it's not clear which action (out of 1 or 2) leads to which state.

multithreading on 12 CPUs at a time. Each environment was represented by a generic Dirichlet-Multinomial prior through which we were able to hardcode representations of the prior knowledge about both LavaRun and SkateTrick. We gave the agents perfect knowledge of the reward functions but not the dynamics, as described in Section 3.4.3. To evaluate the performance, we simply computed the true value function for the optimal policy according to the BAMDP and EHIG-MDP agents under the real dynamics after exploration had concluded.

Indexing Scheme for Belief MDP In order to solve the EHIG-MDP and BAMDP we needed to construct a table that contained the entire $\mathcal{S} \times \mathcal{B}$ domain of the Bayesian agents. As the support of a multinomial distribution over $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$ is the set of all whole number-valued vectors of length $k = |\mathcal{S} \times \mathcal{A} \times \mathcal{S}|$ that sum to the number of observations, we needed to construct quite a large table. In particular, if n observations have been made, there are $\binom{n+k-1}{k-1}$ possible distinct observation vectors. We chose an indexing scheme where we first incremented $n = 0, 1, \ldots, H$, and then for each setting of n had a segment of the table corresponding to beliefs with a total number of obeservations n. Within each segment, we indexed the beliefs of size n in lexical order and then finally by the factor of $|\mathcal{S}|$ corresponding to the actual MDP state of the agent. This setup allowed for a maximally memory-efficient storage arrangement for an admittedly computationally inefficient method of solving these MDPs.

B.3 Approximate Experiments

Implementation Details We ran all experiments three-at-a-time on A60 GPUs rented from a cloud computing service. Besides RL^2, each experiment lasted around 15 hours with a few hours' variation.

We provide the state and action dimensions for each environment in Table B.1. However, note that the dimensionalities listed do not include the additional time dimension added by the algorithm.

We did not spend a large amount of effort on hyperparameter tuning on this project; in fact, we used the default hyperparameters from the implementation of Varibad included with Zintgraf et al. [2020].

Environment	State Dim	Action Dim
HalfCheetahVel	18	6
SparsePoint	2	2
LavaPoint	2	2
BetaLimit	4	1

Table B.1: State and action dimensions of environments used for continuous experiments.

C | Appendix for chapter 4

C.1 Appendix

C.1.1 Auxiliary Results

Lemma 2. *Let $t \in [N]$. Then, for every $(s,a) \in \mathcal{S} \times \mathcal{A}$,*

1. *If $\overline{Q}_h^t(s,a) \geq \mathcal{T}_h^* \overline{Q}_{h+1}^t(s,a)$ holds for all $h \in [H]$, then $\overline{Q}_h^t(s,a) \geq Q_h^*(s,a)$ is true for all $h \in [H]$.*
2. *If $\underline{Q}_h^t(s,a) \leq \mathcal{T}_h^* \underline{Q}_{h+1}^t(s,a)$ holds for all $h \in [H]$, then $Q_h^*(s,a) \geq \underline{Q}_h^t(s,a)$ is true for all $h \in [H]$.*

Proof. In order to prove part 1., let $s \in \mathcal{S}$ and $a \in \mathcal{A}$ and assume $\overline{Q}_h^t(s,a) \geq \mathcal{T}_h^* \overline{Q}_{h+1}^t(s,a)$ for all $h \in [H]$ and $t \in [T]$. We prove $\forall h \in [H]$, $\overline{Q}_h^t(s,a) \geq Q_h^*(s,a)$ by induction on $h = H, H-1, \ldots, 1$. For the initial case $h = H$, we have

$$\overline{Q}_H^t(s,a) \overset{\text{assumption}}{\geq} \mathcal{T}_h^* \overline{Q}_{H+1}^t(s,a) \tag{C.1}$$

$$\overset{\text{Def. of } \mathcal{T}_h^*}{=} r_H(s,a) + \mathbb{E}_{s' \sim \mathbb{P}_H(\cdot|s,a)} \Big[\max_{a' \in \mathcal{A}} \overline{Q}_{H+1}^t(s',a') \Big] \tag{C.2}$$

$$= r_H(s,a) \tag{C.3}$$

$$= Q_H^*(s,a). \tag{C.4}$$

For the inductive step, we assume that $Q_{h+1}^*(s,a) \leq \overline{Q}_{h+1}^t(s,a)$. Then,

$$Q_h^*(s,a) = \mathcal{T}_h^* Q_{h+1}^*(s,a) \tag{C.5}$$

$$\overset{\text{Def. of } \mathcal{T}_h^*}{=} r_h(s,a) + \mathbb{E}_{s' \sim \mathbb{P}_h(\cdot|s,a)} Big[\max_{a' \in \mathcal{A}} Q_{h+1}^*(s',a') \Big] \tag{C.6}$$

$$\overset{\text{inductive hypothesis}}{\leq} r_h(s,a) + \mathbb{E}_{s' \sim \mathbb{P}_h(\cdot|s,a)} \Big[\max_{a' \in \mathcal{A}} \overline{Q}_{h+1}^t(s',a') \Big] \tag{C.7}$$

$$\overset{\text{Def. of } \mathcal{T}_h^*}{=} \mathcal{T}_h^* \overline{Q}_{h+1}^t(s,a) \tag{C.8}$$

$$\overset{\text{assumption}}{\leq} \overline{Q}_h^t(s,a). \tag{C.9}$$

This shows $\overline{Q}_h^t(s,a) \geq Q_h^*(s,a)$ for all $h \in [H]$ and thus concludes the proof of the first claim. The second part can be shown analogously.

□

The following is a standard result that can be found in multiple works.

Lemma 3. *Consider a kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that $k(x, x) \leq 1$ for every $x \in \mathcal{X}$. Then for all $h \in [H]$ and $\lambda \geq 1$ we have*

$$\sum_{t=1}^{N} \sigma_h^t(s_h^t, a_h^t) \leq \sqrt{3\Gamma_k(N, \lambda)N}. \tag{C.10}$$

Proof. We can for example invoke the result of Lemma 3 in Bogunovic and Krause [2021] that in our notation reads as:

$$\sum_{t=1}^{N} \sigma_h^t(s_h^t, a_h^t) \leq \sqrt{\lambda^{-1}(2\lambda + 1)\Gamma_k(N, \lambda)N}, \tag{C.11}$$

for $\lambda > 0$. Setting $\lambda \geq 1$, we obtain

$$\sum_{t=1}^{N} \sigma_h^t(s_h^t, a_h^t) \leq \sqrt{3\Gamma_k(N, \lambda)N}. \tag{C.12}$$

□

C.1.2 Proof of theorem 3

Let $\hat{\pi}_N$ be the best-policy estimate returned by the algorithm. Recall the definition of

$$\pi_T^{*\geq h} := \left(\pi_{T,h'}^{*\geq h}\right)_{h'=1}^{H} := \begin{cases} \hat{\pi}_{N,h'} & \text{for } h' = 1, \ldots, h-1 \\ \pi_{h'}^* & \text{for } h' = h, \ldots, H \end{cases} \tag{C.13}$$

as the policy that equals our best-policy estimate $\hat{\pi}_N$ until step $h - 1$ and then equals the optimal policy π^*.

We start the proof with the following useful lemma.

Lemma 4. *Let $\hat{\pi}_N$ be a best-policy estimate, let $s \in \mathcal{S}$ be an initial state, and let $h \in [H]$. Using the notation from eq. (C.13), we obtain*

$$V_1^{\pi_T^{*\geq h}}(s) - V_1^{\pi_T^{*\geq h+1}}(s) = \mathbb{E}_{a_1,\ldots,s_h \text{ following } \hat{\pi}_N}\left[Q_h^*\big(s_h, \pi_h^*(s_h)\big) - Q_h^*\big(s_h, \hat{\pi}_{N,h}(s_h)\big) \big| s_1 = s\right].$$

Proof. To formally prove the lemma, we first explicitly express $V_1^{\pi_T^{*\geq h}}(s)$ and $V_1^{\pi_T^{*\geq h+1}}(s)$ for an arbitrary initial state $s \in \mathcal{S}$ as

$$V_1^{\pi_T^{*\geq h}}(s) \tag{C.14}$$

$$= \mathbb{E}_{a_1,\ldots,s_H \text{ following } \pi_T^{*\geq h} | s_1 = s}\left[\sum_{h'=1}^{H} r_{h'}(s_{h'}, a_{h'})\right] \tag{C.15}$$

$$= \mathbb{E}_{a_1,\ldots,s_h \text{ following } \hat{\pi}_N | s_1 = s}\left[\mathbb{E}_{a_h,\ldots,s_H \text{ following } \pi^* | s_h}\left[\sum_{h'=1}^{H} r_{h'}(s_{h'}, a_{h'})\right]\right] \tag{C.16}$$

$$= \mathbb{E}_{a_1,\ldots,s_h \text{ following } \hat{\pi}_N | s_1 = s}\left[\sum_{h'=1}^{h} r_{h'}(s_{h'}, a_{h'}) + \mathbb{E}_{a_h,\ldots,s_H \text{ following } \pi^* | s_h}\left[\sum_{h'=h+1}^{H} r_{h'}(s_{h'}, a_{h'})\right]\right], \tag{C.17}$$

and

$$V_1^{\pi_T^{*\geq h+1}}(s_1) \tag{C.18}$$

$$= \mathbb{E}_{a_1,\ldots,s_H \text{ following } \pi_T^{*\geq h+1}|s_1=s}\Big[\sum_{h'=1}^{H} r_{h'}(s_{h'},a_{h'})\Big] \tag{C.19}$$

$$= \mathbb{E}_{a_1,\ldots,s_h \text{ following } \hat{\pi}_N|s_1=s}\Bigg[\mathbb{E}_{a_h,s_{h+1} \text{ following } \hat{\pi}_N|s_h}\Bigg[\mathbb{E}_{a_{h+1},\ldots,s_H \text{ following } \pi^*|s_{h+1}}$$

$$\Big[\sum_{h'=1}^{h} r_{h'}(s_{h'},a_{h'}) + \sum_{h'=h+1}^{H} r_{h'}(s_{h'},a_{h'})\Big]\Bigg]\Bigg] \tag{C.20}$$

$$= \mathbb{E}_{a_1,\ldots,s_h \text{ following } \hat{\pi}_N|s_1=s}\Bigg[\sum_{h'=1}^{h} r_{h'}(s_{h'},a_{h'})$$

$$+ \mathbb{E}_{a_h,s_{h+1} \text{ following } \hat{\pi}_N|s_h}\Bigg[\mathbb{E}_{a_{h+1},\ldots,s_H \text{ following } \pi^*|s_{h+1}}\Big[\sum_{h'=h+1}^{H} r_{h'}(s_{h'},a_{h'})\Big]\Bigg]\Bigg]. \tag{C.21}$$

eqs. (C.15) and (C.19) use the definition of V_1^{π}, eqs. (C.16) and (C.20) use the definition of $\pi_T^{*\geq h}$ and $\pi_T^{*\geq h+1}$ from eq. (C.13), and eqs. (C.17) and (C.21) use the property that integration is a linear operator.

Lemma 4 then follows from eqs. (C.17) and (C.21) as well as the definition of Q_h^*:

$$V_1^{\pi_T^{*\geq h}}(s_1) - V_1^{\pi_T^{*\geq h+1}}(s_1) = \mathbb{E}_{a_1,\ldots,s_h \text{ following } \pi^N|s_1=s}\Bigg[\sum_{h'=1}^{h} r_{h'}(s_{h'},a_{h'}) - \sum_{h'=1}^{h} r_{h'}(s_{h'},a_{h'})$$

$$+ \mathbb{E}_{a_h,\ldots,s_H \text{ following } \pi^*|s_h}\Big[\sum_{h'=h+1}^{H} r_{h'}(s_{h'},a_{h'})\Big]$$

$$- \mathbb{E}_{a_h,s_{h+1} \text{ following } \hat{\pi}_N|s_h}\Bigg[\mathbb{E}_{a_{h+1},\ldots,s_H \text{ following } \pi^*|s_{h+1}}\Big[\sum_{h'=h+1}^{H} r_{h'}(s_{h'},a_{h'})\Big]\Bigg]\Bigg] \tag{C.22}$$

$$= \mathbb{E}_{a_1,\ldots,s_h \text{ following } \hat{\pi}_N|s_1=s}\Big[Q_h^*\big(s_h,\pi_h^*(s_h)\big) - Q_h^*\big(s_h,\hat{\pi}_{N,h}(s_h)\big)\Big]. \tag{C.23}$$

□

We proceed with the proof by using the notation from eq. (C.13). We can decompose the instantaneous regret for an arbitrary initial state $s \in \mathcal{S}$ as follows:

$$V_1^*(s) - V_1^{\hat{\pi}_N}(s) = V_1^{\pi_T^{*\geq 1}}(s) - V_1^{\pi_T^{*\geq H+1}}(s) \tag{C.24}$$

$$= \sum_{h=1}^{H}\Big(V_1^{\pi_T^{*\geq h}}(s) - V_1^{\pi_T^{*\geq h+1}}(s)\Big) \tag{C.25}$$

$$\overset{\text{Lemma 4}}{=} \sum_{h=1}^{H} \mathbb{E}_{s_1,a_1,\ldots,s_h \text{following } \hat{\pi}_N}\Big[Q_h^*\big(s_h,\pi_h^*(s_h)\big) - Q_h^*\big(s_h,\hat{\pi}_{N,h}(s_h)\big)\Big|s_1=s\Big]. \tag{C.26}$$

The intuition behind lemma 4 used in eq. (C.26) is as follows. Both $V_1^{\pi_T^{*\geq h}}(s)$ and $V_1^{\pi_T^{*\geq h+1}}(s)$ refer to the same random trajectory segment $(s_1, a_1, \ldots, s_h)$ until step h (i.e., the same initial state and policy are used), which is captured as $\mathbb{E}_{s_1,a_1,\ldots,s_h \text{ following } \hat{\pi}_N}[\cdot]$. For the remaining steps $h, \ldots, H$, the policies only differ at step h, a property which is captured in the difference $Q_h^*\big(s_h, \pi_h^*(s_h)\big) - Q_h^*\big(s_h, \hat{\pi}_{N,h}(s_h)\big)$.

Conditioning on the event in Lemma 1 holding true and by invoking lemma 2, we have that:

$$\underline{Q}_h^t(s,a) \leq Q_h^*(s,a) \leq \overline{Q}_h^t(s,a), \tag{C.27}$$

holds for every $h \in [H]$, $t \in [T]$, and $(s,a) \in \mathcal{S} \times \mathcal{A}$. Next, we proceed to bound $Q_h^*\big(\cdot, \pi_h^*(\cdot)\big) - Q_h^*\big(\cdot, \hat{\pi}_{N,h}(\cdot)\big)$ from eq. (C.26) uniformly on $\mathcal{S}$. We have:

$$\begin{aligned}
Q_h^*(s,\pi_h^*(s)) - Q_h^*(s,\hat{\pi}_{N,h}(s)) &\overset{\text{eq. (C.27)}}{\leq} Q_h^*(s,\pi_h^*(s)) - \max_{t\in[N]} \underline{Q}_h^t(s,\hat{\pi}_{N,h}(s)) && \text{(C.28)}\\
&\overset{\text{Def. of } \hat{\pi}_{N,h}}{=} Q_h^*(s,\pi_h^*(s)) - \max_{a\in\mathcal{A}} \max_{t\in[N]} \underline{Q}_h^t(s,a) && \text{(C.29)}\\
&= \min_{t\in[T]} \Big(Q_h^*(s,\pi_h^*(s)) - \max_{a\in\mathcal{A}} \underline{Q}_h^t(s,a) \Big) && \text{(C.30)}\\
&\overset{\text{Def. of } \pi_h^*}{=} \min_{t\in[T]} \Big(\max_{a\in\mathcal{A}} Q_h^*(s,a) - \max_{a\in\mathcal{A}} \underline{Q}_h^t(s,a) \Big) && \text{(C.31)}\\
&\overset{\text{eq. (C.27)}}{\leq} \min_{t\in[T]} \Big(\max_{a\in\mathcal{A}} \overline{Q}_h^t(s,a) - \max_{a\in\mathcal{A}} \underline{Q}_h^t(s,a) \Big) && \text{(C.32)}\\
&\overset{\text{eq. (4.10)}}{\leq} \min_{t\in[T]} \Big(\max_{a\in\mathcal{A}} \overline{Q}_h^t(s_h^t,a) - \max_{a\in\mathcal{A}} \underline{Q}_h^t(s_h^t,a) \Big) && \text{(C.33)}\\
&\overset{\text{eq. (4.11)}}{\leq} \min_{t\in[T]} \Big(\overline{Q}_h^t(s_h^t,a_h^t) - \underline{Q}_h^t(s_h^t,a_h^t) \Big) && \text{(C.34)}\\
&\leq \frac{1}{N} \sum_{t=1}^{N} \Big(\overline{Q}_h^t(s_h^t,a_h^t) - \underline{Q}_h^t(s_h^t,a_h^t) \Big). && \text{(C.35)}
\end{aligned}$$

Next, for convenience we introduce the notation

$$d_h^t := \Big(\overline{Q}_h^t(s_h^t,a_h^t) - \mathcal{T}_h^* \overline{Q}_{h+1}^t(s_h^t,a_h^t) \Big) + \Big(\mathcal{T}_h^* \underline{Q}_{h+1}^t(s_h^t,a_h^t) - \underline{Q}_h^t(s_h^t,a_h^t) \Big), \tag{C.36}$$

and obtain the following upper bound on $\overline{Q}_h^t(s_h^t,a_h^t) - \underline{Q}_h^t(s_h^t,a_h^t)$ (from eq. (C.35)) for every $h \in [H]$, $t \in [T]$:

$$\begin{aligned}
\overline{Q}_h^t(s_h^t,a_h^t) - \underline{Q}_h^t(s_h^t,a_h^t) &\overset{\text{eq. (C.36)}}{=} d_h^t + \mathcal{T}_h^* \overline{Q}_{h+1}^t(s_h^t,a_h^t) - \mathcal{T}_h^* \underline{Q}_{h+1}^t(s_h^t,a_h^t) && \text{(C.37)}\\
&\overset{\text{Def. of } \mathcal{T}_h^*}{=} d_h^t + \mathbb{E}_{s'\sim\mathbb{P}_h(\cdot|s_h^t,a_h^t)} \Big(\max_{\overline{a}\in\mathcal{A}} \overline{Q}_{h+1}^t(s',\overline{a}) - \max_{\underline{a}\in\mathcal{A}} \underline{Q}_{h+1}^t(s',\underline{a}) \Big) && \text{(C.38)}\\
&\leq d_h^t + \max_{s'\in\mathcal{S}} \Big(\max_{\overline{a}\in\mathcal{A}} \overline{Q}_{h+1}^t(s',\overline{a}) - \max_{\underline{a}\in\mathcal{A}} \underline{Q}_{h+1}^t(s',\underline{a}) \Big) && \text{(C.39)}\\
&\overset{\text{eq. (4.10)}}{=} d_h^t + \Big(\max_{\overline{a}\in\mathcal{A}} \overline{Q}_{h+1}^t(s_{h+1}^t,\overline{a}) - \max_{\underline{a}\in\mathcal{A}} \underline{Q}_{h+1}^t(s_{h+1}^t,\underline{a}) \Big) && \text{(C.40)}\\
&\overset{\text{eq. (4.11)}}{\leq} d_h^t + \Big(\overline{Q}_{h+1}^t(s_{h+1}^t,a_{h+1}^t) - \underline{Q}_{h+1}^t(s_{h+1}^t,a_{h+1}^t) \Big). && \text{(C.41)}
\end{aligned}$$

Using the definition of $\underline{Q}^t_{H+1}$ and $\overline{Q}^t_{H+1}$ as the zero functions, we can unroll the recursive inequality from eq. (C.41) and upper bound $\overline{Q}^t_h(s^t_h, a^t_h) - \underline{Q}^t_h(s^t_h, a^t_h)$ for every $h \in [H]$, $t \in [T]$ as follows:

$$\overline{Q}^t_h(s^t_h, a^t_h) - \underline{Q}^t_h(s^t_h, a^t_h) \leq \sum_{h'=h}^{H} d^t_{h'}. \tag{C.42}$$

$$\overline{Q}^t_h(s^t_h, a^t_h) - \underline{Q}^t_h(s^t_h, a^t_h) \tag{C.43}$$

$$\leq \sum_{h'=h}^{H} \left[\left(\overline{Q}^t_{h'}(s^t_{h'}, a^t_{h'}) - \mathcal{T}^*_{h'} \overline{Q}^t_{h'+1}(s^t_{h'}, a^t_{h'}) \right) + \left(\mathcal{T}^*_{h'} \underline{Q}^t_{h'+1}(s^t_{h'}, a^t_{h'}) - \underline{Q}^t_{h'}(s^t_{h'}, a^t_{h'}) \right) \right] \tag{C.44}$$

$$\overset{\text{Lemma 1}}{\leq} \sum_{h'=h}^{H} 4\beta \sigma^t_{h'}(s^t_{h'}, a^t_{h'}). \tag{C.45}$$

By substituting the bound from eq. (C.45) in eq. (C.35), and then in eq. (C.26), we arrive at:

$$V^*_1(s) - V^{\hat{\pi}_N}_1(s) \leq 4\beta \sum_{h=1}^{H} \sum_{h'=h}^{H} \frac{1}{N} \sum_{t=1}^{N} \sigma^t_{h'}(s^t_{h'}, a^t_{h'}) \leq 2\sqrt{3}\beta H(H+1) \sqrt{\frac{\Gamma_k(N,\lambda)}{N}}, \tag{C.46}$$

where the last inequality follows from lemma 3. Since eq. (C.46) holds for any $s \in S$, we arrive at our main result:

$$\|V^*_1 - V^{\hat{\pi}_N}_1\|_{\ell^\infty(\mathcal{S})} \leq 2\sqrt{3}\beta H(H+1) \sqrt{\frac{\Gamma_k(N,\lambda)}{N}}. \tag{C.47}$$

C.1.3 Offline contextual Bayesian optimization

Algorithm 8 AE-LSVI for offline contextual Bayesian optimization

Require: kernel function $k(\cdot,\cdot)$, exploration parameter $\beta > 0$, regularization parameter λ,

for $t = 1, \ldots, N$ **do**
  Obtain $\overline{Q}^t$ and $\underline{Q}^t$ from Equation (4.19)
  Choose $s^t \in \operatorname{argmax}_{s \in S} \left[\max_{a \in A} \overline{Q}^t(s,a) - \max_{a \in A} \underline{Q}^t(s,a) \right]$
  Choose $a^t \in \operatorname{argmax}_{a \in \mathcal{A}} \overline{Q}^t(s^t, a)$
  Observe the reward $y_t = Q^*(s^t, a^t) + \eta_t$
end for
Output the policy estimate $\hat{\pi}_N$ such that $\hat{\pi}_N(\cdot) = \operatorname{argmax}_{a \in \mathcal{A}} \max_{t \in [N]} \underline{Q}^t(s,a)$

Our algorithm for the offline contextual Bayesian optimization is presented in Algorithm 8. As a side observation, we note that similarly to Char et al. [2019], we can also simply incorporate context weights (i.e., given $\omega(s)$ that represents some weighting of context s that may depend on the probability of seeing s at evaluation time or the importance of s), in case they are available, into the proposed acquisition function, i.e.,

$$s^t \in \operatorname{argmax}_{s \in S} \left[\left(\max_{a \in A} \overline{Q}^t(s,a) - \max_{a \in A} \underline{Q}^t(s,a) \right) w(s) \right]. \tag{C.48}$$

Proof of Corollary 1

Proof. In this proof, we condition on the event in eq. (4.20) holding true. Similar arguments to the ones in eq. (C.37) – eq. (C.41) lead to the following for every $s \in \mathcal{S}$:

$$\max_{a\in\mathcal{A}} Q^*(s,a) - Q^*(s,\hat{\pi}_N(s)) \overset{\text{eq. (4.20)}}{\leq} \max_{a\in\mathcal{A}} Q^*(s,a) - \max_{t\in[N]} \underline{Q}^t(s,\hat{\pi}_N(s)) \tag{C.49}$$

$$\overset{\text{Def. of } \hat{\pi}_N}{=} \max_{a\in\mathcal{A}} Q^*(s,a) - \max_{a\in\mathcal{A}} \max_{t\in[N]} \underline{Q}^t(s,a) \tag{C.50}$$

$$= \min_{t\in[T]} \left(\max_{a\in\mathcal{A}} Q^*(s,a) - \max_{a\in\mathcal{A}} \underline{Q}^t(s,a) \right) \tag{C.51}$$

$$\overset{\text{eq. (4.20)}}{\leq} \min_{t\in[T]} \left(\max_{a\in\mathcal{A}} \overline{Q}^t(s,a) - \max_{a\in\mathcal{A}} \underline{Q}^t(s,a) \right) \tag{C.52}$$

$$\overset{\text{Def. of } s^t}{\leq} \min_{t\in[T]} \left(\max_{a\in\mathcal{A}} \overline{Q}^t(s^t,a) - \max_{a\in\mathcal{A}} \underline{Q}^t(s^t,a) \right) \tag{C.53}$$

$$\overset{\text{Def. of } a^t}{\leq} \min_{t\in[T]} \left(\overline{Q}^t(s^t,a^t) - \underline{Q}^t(s^t,a^t) \right) \tag{C.54}$$

$$\leq \frac{1}{N} \sum_{t=1}^{N} \left(\overline{Q}^t(s^t,a^t) - \underline{Q}^t(s^t,a^t) \right) \tag{C.55}$$

$$\overset{\text{eq. (4.19)}}{=} \frac{1}{N} \sum_{t=1}^{N} \left(2\beta_t \sigma^t(s^t,a^t) \right) \tag{C.56}$$

$$\leq \frac{2\beta_N}{N} \sum_{t=1}^{N} \sigma^t(s^t,a^t) \tag{C.57}$$

$$\overset{\text{Lemma 3}}{\leq} \frac{2\beta_N \sqrt{3\Gamma_k(N,\lambda)}}{\sqrt{N}}. \tag{C.58}$$

Finally, by setting $\epsilon \geq \frac{2\beta_N \sqrt{3\Gamma_k(N,\lambda)}}{\sqrt{N}}$ and expressing it in terms of N, we arrive at the main result. □

C.2 Additional Experimental Details

C.2.1 Implementation

We use an exact Gaussian Process with a squared exponential kernel with learned scale parameters in each dimension for the value function regression in eq. (4.3). We fit the kernel hyperparameters at each iteration using 1000 iterations of Adam [Kingma and Ba, 2014], maximizing the marginal log likelihood of the training data. We used the TinyGP package [Foreman-Mackey, 2021] built on top of JAX [Bradbury et al., 2018] in order to take advantage of JIT compilation. All experiments are conducted with a fixed bonus $\beta = 0.5$. We have empirically evaluated various settings of β in Appendix C.2.3. We uniformly sample 1,000 points from the state space and evaluate them to find an approximate maximizer to the objective in eq. (4.10).

DDQN and BDQN. For both of these methods we use networks with two hidden layers, each with 256 units. For the bootstrapped DQN, we use a network with 10 different heads, each representing a different Q function. For each step collected during exploration, a corresponding mask is generated and added to the replay buffer that signifies which heads will train on this sample. Each Q function has a probability of 0.5 of being trained on each transition.

C.2.2 Environments

Each environment is defined with a native reward function taken from the literature. We established upper and lower bounds on the reward function value and used them to scale the reward function values to $[0, 1]$ so that our environments would match the theoretical results in this paper.

Cartpole We use a modified version of the cartpole environment from Mehta et al. [2022a] that has dense rewards as implemented in Wang et al. [2019]. The state space is $4D$ and consists of the horizontal position and velocity of the cart as well as the angular position and velocity of the pole. p_0 in this environment is a normal distribution centered with the cart below the goal horizontally with the pole hanging down with very small variance. p_0' is the same distribution displaced 5 meters to the right.

Navigation This is a $2D$ navigation problem with dynamics of the form $s_{t+1} = s_t + B(s_t)a_t$, where $B(t) = \begin{bmatrix} \sin(x_2/10) + 4 & 0 \\ 0 & 1.5\cos(x_1/10) - 2 \end{bmatrix}$. The goal is fixed at $\begin{bmatrix} 6 \\ 9 \end{bmatrix}$. We define p_0 to be the uniform distribution over the axis-aligned rectangle given by corners $\begin{bmatrix} -8 \\ -9 \end{bmatrix}$ and $\begin{bmatrix} -6 \\ -6 \end{bmatrix}$. We define p_0' to be the uniform distribution over the axis-aligned rectangle given by corners $\begin{bmatrix} 1 \\ 4 \end{bmatrix}$ and $\begin{bmatrix} 3 \\ 7 \end{bmatrix}$. The reward function at every timestep is simply the negative ℓ_1-distance between the agent and the goal.

β Tracking and β + Rotation Our two simulated plasma control problems are taken from Mehta et al. [2022a], which gives a thorough description of their relevance to the problem of nuclear fusion. At a high level, β_N is a normalized plasma pressure ratio that is correlated with the economic output of a fusion reactor. Our β **Tracking** environment aims to adjust the injected power in the reactions in order to achieve a target value of $\beta_N = 2\%$. The initial state distribution p_0 is taken from a set of real datapoints from shots on the DIII-D tokamak in San Diego. Our alternate initial state distribution p_0' consists of simply adding 0.4 to each component of a vector sampled from p_0. The reward function is the negative ℓ_1-distance between the β_N value and 2%. The dynamics are given by a learned model of the plasma state as introduced in Char et al. [2023].

The β **+ Rotation** environment is a more complex plasma control problem, introducing an additional actuator (injected torque) and an additional control objective (controlling plasma rotation). Control of plasma rotation is key to plasma stability and this is a reduced version of the realistic problem. This environment also uses a model from Char et al. [2023] for the dynamics, real plasma states for the initial state distribution p_0, and a fixed translation for the alternate initial state distribution p_0'. We also include a randomly drawn target for β_N and rotation in the state space for every episode.

Environment	$\beta = 0.25$	$\beta = 0.5$	$\beta = 1$	$\beta = 2$
Cartpole	$\mathbf{17.2 \pm 0.3}$	16.8 ± 0.4	16.3 ± 0.5	15.1 ± 0.4
Navigation	$\mathbf{22.3 \pm 0.5}$	$\mathbf{22.3 \pm 0.4}$	$\mathbf{22.2 \pm 1.4}$	20.6 ± 1.3
β Tracking	$\mathbf{13.9 \pm 0.3}$	$\mathbf{14.0 \pm 0.4}$	13.2 ± 1.3	13.4 ± 0.7
β + Rotation	$\mathbf{14.8 \pm 0.3}$	14.3 ± 0.2	14.1 ± 0.9	13.3 ± 0.9

Table C.1: Average Return $\pm$ standard error of executing the identified best policy on the MDP starting from p_0' over 5 seeds after collecting 1000 timesteps of data using the method with varying values of the exploration parameter β.

C.2.3 Exploring β values

In the main paper, we report experiments with the exploration parameter $\beta = 0.5$ for all t. In this work, we do not explore principled methods of choosing β and welcome future work in the area. In lieu of this, we provide an empirical analysis of the sensitivity of AE-LSVI to varying settings of β. We ran the AE-LSVI method on our evaluation environments as in the experiments in Table 4.2, where we allowed each method to collect 1,000 timesteps of data and evaluated the identified policies on the environments starting from a evaluation initial distribution p_0' distinct from the initial distribution p_0. In Table C.1 we observe that lower values of β perform better because the confidence bounds seem too wide at higher settings, where the performance becomes similar to that of uncertainty sampling. Therefore, we recommend initially trying β-values around $0.2 - 0.5$ when applying .

C.3 RKHS Regression

At step t, we have data $\{(x_1, a_1, a'_1, w_1), \ldots, (x_t, a_t, a'_t, w_t)\}$. The kernel ridge regression estimate is defined by,

$$\mu_t = \text{argmin}_{f \in \mathcal{H}} \sum_{i=1}^{t} (f(x_i, a_i) - w_i)^2 + \lambda \|f\|^2_{\mathcal{H}} \,. \tag{C.59}$$

Denote by $\boldsymbol{w}_t = [w_1, \ldots, w_t]^T$ the vector of observations, $(K_t)_{i,j=1,\ldots,t} = k(x_i, a_i, x_j, a_j)$ the data kernel matrix, and $k_t(x, a) = [k(x, a, x_1, a_1), \ldots, k(x, a, x_t, a_t)]^T$ the data kernel features. We then have

$$\mu_t(x, a) = k_t(x, a)^T (K_t + \lambda \mathbf{1}_t)^{-1} \boldsymbol{w}_t \,. \tag{C.60}$$

We further have the posterior variance $\sigma_t(x, a)^2$ that determines the width of the confidence intervals,

$$\sigma_t(x, a)^2 = k(x, a, x, a) - k_t(x, a)^T (K_t + \lambda \mathbf{1}_t)^{-1} k_t(x, a) \,. \tag{C.61}$$

C.4 Proof of Theorem 4

In this section we will prove our main Theorem, 4. The overall strategy of the proof is to use our Lipschitz assumption on the link function (more precisely, the relative Lipschitzness of the reward r and the Borda function f_r) in order to go to the Borda function, which we can directly model from data. Then, we use our selection criteria as well as confidence bounds taken from Chowdhury and Gopalan [2017] and convergence rates taken from Kandasamy et al. [2019] in order to complete the argument. We give these cited results as lemmas in what follows.

In order to attain a particular policy performance with probability $1 - \delta$, we must bound the error of the estimates given by our KRR process for a particular confidence level. In order to do so, we adapt the result from Chowdhury and Gopalan [2017], Theorem 2.

Lemma 5. *Let* $\beta_t^{(r)} = 2||f_r||_\kappa + \sqrt{2(\Phi_{t-1}(\mathcal{X} \times \mathcal{A}) + 1 + \log(2/\delta))}$. *Then with probability* $1 - \delta$ *we have for all time* t *and any point* $(x, a) \in \mathcal{X} \times \mathcal{A}$,

$$|\mu_{t-1}(x, a) - f_r(x, a)| \leq \beta_t^{(r)} \sigma_{t-1}(x, a).$$

Proof. To prove this result, we will verify that all the conditions from Theorem 2 of Chowdhury and Gopalan [2017] hold. Recall Assumption 2 which states that $\|f_r\|_\kappa \leq B$. Next, we observe that since $a'_t \sim U(\mathcal{A})$ (independent of everything else), we have that $\mathbb{E}[w_t \mid \mathcal{F}_{t-1}] = f_r(x_t, a_t)$, where $\mathcal{F}_t = \rho\left(\{(x_s, a_s, a'_s, w_s)\}_{s=1}^t\right)$ is the filtration generated by the past observations. Additionally, since $w_t \in \{0, 1\}$ and x_t, a_t are both $\mathcal{F}_{t-1}$ measurable, we see that w_t can be written as

$$w_t = f_r(x_t, a_t) + \eta_t,$$

where η_t is $\mathcal{F}_{t-1}$-conditionally subGaussian. Therefore, we have met all the necessary conditions, and we can apply Theorem 2 of Chowdhury and Gopalan [2017] which gives us the desired result. □

This lemma jointly bounds the modeling error over the Borda function for all time t though it introduces a dependence on the RKHS norm of f_r. This dependence is inherited from prior work,

but we empirically study the relationship between the RKHS norm of a particular reward function and that of the associated Borda function in Section C.5.

We also adapt a result from Lemma 8 of Kandasamy et al. [2019] in order to understand the convergence of our uncertainty function σ_t.

Lemma 6. *Suppose we have n queries $(q_t)_{t=1}^n$ taken from $\mathcal{X} \times \mathcal{A}$. Then the posterior σ_t satisfies*

$$\sum_{q_t} \sigma_{t-1}^2(q_t) \leq \frac{2}{\log(1+\eta^{-2})} \Phi_n(\mathcal{X} \times \mathcal{A}).$$

Lemma 6 gives us a handle on how quickly we can expect the uncertainty function to shrink as additional datapoints are observed.

Now that we have lemmas 5 and 6 in place, we can proceed to the proof of the main result.

Proof. In this proof, we condition on the event in Lemma 5 holding true. Given that occurence, we

can say the following for every $x \in \mathcal{X}$.

$$\max_{a\in\mathcal{A}} r(x,a) - r(x,\hat{\pi}_N(s)) \overset{\text{Assumption 3}}{\leq} L_1 \left(\max_{a\in\mathcal{A}} f_r(x,a) - f_r(x,\hat{\pi}_N(x)) \right) \tag{C.62}$$

$$\overset{\text{Lemma 5}}{\leq} L_1 \left(\max_{a\in\mathcal{A}} f_r(x,a) - \max_{t\in[T]} \underline{f_r^t}(x,\hat{\pi}_N(x)) \right) \tag{C.63}$$

$$\overset{\text{Def. of } \hat{\pi}_N}{=} L_1 \left(\max_{a\in\mathcal{A}} f_r(x,a) - \max_{a\in\mathcal{A}} \max_{t\in[T]} \underline{f_r^t}(x,a) \right) \tag{C.64}$$

$$= L_1 \min_{t\in[T]} \left(\max_{a\in\mathcal{A}} f_r(x,a) - \max_{a\in\mathcal{A}} \underline{f_r^t}(x,a) \right) \tag{C.65}$$

$$\overset{\text{Lemma 5}}{\leq} L_1 \min_{t\in[T]} \left(\max_{a\in\mathcal{A}} \overline{f_r^t}(x,a) - \max_{a\in\mathcal{A}} \underline{f_r^t}(x,a) \right) \tag{C.66}$$

$$\overset{\text{Def. of } x^t}{\leq} L_1 \min_{t\in[T]} \left(\max_{a\in\mathcal{A}} \overline{f_r^t}(x^t,a) - \max_{a\in\mathcal{A}} \underline{f_r^t}(x^t,a) \right) \tag{C.67}$$

$$\overset{\text{Def. of } a^t}{\leq} L_1 \min_{t\in[T]} \left(\overline{f_r^t}(x^t,a^t) - \underline{f_r^t}(x^t,a^t) \right) \tag{C.68}$$

$$\leq \frac{L_1}{T} \sum_{t=1}^{T} \left(\overline{f_r^t}(x^t,a^t) - \underline{f_r^t}(x^t,a^t) \right) \tag{C.69}$$

$$= \frac{L_1}{T} \sum_{t=1}^{T} 2\beta_t^{(r)} \sigma_t(x^t,a^t) \tag{C.70}$$

$$\overset{\beta_t^{(r)} \text{ is increasing}}{\leq} \frac{2L_1\beta_T^{(r)}}{T} \sqrt{\left(\sum_{t=1}^{T} \sigma_t(x^t,a^t) \right)^2} \tag{C.71}$$

$$\overset{\text{Cauchy-Schwarz}}{\leq} \frac{2L_1\beta_T^{(r)}}{T} \sqrt{T \sum_{t=1}^{T} \sigma_t^2(x^t,a^t)} \tag{C.72}$$

$$\overset{\text{Lemma 6}}{\leq} \frac{2L_1\beta_T^{(r)}}{\sqrt{T}} \sqrt{C_1 \Phi_T} \tag{C.73}$$

$$\overset{\text{def of } \beta_T^{(r)}}{=} \frac{2L_1}{\sqrt{T}} (2B + \sqrt{2(\Phi_{t-1} + 1 + \log(2/\delta))}) \sqrt{C_1 \Phi_T} \tag{C.74}$$

$$= O\left(\frac{L_1}{\sqrt{T}} \left(B + \Phi_T \sqrt{\log \frac{1}{\delta}} \right) \right). \tag{C.75}$$

□

C.5 RKHS norms of r and f_r

In order to understand the dependence of our estimation bound on the RKHS norm $||f_r||_\kappa$, we ran numerical experiments on sampled reward functions. For a variety of context and action dimensions, we sampled 1000 reward functions as in Section 5.4.3 and numerically approximated their RKHS norms. We also made a Monte-Carlo estimate of the Borda function f_r for each of the reward functions sampled and numerically approximated its RKHS norm. To do this, we uniformly sample

Context Dimension	Action Dimension	Win Rate	Win Margin
0	1	0.16	-6.3
1	1	0.89	5.1
1	3	1	21.4
3	1	1	21.5
3	3	1	38.7
10	10	1	19.6

Table C.2: Comparison of RKHS norms of reward functions and associated Borda functions

1,000 points x_i from the input space, compute the regularized kernel matrix K for this set x_i, solve the KRR problem $K\alpha = f(x)$ for α. Then we compute the quadratic form $\sqrt{\alpha^T K \alpha}$ as an estimate of the RKHS norm.

In Table C.2, we present the results of comparing the RKHS norms of 1000 reward functions and their associated Borda functions sampled as in Section 5.4.3. A 'win' was counted when the Borda function had smaller RKHS norm and a 'loss' otherwise. The win margin is the average difference in RKHS norms of the reward and Borda functions, with a positive value when the Borda function was of smaller norm. It is clear here that in general (though not always) the RKHS norm of the Borda function f_r for a particular reward function r is smaller than the RKHS norm of the reward function r itself. This relationship seems to grow stronger as the input dimensionality of the reward function grows larger.

C.6 Additional Experiments for Kernelized Setting

In Figure C.1, we depict the progress of the AE-Bordamethod as it continually acquires data. One can see that the estimated optimal policy (red, second row) converges to a function quite similar to the ground truth (red, first row) as more data is collected. In addition, it is clear that the selection criterion targets parts of the domain which are relevant to policy learning while avoiding obviously bad regions. We also see in the fourth row that the uncertainty over the value function decreases relatively smoothly across the context space, supporting the idea that our method controls max-regret effectively.

C.7 The Jeopardy! preference dataset

We generated a set of plausible wrong answers for the Jeopardy! dataset from Huggingface [Wolf et al., 2023] by asking GPT-3.5 for a plausible wrong answer given the question, category, and answer. We found that both the category and correct answer were necessary to include to direct GPT-3.5 to generate an answer which was appropriate for the category and to prevent it from accidentally generating a correct answer. We give the prompt used for this process in Figure C.2.

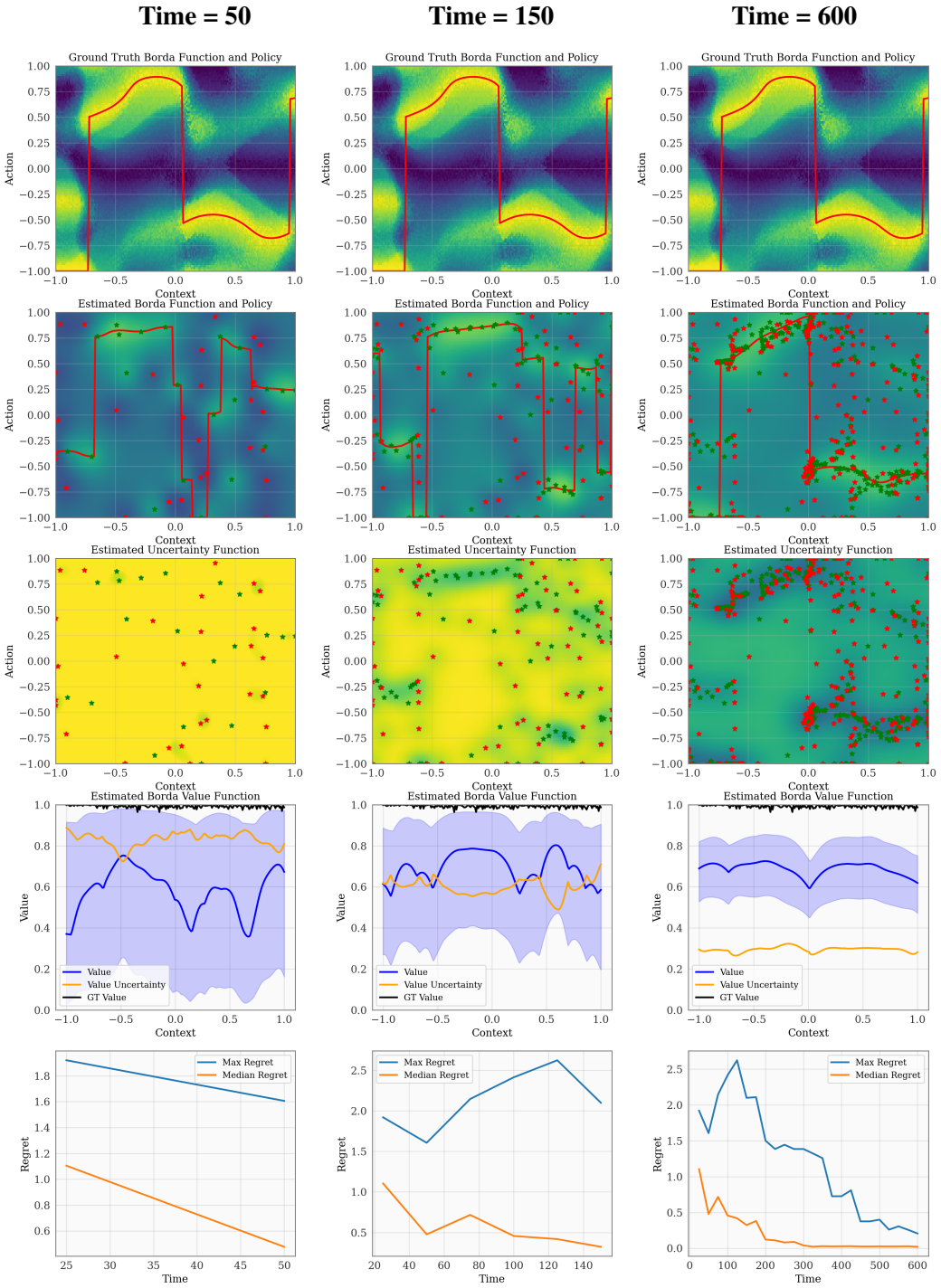

Figure C.1: Progress of AE-Borda across 50, 150, and 600 datapoints. From the top downwards, the charts show the ground truth function, the mean of the posterior estimate of f_r, the uncertainty function, the estimate of the value function as well as the acquisition function given in (5.3), and the regret over time.

```
[System]
You are an assistant coming up with plausible but incorrect answers to
Jeopardy questions (just the answer, no "what is"). Here's an example:\n
Q: 'For the last 8 years of his life, Galileo was under house arrest for
espousing this man's theory'
Category: HISTORY
Correct Answer: Copernicus\n
Response: Brahe
[User]
Q: {question}
Category: {category}
Correct Answer: {answer}
Response:
```

Figure C.2: The prompt used to collect plausible wrong answers for Jeopardy! questions.

C.8 Related Work on Uncertainty Estimation in Large Language Models

Estimating the epistemic uncertainty in large language models is still an active area of research and there are few prior works on this topic. For example, Osband et al. [2022] augment existing models with additional layers to model randomness, and subsequently the uncertainty. However performing uncertainty quantification in a parallelized fashion requires a significant memory overhead. To be more amenable to larger models, we instead use a dropout-augmented model to estimate uncertainty, as detailed in Section 5.5.

C.9 Prompt templates

The prompt templates for GPT-4 as the pairwise comparison evaluation judge and GPT-3.5 as the Jeopardy! single answer correctness judge are listed in Figures C.3 and C.4. We maintain the standardized prompts proved to be effective by Zheng et al. [2023].

C.10 Additional Experiment Details

We train our initial SFT models for 1 epoch on the SHP and HH dataset and 2 epochs on the new Jeopardy! dataset. We select the initial training period based on the amount of training after which we obtained a validation loss which had plateaued. We also find it reasonable to add a dropout layer before the penultimate linear layer since we find that adding a dropout layer not to negatively affect the performance in the SFT phase. To aid in fitting the model on our GPUs, we use QLoRa [Hu et al., 2021, Dettmers et al., 2023] with 4bit quantization for model weights and optimize using the 8-bit Lion optimizer [Chen et al., 2023]. For the methods with a reference model, we put the policy and the reference model on two separate GPUs. Further, we use dropout probability of $p = 0.05$, policy constraint weight $\gamma = 0.1$, an uncertainty bonus $\beta = 4$, a learning rate of 5×10^{-7}, an unlabeled batch size of 128, and a training batch size b of 32. We run all experiments with 3 random seeds. Our implementation was built atop the one provided by the authors of the DPO paper [Rafailov

```
[System]
Please act as an impartial judge and evaluate the quality of the
responses provided by two AI assistants to the user question displayed
below. You should choose the assistant that follows the user's
instructions and answers the user's question better. Your evaluation
should consider factors such as the helpfulness, relevance, accuracy,
depth, creativity, and level of detail of their responses. Avoid any
position biases and ensure that the order in which the responses were
presented does not influence your decision. Do not allow the length of
the responses to influence your evaluation. Do not favor certain names of
 the assistants. Be as objective as possible. Output your final verdict
by strictly following this format: 'A' if assistant A is better, 'B' if
assistant B is better, and 'C' for a tie. Output only that character and
do not include any other characters or spaces.

[User Question]
{question}

[The Start of Assistant A's Answer]
{answer_a}
[The End of Assistant A's Answer]

[The Start of Assistant B's Answer]
{answer_b}
[The End of Assistant B's Answer]
```

Figure C.3: The default prompt for pairwise comparison.

```
[System]
You are a judge on whether a contestant answer to Jeopardy is correct
given a correct answer. If you don't see the correct answer it is not
correct. Answer 'Yes' or 'No' is sufficient. Please don't use any other
words.

[The Start of Correct Answer]
{correct_answer}
[The End of Correct Answer]

[The Start of Contestant Answer]
{contestant_answer}
[The End of Contestant Answer]
```

Figure C.4: The default prompt for evaluating single Jeopardy! answer.

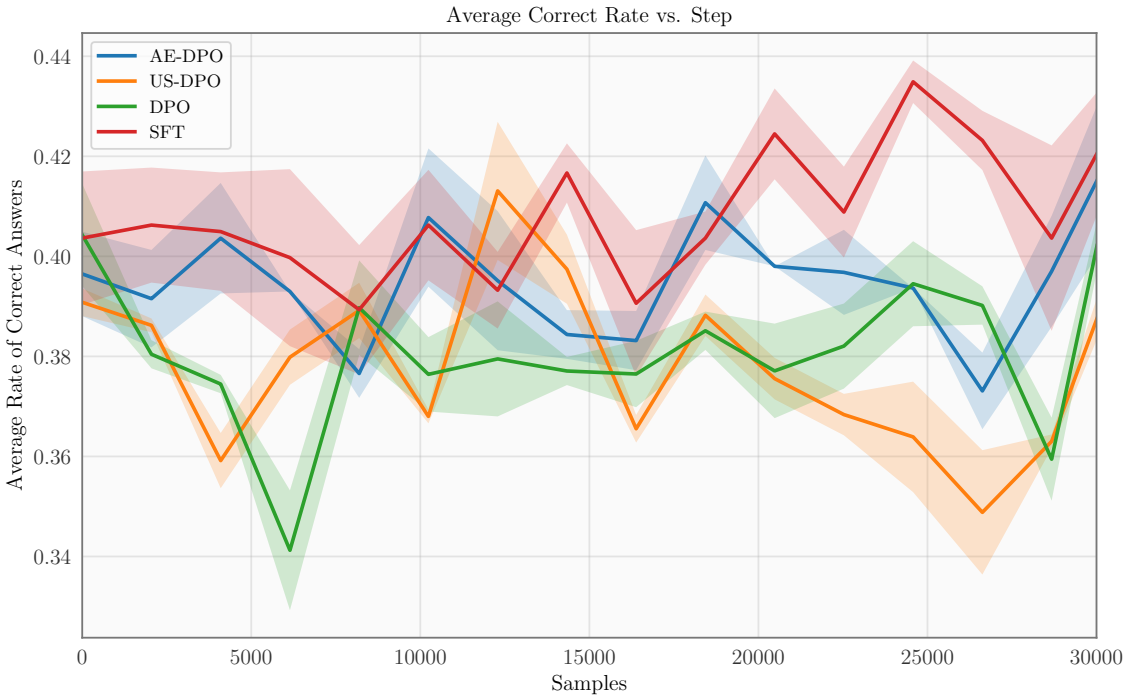

Figure C.5: Rate of correct answers for Jeopardy! over time.

et al., 2023].

C.11 Experiment Runtimes

	Jeopardy!	SHP	HH
Further SFT	2 | 3	4 | 4	3 | 7
DPO	7.5 | 25	10 | 14	10 | 15
US-DPO	8 | 12	79 | 85	31 | 85
AE-DPO	9 | 12	44 | 45	18 | 53

Table C.3: Runtimes (min | max) for each experiment rounded to nearest hour. Several experiments require a significant amount of compute time to complete. Runtimes vary depending on current loads on compute clusters.

C.12 Additional Experiments with LLM

Here, we plot the training curves for the Jeopardy! dataset below. For Jeopardy!, we plot the correctness of the policy over time in Figure C.5. Though this is part of the goal of the agent in the Jeopardy! dataset, note that it is not the entire optimization objective, as we show in Figure 5.4. Here, it is clear that no policy is able to improve at predicting correct answers on the test set. This is unsurprising as trivia is a difficult generalization problem.

C.12.1 Evaluating dropout-based LLM uncertainty estimation

We believe that in general the estimation of uncertainty for LLMs is an important topic of research and progress there will facilitate a more efficient and informed use of this technology. As we discussed in appendix C.8 and section 5.5, we use a dropout-based uncertainty estimation technique to inform the active exploration in this work. Over the course of this study, we considered ensembles and epistemic networks [Osband et al., 2022] as alternative methods for estimating the uncertainty of LLMs. However, each of these methods comes with some additional GPU memory requirement. For epistemic networks, the additional network parameters take GPU memory, while for ensembles, the memory is required to store multiple copies of a network or at least mutiple LoRAs. In our initial studies we found epistemic networks and dropout to perform comparably well and therefore chose dropout due to its smaller memory consumption and good performance. In this section, we explore whether the uncertainties predicted by our estimates differ when the model predicts the correct, incorrect, or null answer and whether these predictions differ in the cases when the model decides to predict null. To do this, we evaluated the log probabilities predicted by π_{SFT} on a test set of 20,560 Jeopardy! clues for the correct, incorrect, and null answer. We computed the sample variances over the log probabilities $\sigma^2(a \mid x) = \sum_{t_i \in a} \sigma^2(t_i \mid x, t_1, \ldots, t_{i-1})$ and plotted their densities in fig. C.6.

We see that the model predicts the highest variances for the log probabilities of incorrect answers. We also see that the the model seems to predict especially low variances for the null token when it decides to output it. The correct answer seems to have a lower variance when the model is willing to predict an answer. We see that the log probabilities of incorrect answers always have a high variance, indicating high uncertainty. We also see that the null token has a low variance when the model has a non-null output indicating certainty that it should not abstain. The variance further drops when it outputs null, indicating certainty about not knowing an answer. The correct answer has a lower variance than the incorrect answer when the model does not abstain. The relative variances of these two curves support that the model provides meaningful indications of uncertainty. Additionally, in the case where the model abstains, even the correct answer has a high variance, indicating a high uncertainty. We believe that these results support that the uncertainty function is at least correlated with the model's knowledge about the input. This offers support to the hypothesis that our estimates of the variance are somewhat meaningful. However, we believe that this is an important research topic and warrants substantial further study under a variety of lenses. We hope that this work will encourage further research in this area.

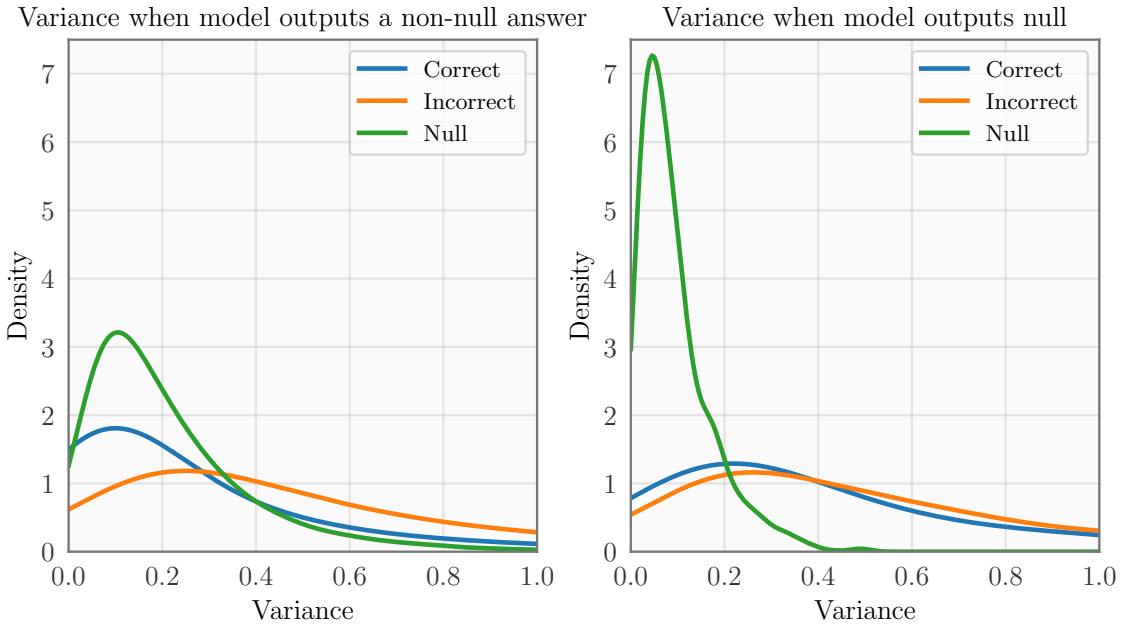

Figure C.6: Density of $\sigma(a \mid x)$ conditioned on correct, incorrect, and null values for a. The left hand plot depicts the variance distributions conditional on the model outputing a non-null completion, while the right hand is conditional on a null completion.

D | Appendix for part II

D.1 Experiment Details

D.1.1 Training Details

Each model takes 32 timesteps of state and control information as input and are trained on predictions for the following 16 timesteps. The ODE-based models are integrated from the initial conditions of the last given state. All neural networks are all trained with a learning rate of 3×10^{-3}, which was seen to work well across models. We generated a training set of 100,000 trajectories, test set of 20,000 trajectories, and validation set of 10,000 trajectories. Training was halted if validation error did not improve for 3 consecutive epochs.

D.1.2 Comparison Methods

We compare our models with other choices along the spectrum of structured to flexible models from both machine learning and system identification. In our paper, we compared the following methods in our experiments:

- **Full NDS**: A Neural Dynamical System with the full system dynamics for the problem being analyzed. The full construction of this model is given by Equation 7.2. For the functions $h_\theta, c_\vartheta, d_\tau$, we use fully connected networks with 2 layers, Softplus activations, 64 hidden nodes in each layer, and batch normalization.
- **Partial NDS**: A Neural Dynamical System with partial system dynamics for the problem being analyzed. These follow Equation 7.5 as applied to Equation 7.2. For the Ballistic system, we only provide equations for $\dot{x}$ and $\ddot{x}$, excluding the information about vertical motion from our network. For the Lorenz system, we only provide equations for $\dot{x}$ and $\dot{y}$, excluding information about motion in the z direction. For the Cartpole system, we only provide information about $\dot{\theta}$ and $\ddot{\theta}$. These equations were chosen somewhat arbitrarily to illustrate the partial NDS effectiveness. We use similar neural networks here as for the Full NDS.
- **NDS0**: A Full NDS with residual terms removed. This serves as an ablation which shows the use of the residual terms.
- **Fully Connected** (FC): A Fully-Connected Neural Network with 4 hidden layers containing 128 nodes with ReLU activations and batch normalization.
- **Fully Connected Neural ODE** (FC NODE): A larger version of the Neural ODE as given in Chen et al. [2018], we use 3 hidden layers with 128 nodes, batch norm, and Softplus activations for $\dot{s}$. This can be interpreted as a version of our NDS with no prior knowledge, i.e. $g(s) = 0$.

- **LSTM**: A stacked LSTM with 8 layers as in Graves [2013]. The data is fed in sequentially and we regress the outputs of the LSTM against the true values of the trajectory.
- **Gray Box Optimization** (GBO): We use MATLAB's gray-box system identification toolbox [Ljung et al., 2009] along with the prior knowledge ODEs to fit the parameters $\hat{\phi}$ as an alternative to using neural networks. This algorithm uses trust-region reflective nonlinear least squares with finite differencing [Coleman and Li, 1996] to find the parameter values which minimize the error of the model rollouts over the observed data.
- **Sparse Identification of Nonlinear Systems** (SR): We use the method from Brunton et al. [2015] to identify the dynamical systems of interest. This method uses sparse symbolic regression to learn a linear mapping from basis functions of the state x_t and control u_t to the derivatives $\dot{x}_t$ computed by finite differences. Our synthetic systems are in the span of the polynomial basis that we used.
- **APHYNITY**: We use the method from Yin et al. [2021], which fits a min-error parameter but then has an additional neural network component to model unknown dynamics.

Bibliography

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

Yasin Abbasi-Yadkori. Online learning for linearly parametrized control problems. 2012.

Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. *Advances in neural information processing systems*, 24, 2011.

Yasin Abbasi-Yadkori, Peter Bartlett, Kush Bhatia, Nevena Lazic, Csaba Szepesvari, and Gellért Weisz. Politex: Regret bounds for policy iteration using expert prediction. In *International Conference on Machine Learning*, pages 3692–3702. PMLR, 2019.

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

Joseph Abbate, R Conlin, and E Kolemen. Data-driven profile prediction for diii-d. *Nuclear Fusion*, 61(4):046027, 2021b.

Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information fusion*, 76:243–297, 2021.

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=S1ANxQW0b.

Jan Achterhold and Joerg Stueckler. Explore the context: Optimal data collection for context-conditional dynamics models. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 3529–3537. PMLR, 13–15 Apr 2021. URL https://proceedings.mlr.press/v130/achterhold21a.html.

Alekh Agarwal, Nan Jiang, Sham M Kakade, and Wen Sun. Reinforcement learning: Theory and algorithms. 2019.

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, editors, *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pages 67–83. PMLR, 09–12 Jul 2020. URL https://proceedings.mlr.press/v125/agarwal20b.html.

Riad Akour. Robust preference learning-based reinforcement learning. 2014.

M. Ariola, G. De Tommasi, A. Pironti, and F. Villone. Control of resistive wall modes in tokamak plasmas. *Control Engineering Practice*, 24:15–24, 2014. ISSN 0967-0661. doi: https://doi.org/10.1016/j.conengprac.2013.11.009. URL https://www.sciencedirect.com/science/article/pii/S096706611300213X.

Marco Ariola, Alfredo Pironti, et al. *Magnetic control of tokamak plasmas*, volume 187. Springer, 2008.

Dilip Arumugam and Benjamin Van Roy. The value of information when deciding what to learn, 2021.

Jordan T. Ash, Cyril Zhang, Surbhi Goel, Akshay Krishnamurthy, and Sham M. Kakade. Anti-concentrated confidence bonuses for scalable exploration. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=RXQ-FPbQYVn.

Peter Auer and Ronald Ortner. Logarithmic online regret bounds for undiscounted reinforcement learning. *Advances in neural information processing systems*, 19, 2006.

Peter Auer, Thomas Jaksch, and Ronald Ortner. Near-optimal regret bounds for reinforcement learning. *Advances in neural information processing systems*, 21, 2008.

I. Ayed, E. de Bézenac, A. Pajot, J. Brajard, and P. Gallinari. Learning dynamical systems from partial observations. *arXiv:1902.11136 [physics]*, February 2019.

Alex Ayoub, Zeyu Jia, Csaba Szepesvari, Mengdi Wang, and Lin Yang. Model-based reinforcement learning with value-targeted regression. In *International Conference on Machine Learning*, pages 463–474. PMLR, 2020.

Mohammad Gheshlaghi Azar, Rémi Munos, and Bert Kappen. On the sample complexity of reinforcement learning with a generative model. *arXiv preprint arXiv:1206.6461*, 2012.

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.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

Maximilian Balandat, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew G Wilson, and Eytan Bakshy. Botorch: a framework for efficient monte-carlo bayesian optimization. *Advances in neural information processing systems*, 33:21524–21538, 2020.

Philip Ball, Jack Parker-Holder, Aldo Pacchiano, Krzysztof Choromanski, and Stephen Roberts. Ready policy one: World building through active learning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119

of *Proceedings of Machine Learning Research*, pages 591–601. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/ball20a.html.

E. A. Baltz, E. Trask, M. Binderbauer, M. Dikovsky, H. Gota, R. Mendoza, J. C. Platt, and P. F. Riley. Achievement of sustained net plasma heating in a fusion experiment with the optometrist algorithm. *Scientific Reports*, 7(1):6425, December 2017. ISSN 2045-2322. doi: 10.1038/s41598-017-06645-7. URL http://www.nature.com/articles/s41598-017-06645-7.

Matteo Barbarino. A brief history of nuclear fusion. *Nature Physics*, 16(9):890–893, 2020.

Stephane RA Barde, Soumaya Yacout, and Hayong Shin. Optimal preventive maintenance policy based on reinforcement learning of a fleet of military trucks. *Journal of Intelligent Manufacturing*, 30(1):147–161, 2019.

Jayson Barr, Brian Sammuli, Dave A Humphreys, E Olofsson, XD Du, Cristina Rea, Will P Wehner, Mark D Boyer, Nicholas W Eidietis, Robert Granetz, et al. Development and experimental qualification of novel disruption prevention techniques on diii-d. *Nuclear Fusion*, 61(12):126019, 2021.

Jonathan Bassen, Bharathan Balaji, Michael Schaarschmidt, Candace Thille, Jay Painter, Dawn Zimmaro, Alex Games, Ethan Fast, and John C Mitchell. Reinforcement learning for the adaptive scheduling of educational activities. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2020.

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.

Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

Richard Bellman, Robert Kalaba, and Bella Kotkin. Polynomial approximation–a new computational technique in dynamic programming: Allocation processes. *Mathematics of Computation*, 17(82): 155–161, 1963.

Viktor Bengs, Robert Busa-Fekete, Adil El Mesaoudi-Paul, and Eyke Hüllermeier. Preference-based online learning with dueling bandits: A survey, 2021.

Ilija Bogunovic and Andreas Krause. Misspecified gaussian process bandit optimization. *Advances in Neural Information Processing Systems*, 34:3004–3015, 2021.

Ilija Bogunovic, Jonathan Scarlett, Stefanie Jegelka, and Volkan Cevher. Adversarially robust optimization with gaussian processes. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

A. Bondeson and D.J. Ward. Stabilization of external modes in tokamaks by resistive walls and plasma rotation. *Physical Review Letters*, 72(17):2709–2712, 1994.

Dan Boyer, Keith Erickson, Stanley Kaye, Vaish Gajaraj, Justin Kunimune, and Michael Zarnstorff. Nubeamnet: Accelerated predictive modeling of nstx-u beam deposition for optimization and control. In *APS Division of Plasma Physics Meeting Abstracts*, volume 2018, pages TP11–102, 2018.

M. D. Boyer, K. G. Erickson, B. A. Grierson, D. C. Pace, J. T. Scoville, J. Rauch, B. J. Crowley, J. R. Ferron, S. R. Haskey, D. A. Humphreys, R. Johnson, R. Nazikian, and C. Pawley. Feedback control of stored energy and rotation with variable beam energy and perveance on diii-d. *Nuclear Fusion*, 59(7):076004, May 2019a. ISSN 0029-5515.

M. D. Boyer, S. Kaye, and K. Erickson. Real-time capable modeling of neutral beam injection on nstx-u using neural networks. *Nuclear Fusion*, 59(5):056008, March 2019b. ISSN 0029-5515.

MD Boyer, C Rea, and M Clement. Toward active disruption avoidance via real-time estimation of the safe operating region and disruption proximity in tokamaks. *Nuclear Fusion*, 62(2):026005, 2021.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

Mary Ann Branch, Thomas F Coleman, and Yuying Li. A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems. *SIAM Journal on Scientific Computing*, 21(1):1–23, 1999.

Franklin H Branin. Widely convergent method for finding multiple solutions of simultaneous nonlinear equations. *IBM Journal of Research and Development*, 16(5):504–522, 1972.

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.

William J Broad. Breakthrough in nuclear fusion offers hope for power of future. *The New York Times*, page 1, Nov 1991.

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

C. Brose. *The Kill Chain: Defending America in the Future of High-Tech Warfare*. Hachette Books, 2020. ISBN 9780316533362. URL https://books.google.com/books?id=CW-nDwAAQBAJ.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020a. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020b.

S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data: Sparse identification of nonlinear dynamical systems. September 2015. doi: 10.1073/pnas.1517384113.

Mona Buisson-Fenet, Friedrich Solowjow, and Sebastian Trimpe. Actively learning gaussian process dynamics. In Alexandre M. Bayen, Ali Jadbabaie, George Pappas, Pablo A. Parrilo, Benjamin Recht, Claire Tomlin, and Melanie Zeilinger, editors, *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, pages 5–15. PMLR, 10–11 Jun 2020. URL https://proceedings.mlr.press/v120/buisson-fenet20a.html.

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.

E. F. Camacho and C. B. Alba. *Model predictive control*. Springer Science & Business Media, 2013.

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

Ian Char, Youngseog Chung, Willie Neiswanger, Kirthevasan Kandasamy, Andrew Oakleigh Nelson, Mark Boyer, Egemen Kolemen, and Jeff Schneider. Offline contextual bayesian optimization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/7876acb66640bad41f1e1371ef30c180-Paper.pdf.

Ian Char, Joseph Abbate, László Bardóczi, Mark D Boyer, Youngseog Chung, Rory Conlin, Keith Erickson, Viraj Mehta, Nathan Richner, Egemen Kolemen, and Jeff Schneider. Offline model-based reinforcement learning for tokamak control. In *Learning for Dynamics and Control*, pages 1–16. PMLR, 2023.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*, 2017a.

Francis F Chen et al. *Introduction to plasma physics and controlled fusion*, volume 1. Springer, 1984.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

R. T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *Neurips*, 2018. arXiv: 1806.07366.

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

S. Chen, S. A. Billings, and P. M. Grant. Non-linear system identification using neural networks. *International Journal of Control*, 51(6):1191–1214, January 1990. ISSN 0020-7179. doi: 10.1080/00207179008934126.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, et al. Symbolic discovery of optimization algorithms. *arXiv preprint arXiv:2302.06675*, 2023.

Z. Chen, J. Zhang, Martin Arjovsky, and Léon Bottou. Symplectic recurrent neural networks. *arXiv:1909.13334 [cs, stat]*, September 2019. arXiv: 1909.13334.

W. S. Cho, P. Zhang, Y. Zhang, X. Li, M. Galley, C. Brockett, M. Wang, and J. Gao. Towards coherent and cohesive long-form text generation. 2018.

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

Paul Christiano, Jan Leike, Tom B Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinformcement learning from human preferences. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, 2017. URL https://papers.nips.cc/paper_files/paper/2017/hash/d5e2c0adad503c91f91df240d0cd4e49-Abstract.html.

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, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/3de568f8597b94bda53149c7d7f5958c-Paper.pdf.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Daniel Clery. Explosion marks laser fusion breakthrough. *Science*, 378(6625):1154, 2022.

T. F. Coleman and Y. Li. An interior trust region approach for nonlinear minimization subject to bounds. *SIAM Journal on Optimization*, 6(2):418–445, May 1996.

Rory Conlin, Keith Erickson, Joseph Abbate, and Egemen Kolemen. Keras2c: A library for converting keras neural networks to real-time compatible c. *Engineering Applications of Artificial Intelligence*, 100:104182, 2021.

N. Cressie and C. K. Wikle. *Statistics for Spatio-Temporal Data*. John Wiley & Sons, November 2015. ISBN 978-1-119-24304-5.

Sebastian Curi, Felix Berkenkamp, and Andreas Krause. Efficient model-based reinforcement learning through optimistic policy search and planning. In *NeurIPS*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/a36b598abb934e4528412e5a2127b931-Abstract.html.

James W Daniel. Splines and efficiency in dynamic programming. *Journal of Mathematical Analysis and Applications*, 54(2):402–407, 1976.

E. de Bezenac, A. Pajot, and P. Gallinari. Deep learning for physical processes: Incorporating prior scientific knowledge. *arXiv:1711.07970 [cs, stat]*, January 2018. arXiv: 1711.07970.

Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.

PC De Vries, MF Johnson, B Alper, P Buratti, TC Hender, HR Koslowski, V Riccardo, JET-EFDA Contributors, et al. Survey of disruption causes at jet. *Nuclear fusion*, 51(5):053018, 2011.

Peter C de Vries, Timothy C Luce, YS Bae, S Gerhardt, Xianzu Gong, Yuri Gribov, D Humphreys, A Kavin, RR Khayrutdinov, C Kessel, et al. Multi-machine analysis of termination scenarios with comparison to simulations of controlled shutdown of iter discharges. *Nuclear Fusion*, 58(2): 026019, 2017.

Richard Dearden, Nir Friedman, and Stuart Russell. Bayesian q-learning. In *Aaai/iaai*, pages 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.

Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897): 414–419, 2022.

Morris H DeGroot. Uncertainty, information, and sequential experiments. *The Annals of Mathematical Statistics*, 33(2):404–419, 1962.

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, page 465–472, Madison, WI, USA, 2011. Omnipress. ISBN 9781450306195.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms, 2023.

Omar Darwiche Domingues, Pierre Ménard, Matteo Pirotta, Emilie Kaufmann, and Michal Valko. Kernel-based reinforcement learning: A finite-time analysis. In *International Conference on Machine Learning*, pages 2783–2792. PMLR, 2021.

J. R. Dormand and P. J. Prince. A family of embedded runge-kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26, March 1980.

Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin. Catboost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363*, 2018.

Simon Du, Sham Kakade, Jason Lee, Shachar Lovett, Gaurav Mahajan, Wen Sun, and Ruosong Wang. Bilinear classes: A structural framework for provable generalization in rl. In *International Conference on Machine Learning*, pages 2826–2836. PMLR, 2021.

Simon Shaolei Du, Sham M. Kakade, Ruosong Wang, and Lin F. Yang. Is a good representation sufficient for sample efficient reinforcement learning? *International Conference on Learning Representations*, 2020.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Miroslav Dudík, Katja Hofmann, Robert E. Schapire, Aleksandrs Slivkins, and Masrour Zoghi. Contextual dueling bandits, 2015.

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

Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12873–12883, 2021.

Kawin Ethayarajh, Yejin Choi, and Swabha Swayamdipta. Understanding dataset difficulty with $\mathcal{V}$-usable information. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 5988–6008. PMLR,

17–23 Jul 2022.

Mark R Fahey and Jeff Candy. Gyro: A 5-d gyrokinetic-maxwell solver. In *SC'04: Proceedings of the 2004 ACM/IEEE Conference on Supercomputing*, pages 26–26. IEEE, 2004.

Max E Fenstermacher, J Abbate, S Abe, T Abrams, M Adams, B Adamson, N Aiba, T Akiyama, P Aleynikov, E Allen, et al. Diii-d research advancing the physics basis for optimizing the tokamak approach to fusion energy. *Nuclear Fusion*, 62(4):042024, 2022.

J. R. Ferron, M. L. Walker, L. L. Lao, H. E. St John, D. A. Humphreys, and J. A. Leuer. Real time equilibrium reconstruction for tokamak discharge control. *Nuclear Fusion*, 38(7):1055–1066, July 1998.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

Dan Foreman-Mackey. Tinygp, 2021. URL https://tinygp.readthedocs.io.

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

Peter I. Frazier. A tutorial on bayesian optimization. *arXiv:1807.02811 [cs, math, stat]*, July 2018. URL http://arxiv.org/abs/1807.02811. arXiv: 1807.02811.

Peter I Frazier, Warren B Powell, and Savas Dayanik. A knowledge-gradient policy for sequential information collection. *SIAM Journal on Control and Optimization*, 47(5):2410–2439, 2008.

Jeffrey P. Freidberg. *Plasma Physics and Fusion Energy*. Cambridge University Press, 2007. doi: 10.1017/CBO9780511755705.

J. Frøyland and K. H. Alfsen. Lyapunov-exponent spectra for the lorenz model. *Physical Review A*, 29(5):2928–2931, May 1984.

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

Yichen Fu, David Eldon, Keith Erickson, Kornee Kleijwegt, Leonard Lupin-Jimenez, Mark D. Boyer, Nick Eidietis, Nathaniel Barbour, Olivier Izacard, and Egemen Kolemen. Machine learning control for disruption and tearing mode avoidance. *Physics of Plasmas*, 27(2):022501, February 2020b. ISSN 1070-664X. doi: 10.1063/1.5125581. URL http://aip.scitation.org/doi/full/10.1063/1.5125581. Publisher: American Institute of Physics.

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

Fusion Industry Association. The global fusion industry in 2023. Fusion Industry Association, July 2023.

Johannes Fürnkranz, Eyke Hüllermeier, Weiwei Cheng, and Sang-Hyeun Park. Preference-based reinforcement learning: a formal framework and a policy iteration algorithm. *Mach Learn*, 2012.

Victor Gabillon, Mohammad Ghavamzadeh, Alessandro Lazaric, and Sébastien Bubeck. Multi-bandit best arm identification. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran As-

sociates, Inc., 2011. URL https://proceedings.neurips.cc/paper/2011/file/c4851e8e264415c4094e4e85b0baa7cc-Paper.pdf.

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

Manuel Garcia-Munoz, SE Sharapov, MA Van Zeeland, Enrique Ascasibar, A Cappa, L Chen, J Ferreira, Joaquin Galdon-Quiroga, Benedikt Geiger, J Gonzalez-Martin, et al. Active control of alfvén eigenmodes in magnetically confined toroidal plasmas. *Plasma Physics and Controlled Fusion*, 61(5):054007, 2019.

Jacob R Gardner, Geoff Pleiss, David Bindel, Kilian Q Weinberger, and Andrew Gordon Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, 2018.

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.

M Giacomin, A Pau, P Ricci, O Sauter, T Eich, JET Contributors, ASDEX Upgrade Team, et al. First-principles density limit scaling in tokamaks based on edge turbulent transport and implications for iter. *Physical Review Letters*, 128(18):185003, 2022.

A. Graves. Generating sequences with recurrent neural networks. August 2013. URL https://arxiv.org/abs/1308.0850v5.

Martin Greenwald, JL Terry, SM Wolfe, S Ejima, MG Bell, SM Kaye, and GH Neilson. A new look at density limits in tokamaks. *Nuclear Fusion*, 28(12):2199, 1988.

R. J. Groebner, K. H. Burrell, and R. P. Seraydarian. Role of edge electric field and poloidal rotation in the l-h transition. *Physical Review Letters*, 64(25):3015–3018, 1990. ISSN 00319007. doi: 10.1103/PhysRevLett.64.3015.

Peter D Grünwald and A Philip Dawid. Game theory, maximum entropy, minimum discrepancy and robust bayesian decision theory. *Ann. Stat.*, 32(4):1367–1433, August 2004.

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, pages 1025–1033, Red Hook, NY, USA, 2012. Curran Associates Inc.

Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Thomas Paine, Sergio Gómez, Konrad Zolna, Rishabh Agarwal, Josh S Merel, Daniel J Mankowitz, Cosmin Paduraru, et al. Rl unplugged: A suite of benchmarks for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:7248–7259, 2020.

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

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant.

Array programming with numpy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

S. R. Haskey, B. A. Grierson, L. Stagner, C. Chrystal, A. Ashourvan, A. Bortolon, M. D. Boyer, K. H. Burrell, C. Collins, R. J. Groebner, D. H. Kaplan, and N. A. Pablant. Active spectroscopy measurements of the deuterium temperature, rotation, and density from the core to scrape off layer on the diii-d tokamak (invited). *Review of Scientific Instruments*, 89(10):10D110, August 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Tim Head, Manoj Kumar, Holger Nahrstaedt, Gilles Louppe, and Iaroslav Shcherbatyi. scikit-optimize/scikit-optimize, October 2021. URL https://doi.org/10.5281/zenodo.5565057.

Philipp Hennig and Christian J Schuler. Entropy search for information-efficient global optimization. *J. Mach. Learn. Res.*, 13(57):1809–1837, 2012.

Robin Henry and Damien Ernst. Gym-anm: Reinforcement learning environments for active network management tasks in electricity distribution systems. *Energy and AI*, 5:100092, 2021. ISSN 2666-5468. doi: https://doi.org/10.1016/j.egyai.2021.100092. URL https://www.sciencedirect.com/science/article/pii/S266654682100046X.

José Miguel Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. *Advances in neural information processing systems*, 27, 2014.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, November 1997.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack William Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=iBBcRUlOAPR.

EM Hollmann, PB Aleynikov, Tünde Fülöp, DA Humphreys, VA Izzo, M Lehnen, VE Lukash, Gergely Papp, G Pautasso, F Saint-Laurent, et al. Status of research toward the iter disruption mitigation system. *Physics of Plasmas*, 22(2), 2015.

Norbert Holtkamp, ITER Project Team, et al. An overview of the iter project. *Fusion Engineering and Design*, 82(5-14):427–434, 2007.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

Pihe Hu, Yu Chen, and Longbo Huang. Nearly minimax optimal reinforcement learning with linear function approximation. In *ICML*, 2022.

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. *Advances in Neural Information Processing Systems*, 33:15931–15941, 2020.

DA Humphreys, RD Deranian, JR Ferron, AW Hyatt, RD Johnson, RR Khayrutdinov, RJ La Haye, JA Leuer, BG Penaflor, JT Scoville, et al. Integrated plasma control in diii-d. *Fusion science and technology*, 48(2):1249–1263, 2005.

IH Hutchinson, R Boivin, F Bombarda, P Bonoli, S Fairfax, C Fiore, J Goetz, S Golovato, R Granetz, M Greenwald, et al. First results from alcator-c-mod. *Physics of Plasmas*, 1(5):1511–1518, 1994.

Robert B Jackson, Corinne Le Quéré, RM Andrew, Josep G Canadell, Jan Ivar Korsbakken, Zhu Liu, Glen P Peters, and Bo Zheng. Global energy growth is outpacing decarbonization. *Environmental Research Letters*, 13(12):120401, 2018.

M. Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Neurips*, 2019a. arXiv: 1906.08253.

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, 2019b.

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

H.ST. John. Equations and associated definitions used in onetwo, 2005.

Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455, 1998.

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

Shivaram Kalyanakrishnan, Ambuj Tewari, Peter Auer, and Peter Stone. Pac subset selection in stochastic multi-armed bandits. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, page 227–234, Madison, WI, USA, 2012. Omnipress. ISBN 9781450312851.

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

Kirthevasan Kandasamy, Gautam Dasarathy, Junier Oliva, Jeff Schneider, and Barnabas Poczos. Multi-fidelity gaussian process bandit optimisation. *Journal of Artificial Intelligence Research*, 66:151–196, 2019.

Kirthevasan Kandasamy, Karun Raju Vysyaraju, Willie Neiswanger, Biswajit Paria, Christopher R Collins, Jeff Schneider, Barnabas Poczos, and Eric P Xing. Tuning hyperparameters without grad students: Scalable and robust bayesian optimisation with dragonfly. *The Journal of Machine Learning Research*, 21(1):3098–3124, 2020.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

J. Kates-Harbeck, A. Svyatkovskiy, and W. Tang. Predicting disruptive instabilities in controlled

fusion plasmas through deep learning. *Nature*, 568(7753):526–531, April 2019. ISSN 1476-4687.

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.

CE Kessel, G Giruzzi, ACC Sips, RV Budny, JF Artaud, V Basiuk, F Imbeaux, E Joffrin, M Schneider, M Murakami, et al. Simulation of the hybrid and steady state advanced operating modes in iter. *Nuclear Fusion*, 47(9):1274, 2007.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

Johannes Kirschner, Ilija Bogunovic, Stefanie Jegelka, and Andreas Krause. Distributionally robust bayesian optimization. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2174–2184. PMLR, 26–28 Aug 2020. URL https://proceedings.mlr.press/v108/kirschner20a.html.

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

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Andreas Krause and Cheng Ong. Contextual gaussian process bandit optimization. *Advances in neural information processing systems*, 24, 2011.

J. Kreutzer, S. Khadivi, E. Matusov, and S Riezler. Can neural machine translation be improved with user feedback? 2018.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

LL Lao, HE St John, Q Peng, JR Ferron, EJ Strait, TS Taylor, WH Meyer, C Zhang, and KI You. Mhd equilibrium reconstruction in the diii-d tokamak. *Fusion science and technology*, 48(2): 968–977, 2005.

Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

Tor Lattimore, Csaba Szepesvari, and Gellert Weisz. Learning with good feature representations in bandits and in rl with a generative model. In *International Conference on Machine Learning*, pages 5662–5670. PMLR, 2020.

C. Lawrence and S. Riezler. Improving a neural semantic parser by counterfactual learning from human bandit feedback. 2018.

Nevena Lazic, Craig Boutilier, Tyler Lu, Eehern Wong, Binz Roy, MK Ryu, and Greg Imwalle. Data center cooling using model-predictive control. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/059fdcd96baeb75112f09fa1dcc740cc-Paper.pdf.

Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. In *International Conference on Machine Learning*, pages 6131–6141. PMLR, 2021.

Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International journal of robotics research*, 37(4-5):421–436, 2018.

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

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

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, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 12861–12872. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/96ea64f3a1aa2fd00c72faacf0cb8ac9-Paper.pdf.

Xiang Li, Viraj Mehta, Johannes Kirschner, Ian Char, Willie Neiswanger, Jeff Schneider, Andreas Krause, and Ilija Bogunovic. Near-optimal policy identification in active reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=3OR2tbtnYC-.

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.

David Lindner, Matteo Turchetta, Sebastian Tschiatschek, Kamil Ciosek, and Andreas Krause. Information directed reward learning for reinforcement learning. 35, 2021.

Shuang Liu and Hao Su. Provably efficient kernelized q-learning. *arXiv preprint arXiv:2204.10349*, 2022.

Lennart Ljung, Rajiv Singh, Qinghua Zhang, Peter Lindskog, and Anatoli Iouditski. Developments in the mathworks system identification toolbox. *IFAC Proceedings Volumes*, 42(10):522–527, 2009. ISSN 14746670.

N.C. Logan, T.M. Wilks, and the DIII-D NBI Operators. Diii-d nbi operator handbook, 2023. URL https://docs.google.com/document/d/1FMhkQraW0jc0HSzzQrmkk587ajEB6U19rUo3GJvc5BM/edit. Accessed online 2023-11-15.

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, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/a0a080f42e6f13b3a2df133f073095dd-Paper.pdf.

E. N. Lorenz. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20(2):130–141, March 1963.

Xiuyuan Lu, Benjamin Van Roy, Vikranth Dwaracherla, Morteza Ibrahimi, Ian Osband, Zheng Wen, et al. Reinforcement learning, bit by bit. *Foundations and Trends® in Machine Learning*, 16(6): 733–865, 2023.

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

Andrey Malinin, Liudmila Prokhorenkova, and Aleksei Ustimenko. Uncertainty in gradient boosting via ensembles. *International Conference on Learning Representations*, 2021.

G. Manek and J. Z. Kolter. Learning stable deep dynamics models. January 2020.

Andrew D Maris, Allen Wang, Cristina Rea, Robert Granetz, and Earl Marmar. The impact of disruptions on the economics of a tokamak power plant. *Fusion Science and Technology*, pages 1–17, 2023.

Viraj Mehta, Ian Char, Willie Neiswanger, Youngseog Chung, Andrew Nelson, Mark Boyer, Egemen Kolemen, and Jeff Schneider. Neural dynamical systems: Balancing structure and flexibility in physical prediction. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 3735–3742. IEEE, 2021.

Viraj Mehta, Ian Char, Joseph Abbate, Rory Conlin, Mark D Boyer, Stefan Ermon, Jeff Schneider, and Willie Neiswanger. Exploration via planning for information about the optimal trajectory. In *Advances in Neural Information Processing Systems*, volume 35, 2022a.

Viraj Mehta, Biswajit Paria, Jeff Schneider, Stefano Ermon, and Willie Neiswanger. An experimental design perspective on model-based reinforcement learning. In *International Conference on Learning Representations*, 2022b.

Viraj Mehta, Biswajit Paria, Jeff Schneider, Stefano Ermon, and Willie Neiswanger. An experimental design perspective on model-based reinforcement learning. In *International Conference on Learning Representations*, 2022c.

Viraj Mehta, Joseph Abbate, Allen Wang, Andrew Rothstein, Ian Char, Jeff Schneider, Egemen Kolemen, Cristina Rea, and Darren Garnier. Towards LLMs as operational copilots for fusion reactors. In *NeurIPS 2023 AI for Science Workshop*, 2023. URL https://openreview.net/forum?id=yGVChrbJ4E.

O. Meneghini, S. P. Smith, L. L. Lao, O. Izacard, Q. Ren, J. M. Park, J. Candy, Z. Wang, C. J. Luna, V. A. Izzo, B. A. Grierson, P. B. Snyder, C. Holland, J. Penna, G. Lu, P. Raum, A. McCubbin, D. M. Orlov, E. A. Belli, N. M. Ferraro, R. Prater, T. H. Osborne, A. D. Turnbull, and G. M. Staebler. Integrated modeling applications for tokamak experiments with omfit. *Nuclear Fusion*, 55(8):083008, July 2015.

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.

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.

M. C. Mozer and J. Bachrach. Discovering the structure of a reactive environment by exploration. In *Advances in Neural Information Processing Systems 2*, pages 439–446. 1990.

Jonas Močkus. On bayesian methods for seeking the extremum. In *Optimization techniques IFIP*

technical conference, pages 400–404. Springer, 1975.

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)*, pages 7559–7566. IEEE, 2018.

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*, pages 8005–8015. PMLR, 2021.

Willie Neiswanger, Lantao Yu, Shengjia Zhao, Chenlin Meng, and Stefano Ermon. Generalizing bayesian optimization with decision-theoretic entropies. In *Advances in Neural Information Processing Systems*, volume 36, 2022.

O. Nelles. *Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models*. Springer Science & Business Media, March 2013. ISBN 978-3-662-04323-3.

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.

OpenAI. GPT-4 technical report. abs/2303.08774, 2023. doi: 10.48550/arXiv.2303.08774. URL https://doi.org/10.48550/arXiv.2303.08774.

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

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. *Advances in neural information processing systems*, 29, 2016b.

Ian Osband, Benjamin Van Roy, Daniel J. Russo, and Zheng Wen. Deep exploration via randomized value functions. *Journal of Machine Learning Research*, 20(124):1–62, 2019. URL http://jmlr.org/papers/v20/18-339.html.

Ian Osband, Seyed Mohammad Asghari, Benjamin Van Roy, Nat McAleese, John Aslanides, and Geoffrey Irving. Fine-tuning language models via epistemic neural networks. *arXiv preprint arXiv:2211.01568*, 2022.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744, 2022.

Biswajit Paria, Barnabàs Pòczos, Pradeep Ravikumar, Jeff Schneider, and Arun Sai Suggala. Be greedy–a simple algorithm for blackbox optimization using neural networks. In *ICML2022 Workshop on Adaptive Experimental Design and Active Learning in the Real World*, 2022.

Matthew S Parsons. Interpretation of machine-learning-based disruption models for plasma control. *Plasma Physics and Controlled Fusion*, 59(8):085001, 2017.

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

PMLR, 2017.

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

Michael Pearce and Juergen Branke. Continuous multi-task bayesian optimisation with correlation. *European Journal of Operational Research*, 270(3):1074–1085, 2018.

Michael Pearce, Janis Klaise, and Matthew Groves. Practical bayesian optimization of objectives with conditioning variables. *arXiv preprint arXiv:2002.09996*, 2020.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

E. Perez, S. Karamcheti, R. Fergus, J. Weston, D. Kiela, and K Cho. Finding generalizable evidence by learning to convince q&a models. 2019.

Victor Picheny, Tobias Wagner, and David Ginsbourger. A benchmark of kriging-based infill criteria for noisy optimization. *Structural and multidisciplinary optimization*, 48(3):607–626, 2013.

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.

Alfredo Pironti and Michael Walker. Fusion, tokamaks, and plasma control: an introduction and tutorial. *IEEE Control Systems Magazine*, 25(5):30–43, 2005.

G. D. Portwood, Peetak P. Mitra, Mateus Dias Ribeiro, Tan Minh Nguyen, Balasubramanya T. Nadiga, Juan A. Saenz, Michael Chertkov, Animesh Garg, Anima Anandkumar, Andreas Dengel, Richard Baraniuk, and David P. Schmidt. Turbulence forecasting via neural ode. *arXiv:1911.05180 [physics]*, November 2019. arXiv: 1911.05180.

Apostolos F Psaros, Xuhui Meng, Zongren Zou, Ling Guo, and George Em Karniadakis. Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons. *Journal of Computational Physics*, 477:111902, 2023.

C. Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, and Ali Jasim Ramadhan. Universal differential equations for scientific machine learning. *CoRR*, abs/2001.04385, 2020. URL https://arxiv.org/abs/2001.04385.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model,

2023.

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

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5331–5340. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/rakelly19a.html.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022. URL https://arxiv.org/abs/2204.06125.

Shyam Sundhar Ramesh, Pier Giuseppe Sessa, Yifan Hu, Andreas Krause, and Ilija Bogunovic. Distributionally robust model-based reinforcement learning with large state spaces. *arXiv preprint arXiv:2309.02236*, 2023.

C Radhakrishna Rao. Convexity properties of entropy functions and analysis of diversity. *Lecture Notes-Monograph Series*, pages 68–77, 1984.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, Cambridge, Mass., 3. print edition, 2008. ISBN 978-0-262-18253-9. OCLC: 552376743.

Christina Rea, KJ Montes, KG Erickson, RS Granetz, and RA Tinguely. A real-time machine learning-based disruption predictor in diii-d. *Nuclear Fusion*, 59(9):096016, 2019.

P.-H Rebut. Iter: the first experimental fusion reactor. *Fusion Engineering and Design*, 27:3–16, 1995. ISSN 0920-3796. doi: https://doi.org/10.1016/0920-3796(95)90113-2. URL https://www.sciencedirect.com/science/article/pii/0920379695901132. Proceedings of the Third International Symposium on Fusion Nuclear Technology.

A. Reiman and D. Monticello. Tokamak error fields and locked modes. *Physics of Fluids B: Plasma Physics*, 3(8):2230–2235, 1991. doi: 10.1063/1.859640. URL https://doi.org/10.1063/1.859640.

Pablo Rodriguez-Fernandez, NT Howard, MJ Greenwald, AJ Creely, JW Hughes, JC Wright, C Holland, Y Lin, F Sciortino, Sparc Team, et al. Predictions of core plasma performance for the sparc tokamak. *Journal of Plasma Physics*, 86(5):865860503, 2020.

S. Ross and J. A. Bagnell. Agnostic system identification for model-based reinforcement learning. *arXiv:1203.1007 [cs, stat]*, July 2012.

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

S. H. Rudy, J. Nathan Kutz, and S. L. Brunton. Deep learning of dynamics and signal-noise decomposition with time-stepping constraints. *Journal of Computational Physics*, 396:483–506, November 2019.

C. Runge. Ueber die numerische auflösung von differentialgleichungen. *Mathematische Annalen*, 46(2):167–178, June 1895.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-

Fei. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014. URL http://arxiv.org/abs/1409.0575.

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, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/301ad0e3bd5cb1627a2044908a42fdc2-Paper.pdf.

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.

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.

G Sannazzaro, C Bachmann, DJ Campbell, S Chiocchio, JP Girard, Y Gribov, S Reyes, M Sugihara, E Tada, and N Taylor. Structural load specification for iter tokamak components. In *2009 23rd IEEE/NPSS Symposium on Fusion Engineering*, pages 1–4. IEEE, 2009.

Shlomo S Sawilowsky. New effect size rules of thumb. *Journal of modern applied statistical methods*, 8(2):26, 2009.

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*, pages 222–227, 1991.

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

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

Matthias Schultheis, Boris Belousov, Hany Abdulsamad, and Jan Peters. Receding horizon curiosity. In *Conference on robot learning*, pages 1278–1288. PMLR, 2020.

Paul J Schweitzer and Abraham Seidmann. Generalized polynomial approximations in markovian decision processes. *Journal of mathematical analysis and applications*, 110(2):568–582, 1985.

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*, pages 8583–8592. PMLR, 2020.

J Seo, Y-S Na, B Kim, CY Lee, MS Park, SJ Park, and YH Lee. Development of an operation trajectory design algorithm for control of multiple 0d parameters using deep reinforcement learning in kstar. *Nuclear Fusion*, 62(8):086049, 2022.

Jaemin Seo, Y-S Na, B Kim, CY Lee, MS Park, SJ Park, and YH Lee. Feedforward beta control in the kstar tokamak by deep reinforcement learning. *Nuclear Fusion*, 61(10):106010, 2021.

Jaemin Seo, Rory Conlin, Andrew Rothstein, SangKyeun Kim, Joseph Abbate, Azarakhsh Jalalvand, and Egemen Kolemen. Multimodal prediction of tearing instabilities in a tokamak. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2023.

Pier Giuseppe Sessa, Ilija Bogunovic, Maryam Kamgarpour, and Andreas Krause. No-regret learning

in unknown games with correlated payoffs. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

Pier Giuseppe Sessa, Ilija Bogunovic, Maryam Kamgarpour, and Andreas Krause. Mixed strategies for robust optimization of unknown objectives. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

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, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5779–5788. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/shyam19a.html.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

Chris Llewellyn Smith and Steve Cowley. The path to fusion power. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 368(1914):1091–1108, 2010.

Marta Soare, Alessandro Lazaric, and Remi Munos. Best-arm identification in linear bandits. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/f387624df552cea2f369918c5e1e12bc-Paper.pdf.

B. Sohlberg and E. W. Jacobsen. Grey box modelling – branches and experiences. *IFAC Proceedings Volumes*, 41(2):11415–11420, January 2008.

Benjamin Solnik, Daniel Golovin, Greg Kochanski, John Elliot Karro, Subhodeep Moitra, and D Sculley. Bayesian optimization for a better dessert. 2017.

Carl R Sovinec, AH Glasser, TA Gianakon, DC Barnes, RA Nebel, SE Kruger, DD Schnack, SJ Plimpton, A Tarditi, MS Chu, et al. Nonlinear magnetohydrodynamics simulation using high-order finite elements. *Journal of Computational Physics*, 195(1):355–386, 2004.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

N. Stiennon, L. Ouyang, J. Wu, D. M. Ziegler, R. Lowe, C. Voss, A. Radford, D. Amodei, and P. Christiano. Learning to summarize from human feedback. 2020.

JA Stillerman, TW Fredian, KA Klare, and G Manduchi. Mdsplus data acquisition system. *Review of Scientific Instruments*, 68(1):939–942, 1997.

Malcolm Strens. A bayesian framework for reinforcement learning. In *ICML*, volume 2000, pages 943–950, 2000.

Richard Sutton. The bitter lesson. *Incomplete Ideas (blog)*, 13(1), 2019.

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.

Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. *Advances in neural information processing systems*, 26, 2013.

Csaba Szepesvári. Lecture notes in reinforcement learning theory, Aug 2022. URL https://rltheory.github.io/lecture-notes/planning-in-mdps/lec13/.

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, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 2753–2762, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3a20f62a0af1aa152670bab3c602feed-Abstract.html.

AA Teplukhina, O Sauter, F Felici, A Merle, D Kim, TCV Team, ASDEX Upgrade Team, EUROfusion MST1 Team, et al. Simulation of profile evolution from ramp-up to ramp-down and optimization of tokamak plasma termination with the raptor code. *Plasma Physics and Controlled Fusion*, 59(12):124004, 2017.

Matthew Tesch, Jeff Schneider, and Howie Choset. Expensive function optimization with stochastic binary outcomes. In *International Conference on Machine Learning*, pages 1283–1291. PMLR, 2013.

Louis L Thurstone. The method of paired comparisons for social values. *The Journal of Abnormal and Social Psychology*, 21(4):384, 1927.

Michael Tinkham. *Introduction to superconductivity*. Courier Corporation, 2004.

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*, pages 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

Francis Troyon, R Gruber, H Saurenmann, S Semenzato, and S Succi. Mhd-limits to plasma confinement. *Plasma physics and controlled fusion*, 26(1A):209, 1984.

Alexander Tschantz, Beren Millidge, Anil K Seth, and Christopher L Buckley. Reinforcement learning through active inference. *arXiv preprint arXiv:2002.12636*, 2020.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

B. Ph. van Milligen, V. Tribaldos, and J. A. Jiménez. Neural network differential equation and

plasma equilibrium solver. *Physical Review Letters*, 75(20):3594–3597, November 1995.

Julien Villemonteix, Emmanuel Vazquez, and Eric Walter. An informational approach to the global optimization of expensive-to-evaluate functions. *Journal of Global Optimization*, 44:509–534, 2009.

P. Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C. J. Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1 0 Contributors. Scipy 1.0–fundamental algorithms for scientific computing in python. *arXiv:1907.10121 [physics]*, July 2019. arXiv: 1907.10121.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. Scipy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Fritz Wagner, G Becker, K Behringer, D Campbell, A Eberhagen, W Engelhardt, G Fussmann, O Gehre, J Gernhardt, G v Gierke, et al. Regime of improved confinement and high beta in neutral-beam-heated divertor discharges of the asdex tokamak. *Physical Review Letters*, 49(19): 1408, 1982.

Michael L Walker, Peter De Vries, Federico Felici, and Eugenio Schuster. Introduction to tokamak plasma control. In *2020 American Control Conference (ACC)*, pages 2901–2918. IEEE, 2020.

T. Wang and J. Ba. Exploring model-based planning with policy networks. *arXiv:1906.08649 [cs, stat]*, June 2019.

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.

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

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*, pages 10292–10302. PMLR, 2020.

Thomas Wolf, Lewis Tunstall, and Patrick von Platen. Jeopardy dataset, 2023.

Yichong Xu, Aparna Joshi, Aarti Singh, and Artur Dubrawski. Zeroth order non-convex optimization with dueling-choice bandits. In *Conference on Uncertainty in Artificial Intelligence*, pages 899–908. PMLR, 2020.

Jun Yang, Xinghui You, Gaoxiang Wu, Mohammad Mehedi Hassan, Ahmad Almogren, and Joze

Guna. Application of reinforcement learning in uav cluster task scheduling. *Future generation computer systems*, 95:140–148, 2019.

Zhuoran Yang, Chi Jin, Zhaoran Wang, Mengdi Wang, and Michael I Jordan. On function approximation in reinforcement learning: Optimism in the face of large state spaces. *arXiv preprint arXiv:2011.04622*, 2020.

Yuan Yin, Vincent Le Guen, Jérémie Dona, Emmanuel de Bézenac, Ibrahim Ayed, Nicolas Thome, and Patrick Gallinari. Augmenting physical models with deep networks for complex dynamics forecasting. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124012, 2021.

Y. Yue, J. Broder, R. Kleinberg, and T. Joachims. The k-armed dueling bandits problem. 2012.

Alexei Yumashev, Beata Ślusarczyk, Sergey Kondrashev, and Alexey Mikhaylov. Global indicators of sustainable development: Evaluation of the influence of the human development index on consumption and quality of energy. *Energies*, 13(11):2768, 2020.

Andrea Zanette, Alessandro Lazaric, Mykel Kochenderfer, and Emma Brunskill. Learning near optimal policies with low inherent bellman error. In *International Conference on Machine Learning*, pages 10978–10989. PMLR, 2020.

Jin Zhang, Jianhao Wang, Hao Hu, Tong Chen, Yingfeng Chen, Changjie Fan, and Chongjie Zhang. Metacure: Meta reinforcement learning with empowerment-driven exploration. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12600–12610. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/zhang21w.html.

Ze Jia Zhang and Karthikeyan Duraisamy. Machine learning methods for data-driven turbulence modeling. In *22nd AIAA computational fluid dynamics conference*, page 2460, 2015.

Jun Zhao, Guang Qiu, Ziyu Guan, Wei Zhao, and Xiaofei He. Deep reinforcement learning for sponsored search real-time bidding. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1021–1030, 2018.

Shengjia Zhao, Abhishek Sinha, Yutong He, Aidan Perreault, Jiaming Song, and Stefano Ermon. Comparing distributions by measuring differences that affect decision making. In *International Conference on Learning Representations*, 2021.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.

Dongruo Zhou, Quanquan Gu, and Csaba Szepesvari. Nearly minimax optimal reinforcement learning for linear mixture markov decision processes. In *Conference on Learning Theory*, pages 4532–4576. PMLR, 2021.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

L Zintgraf, K Shiarlis, M Igl, S Schulze, Y Gal, K Hofmann, and S Whiteson. Varibad: a very good method for bayes-adaptive deep rl via meta-learning. *Proceedings of ICLR 2020*, 2020.