SCALING UP DECISION-MAKING UNDER UNCERTAINTY

by

MOHIT RAJPAL

(B.Sc., University of Illinois at Urbana-Champaign, M.Sc., Columbia University)

A THESIS SUBMITTED FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF SINGAPORE

2024

Supervisor:
A/P Bryan Kian Hsiang Low

Examiners:
A/P Jonathan Mark Scarlett
Dr Bryan Hooi Kuen-Yew

Declaration

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Mohit RAJPAL

31 October 2024

Acknowledgments

I, foremost, would like to acknowledge my advisor, Dr. Bryan Kian Hsiang Low.

Many other teachers strived and collaborated to get me to this moment in my life. Some of my fondest memories of secondary school were of attending the classes taught by Dr. Richard DeCoster, Mr. Matthew Fahrenbacher, Dr. Ami LeFevre, Ms. Sharon Mikula, and Mr. Paul Wack.

The guidance and thought provoking questions asked by Dr. Feynman, Dr. Oppenheimer, and Dr. Sagan always helped me think about questions as a researcher and scientist in the 21^{st} century.

The awe inspiring science-fiction stories written by Dr. Asimov, Mr. Clarke, and Mr. Heinlein always inspired me to keep pushing the frontiers of science and mathematics to help bring some of their dreams to reality.

My appreciation for courses, professors, and knowledge gained during my time at University of Illinois at Urbana-Champaign and Columbia University continues to be immense.

Finally, I would like to thank Dr. Sébastien Bubeck for inspiring me to think critically about sequential decision making under uncertainty during my doctoral studies.

To my teachers.

Contents

Abstract

Scaling Up Decision-Making Under Uncertainty

by

Mohit RAJPAL

Doctor of Philosophy in Computer Science

National University of Singapore

Recent advances in deep learning and reinforcement learning have demonstrated impressively scalable performance in many tasks. This naturally raises the question whether approaches rooted in probabilistic methods can also be increased in scalability to be competitive with these approaches.

We consider this question in both the deep learning and reinforcement learning setting. We consider the problem scenario of decision-making under uncertainty, which is a general problem setting that arises both in deep learning and reinforcement learning.

In deep learning, an important problem setting is the early pruning of neural network elements. Here, early pruning means pruning ineffective network elements during the training process to speed up the training process. This is posed as a decision-making problem where decisions are made on which elements to keep or prune and when to prune these elements. In this setting, we make the following contributions:

- We formalize the early pruning problem into a constrained optimization problem.

- We prove several helper lemmas to show how this constrained optimization problem can be efficiently solved.
- We utilize Multi-Output Gaussian Process to infer the performance of neural network elements.
- Using this inference model, and the helper lemmas, we propose an algorithm to perform early pruning with theoretical guarantees on its performance.
- We perform extensive validation, showing how our approach outperforms approaches from the deep learning literature at better preserving network performance when a significant portion of network elements are pruned.
- In addition, our approach is robust to changes in hyperparameter.

In reinforcement learning, an important problem setting is finding the optimal policy for a given task. This is posed as a decision-making problem where decisions are made on which policy to attempt next given the history of previous policies attempted so far. In this setting, we make the following contributions:

- We propose a parameter efficient policy model which is well suited for usage on memory-constrained devices such as Internet of Things (IoT) devices.
- We propose a variant of Bayesian Optimization to optimize this policy model which scales to a higher number of dimensions.
- We show that our Bayesian Optimization comes with strong regret guarantees.

- We perform extensive validation showing our proposed approach outperforms competing reinforcement learning approaches in sparse or malformed reward scenarios.

We make progress on the above decision-making problems and show superior performance to competing approaches under specific scenarios. To finalize our thesis, we propose the study of Adversarially Designed Games, built upon our work in the reinforcement learning setting. Adversarially Designed Games consider the question whether there exist games that with large suboptimality gaps when attempting to solve them with reinforcement learning approaches. In this setting we make the following contributions:

- We design a family of games which are difficult for reinforcement learning methods to solve.
- We prove theoretical results showing the difficulty of solving these games with reinforcement learning, and the relative ease of solving them with Bayesian Optimization thus indicating a suboptimality gap.
- We validate this suboptimality gap showing poor performance using reinforcement learning methods, and good performance with Bayesian Optimization thus empirically confirming a suboptimality gap.

These works show the recent developments and value of scaling up decision-making under uncertainty. Our proposal of Adversarially Designed Games also opens up further avenues for research in scaling up decision-making under uncertainty.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Setting and Motivation

Decision-making under uncertainty using probabilistic models is known to suffer from scalability issues (K. A. Wang et al. 2019). These issues are exhibited in all subareas involving probabilistic models including Graphical Models (Chaohui Wang, Komodakis, and Paragios 2013), Gaussian Processes (Srinivas et al. 2010), Bayesian Inference (Box and Tiao 2011), Monte Carlo methods (Luengo et al. 2020), et cetera. Comparatively, deep learning based approaches suffer fewer scalability concerns (Krizhevsky, Sutskever, and Geoffrey E Hinton 2012). This discrepancy has allowed deep learning approaches to outcompete and outperform traditional approaches to Computer Vision (K. He et al. 2016a), Exact Inference (K. A. Wang et al. 2019), Statistical Machine Translation (Sutskever, Vinyals, and Q. V. Le 2014), and other subareas.

Another subarea that has recently demonstrated strong empirical performance is reinforcement learning. The success of reinforcement learning has been demonstrated in several game playing environments such as Atari® game playing suite (Mnih,

Kavukcuoglu, Silver, Graves, et al. 2013), Go (Weiqi) (Silver, Huang, et al. 2016), Blizzard Entertainment StarCraft II (Vinyals, Babuschkin, et al. 2019), and Valve Software Defense of the Ancients II (Berner et al. 2019). Despite this recent success, reinforcement learning approaches perform poorly in games with sparse reward, such as Montezuma's Revenge (Salimans and R. Chen 2018). In order to better adapt reinforcement learning to such games, much research work has been attempted in curiosity based "intrisic reward" (Linke et al. 2020) which encourages exploration. Nonetheless, significant performance disparity exists between games with dense reward, and sparse reward. This demonstrates the lack of efficacy of reinforcement learning based approaches under certain scenarios.

Alongside the above research advancements, recent advancements in Bayesian Optimization (BO) for policy search (Müller, Rohr, and Trimpe 2021) show it is a competitive approach to reinforcement learning for many tasks. A policy is a set of rules for making decisions in some interactive environment (e.g., a game). In such an environment, a single entity (an agent) or multiple entities (multi-agent) work towards accomplishing some goal. Policy search involves searching over the policy space in some intelligent manner to find a well performing policy. In contrast to reinforcement learning, BO is able to optimize a black-box function under mild smoothness assumptions (Srinivas et al. 2010). BO has found great success in the areas of hyperparameter optimization (Snoek, Larochelle, and Ryan P. Adams 2012), Neural Architecture Search (White, Neiswanger, and Savani 2021; Zoph and Q. V. Le 2017), and gradient-based optimization (J. Wu et al. 2017). The comparative strength of BO is its flexibility in being able to optimize arbitrary

functions, including those of finding the optimal policy in a policy search task. Furthermore, BO approaches often give theoretical optimality guarantees under mild assumptions.

Given the above setting, the central motivation of this thesis is to improve algorithms and techniques to allow for scaling up decision-making under uncertainty in order to offer competitive performance to deep learning and reinforcement learning approaches. We consider two scenarios to show that it is possible to scale up approaches using uncertainty modeling using algorithmic tools to offer competitive performance to deep learning and reinforcement learning, respectively. In the deep learning area, we consider the problem of *early pruning* a neural network to improve training performance. In the reinforcement learning area, we consider the problem of policy search both in the single agent and multi-agent setting.

Our first work in the deep learning area focuses on early pruning. *Deep neural networks* (DNNs) are costly to train. Pruning is an approach to alleviate model complexity by zeroing out or *pruning* DNN elements with little to no efficacy for a given learning task and has shown promise in reducing training costs for DNNs. Our work in the deep learning area presents a novel algorithm to perform *early* pruning of DNN elements (e.g., neurons or convolutional filters) *during the training process* while minimizing losses to model performance. To achieve this, we model the efficacy of DNN elements with a Bayesian paradigm conditioned on efficacy data collected during the training and prune DNN elements with low *predictive* efficacy after training completion. Empirical evaluations show that our Bayesian early pruning algorithm improves the computational efficiency of DNN training while

better preserving model performance compared to other tested pruning methods.

Our approach to Bayesian early pruning requires optimizing an NP-hard problem known as submodular set maximization (Krause and Golovin 2014). In this scenario, the set maximization problem is which DNN elements to keep or prune dependent on their collected efficacy behavior during training. To efficiently optimize this problem, thus allowing *the approach to scale* we model the set elements with a Multi-Output Gaussian Process (Álvarez and Lawrence 2011). This mild assumption allows us to optimize an NP-hard problem in linear time. We justify this assumption by showing that the Multi-Output Gaussian Process modeling approach well captures the behavior of DNN elements during the training process. We show empirically that our approach outperforms competing approaches from the deep learning literature at solving the early pruning problem. We also show that our approach is *robust* without extensive hyperparameter tuning.

Our second work in the reinforcement learning area focuses on policy search. Many approaches for optimizing decision-making models rely on gradient based methods (e.g., reinforcement learning) requiring informative feedback from the environment. Here, a decision-making model is some model which takes as input the environment state (e.g., in a game) and decides the resultant action. In environments where reward feedback is sparse or uninformative, such approaches may result in poor performance due to a lack of valuable behavior to *reinforce*. Gradient-free approaches not relying on informative feedback such as BO mitigate the dependency on the quality of gradient feedback, but are known to scale poorly in the high-dimension setting of complex decision-making models necessary in complex environments. This

problem is exacerbated if the model complexity is increased due to interactions between several agents cooperating to accomplish a shared goal. To address the dimensionality challenge, we propose a compact multi-layered architecture modeling the dynamics of agent interactions through the concept of role. We introduce Hessian-Aware Bayesian Optimization to efficiently optimize the multi-layered architecture parameterized by a relatively larger number of parameters. Our approach shows strong empirical results under malformed or sparse reward.

Our approach to policy search approaches the high-dimensional complex optimization problem along two directions. Along the first direction, we design a Higher-Order Model (HOM) which *generates* a model. This HOM approach has shown to exhibit parameter efficiency, allowing for complex models to be expressed with fewer parameters (Y. Lee et al. 1986). The second direction of improvement is by designing Hessian-Aware GP-UCB (HA-GP-UCB) which builds upon the work of the celebrated GP-UCB algorithm (Srinivas et al. 2010). HA-GP-UCB decomposes a complex high-dimensional policy search task into several easier to learn optimization tasks which can be learned independently. Our key contribution in the area of High-Dimensional Bayesian Optimization (HDBO) is *learning* how to decompose a complex high-dimensional task into many easy to learn tasks. We are able to do this by observing the Hessian, or the interaction effect, between different dimensions of the policy space. By observing how the different dimensions of the policy space interact, we are able to deduce how best to partition a large complex task into easier to optimize smaller tasks. Putting these contributions together allows us to outperform reinforcement learning and multi-agent reinforcement

learning approaches on smaller policies suitable for Internet of Things (IoT) and autonomous drone environments. Such use cases are becoming more important due to the presence of IoT in ubiquitous computing (Merenda, Porcaro, and Iero 2020) as well as autonomous drone delivery systems (Dorling et al. 2017). Our approach is also *robust* requiring no hyperparameter optimization on a per-environment basis.

To unify the thematic elements of our two works and propose a valuable direction for future work, we propose the study of Adversarially Designed Games for reinforcement learning. Adversarial reinforcement learning, the analogue of Adversarial Machine Learning in the reinforcement learning setting, has received some attention in recent years (Gronauer and Diepold 2022). However, compared to Adversarial Machine Learning in the deep learning setting, it is still a nascent research area. Thus, it remains an open and valuable question whether similar shortcomings exist in the reinforcement learning setting.

The above research question directly arises from the susceptibility of deep learning approaches to adversarial attacks. Adversarial Machine Learning often demonstrates the fragility of deep learning models. Adversarial Machine Learning (Machado, Silva, and Goldschmidt 2023) presents several attacks designed to cause deep learning models to make significant prediction errors and drastically reduce their empirical performance. Several attack vectors exist in Adversarial Machine Learning, including training time attacks (e.g., data poisoning), deployment time attacks (e.g., weight perturbations), and inference time attacks (e.g., adversarially crafted example). These attacks demonstrate, that although the empirical performance of deep learning approaches has shown tremendous improvement, this has come at the cost of the

fragility of deep learning models.

To further extend Adversarial Machine Learning into the reinforcement learning area, we study whether there exist environments (i.e., games) where reinforcement learning shows provably poor performance. We study this question in for policy gradient approaches (Schulman et al. 2017), which have recently become more popular and demonstrated superiority over Deep Q-Learning (Mnih, Kavukcuoglu, Silver, Graves, et al. 2013) in common reinforcement learning benchmarks. However, we propose that this has shortcomings where there exist adversarially designed environments where policy gradient methods exhibit poor performance. To attempt to demonstrate this in a theoretically principled way, we propose the study of Adversarially Designed Games, which policy gradient approaches provably have difficulty solving. To demonstrate this, we show theoretical results illustrating why and how policy gradient approaches have difficulty solving such games. In addition, we provably show that common BO approaches can solve such games with performance guarantees. This combination of results shows a likely *suboptimality gap* between the performance of reinforcement learning and BO for Adversarially Designed Games. Finally, we empirically validate the existence of this suboptimality gap by comparing policy gradient and BO methods at solving Adversarially Designed Games.

This thesis shows under which scenarios, modern deep learning and reinforcement learning methods are outperformed by approaches using uncertainty modeling. In all our demonstrated scenarios, uncertainty modeling allows for robust and scalable decision-making outperforming competing approaches from the deep learning

or reinforcement learning literature. We extend the results in these scenarios by proposing Adversarially Designed Games, showing theoretical shortcomings in modern reinforcement learning approaches. These theoretical results are then empirically validated showing a clear suboptimality gap. Thus, scaling up decision-making under uncertainty continues to be a valuable direction for further study.

1.2 Contributions

The contributions of this thesis are as follows. We highlight the theoretical and empirical contributions of our two works in scaling up decision-making under uncertainty. Finally, we unify our two works by proposing and studying Adversarially Designed Games showing that our research direction is a valuable direction for further study.

Pruning During Training by Network Efficacy Modeling. To improve upon pruning methods in the deep learning literature, we propose a scalable algorithm to perform early pruning. This pruning algorithm is able to prune network elements during training to speed up the training process while preserving network prediction efficacy at test-time. To develop this algorithm, we first mathematically formalize the early pruning problem as an optimization problem. Following this formalization, we show that solving this optimization problem is NP-hard. Although this problem is NP-hard, we show that this optimization problem is submodular allowing for efficient approximation with a small suboptimality gap. To improve upon this efficiency, we additionally assume that the *distribution* of network element efficacy follows a

multivariate Gaussian. Using this assumption, we are able to solve the early pruning optimization problem in linear time and prove bounds on the suboptimality gap.

To implement this algorithm, we model the efficacy of network elements using a multi-output Gaussian Process. We show this modeling approach is able to well capture the evolution of network element efficacy over time. Finally, we show extensive validation showing our approach is better able to preserve network prediction efficacy at test-time while still delivering significant training-time speedup. We additionally show our approach is robust to changes in hyperparameters, and does not require hyperparameter tuning on a per-network basis.

Hessian-Aware Bayesian Optimization for Decision-Making Models. To offer superior performance to reinforcement learning based approaches for optimizing decision-making policies on memory constrained devices, we propose a Higher-Order Model and a variant of High-Dimensional Bayesian Optimization. The proposed Higher-Order Model is a model which generates a model and is more memory efficient than traditional modeling approaches. We propose a variant of High-Dimensional Bayesian Optimization to efficiently optimize this model. Our proposed approach decomposes a large optimization problem into many smaller optimization problems which are easier to solve. We prove a regret bound under mild and general assumptions which improves upon the regret bounds proved by other related works.

We validate our approach against several reinforcement learning and Bayesian Optimization approaches used for policy optimization. In this validation, we show that our approach outperforms competing approaches under sparse and malformed

reward scenarios. Finally, our approach to policy optimization is robust given its usage of only two hyperparameters, which are held constant throughout all tested tasks.

Adversarially Designed Games. To extend our results in both the deep learning and reinforcement learning setting, we expand the field of Adversarial Machine Learning in the reinforcement learning setting by proposing Adversarially Designed Games. We formally propose and design Adversarially Designed Games. Following this design, we derive policy gradient rules in order to apply reinforcement learning to this family of games. We prove theoretical results on the difficulty of solving these games using policy gradient approaches, and the effectiveness of Bayesian Optimization for doing the same, thus showing a likely suboptimality gap.

We empirically validate the presence of this suboptimality gap by comparing policy gradient approaches and Bayesian Optimization.

1.3 Organization

In the remaining portions of this thesis, we present the necessary background in Chapter 2, important related works in Chapter 3, scaling up decision-making under uncertainty in a deep learning setting in Chapter 4, scaling up decision-making under uncertainty in a reinforcement learning setting in Chapter 5. Lastly, we consider Adversarially Designed Games in Chapter 6. We finally conclude in Chapter 7.

Chapter 2

Background

In this chapter, we introduce the necessary background and notation for understanding of this thesis. We introduce Gaussian Process, a Bayesian approach to modeling regression type problems which also quantifies uncertainty in its inference procedure. In addition, we introduce Bayesian Optimization, a Bayesian approach to zero'th order optimization that does not depend on gradient feedback from the function being optimized.

2.1 Gaussian Process

A Gaussian process (GP) (Rasmussen and C. K. I. Williams 2006) models a set of random variables as a multivariate Gaussian Distribution. This effectively allows a GP to model a distribution over functions $f : \mathcal{X} \to \mathbb{R}$ where $\mathcal{X} \subset \mathbb{R}^d$ for some dimensionality d. Generally, GP approaches focus on the noisy setting where every observation of the function f is corrupted by zero mean additive Gaussian noise : $y(\mathbf{x}) = f(\mathbf{x}) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ where σ^2 is the noise factor. A GP is represented as $\mathcal{GP}(\mu(\cdot), k(\cdot, \cdot))$ where $\mu : \mathcal{X} \to \mathbb{R}$ is the mean function, and $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ represents a similarity or covariance function between points $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$. The inductive

bias or assumed smoothness of the function, f, is encoded in the kernel function, k. It is generally assumed that w.l.o.g. that $\mu(\mathbf{x}) = 0$ and $k(\mathbf{x}, \mathbf{x}') \leq 1$.

As a consequence of this definition, given a set of T observations $[(\mathbf{x}_t, y_t)]_{t=1,\ldots,T}$, the posterior belief at any input $\mathbf{x}$ is a Gaussian distribution with the following posterior mean and variance.

$$\mu_T(\mathbf{x}) \triangleq \mathbf{k}_T(\mathbf{x})^\top (\mathbf{K}_T + \sigma^2 \mathbf{I})^{-1} \mathbf{y}_T, \;\; \sigma_T^2(\mathbf{x}) \triangleq k(\mathbf{x}, \mathbf{x}) - \mathbf{k}_T(\mathbf{x})^\top (\mathbf{K}_T + \sigma^2 \mathbf{I})^{-1} \mathbf{k}_T(\mathbf{x}) \tag{2.1}$$

where $\mathbf{K}_T \triangleq [k(\mathbf{x}_t, \mathbf{x}_{t'})]_{t,t'=1,\ldots,T}$, $\mathbf{k}_T(\mathbf{x}) \triangleq [k(\mathbf{x}_t, \mathbf{x})]_{t=1,\ldots,T}^\top$ and $\mathbf{y}_t \triangleq [y_t]_{t=1,\ldots,T}$. Here $\mathbf{K}_T$ is a $T \times T$ Gram matrix. $\mathbf{k}_T$ and $\mathbf{y}_T$ are both $T \times 1$ column vectors.

2.2 Bayesian Optimization

Bayesian Optimization (BO) involves sequentially maximizing an unknown objective function $v : \Theta \to \mathbb{R}$. In each iteration $t = 1, \ldots, T$, an input query θ_t is evaluated to yield a noisy observation $y_t \triangleq v(\theta_t) + \epsilon$ with i. i. d. Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$ as in Section 2.1. In this thesis, we assume that $\Theta \subset \mathbb{R}^d$ for some d, however BO also works well in discrete domains (Srinivas et al. 2010).

BO selects input queries to approach the global maximizer $\theta^* \triangleq \arg\max_{\theta \in \Theta} v(\theta)$ as rapidly as possible. The performance of BO is most commonly measured by *cumulative* regret $R_T \triangleq \sum_{t=1}^T r(\theta_t)$, where $r(\theta_t) \triangleq v(\theta^*) - v(\theta_t)$.

The belief of v is modeled by a *Gaussian process* (GP), denoted GP $(\mu(\theta), k(\theta, \theta'))$, that is, every finite subset of $\{v(\theta)\}_{\theta \in \Theta}$ follows a multivariate Gaussian distribu-

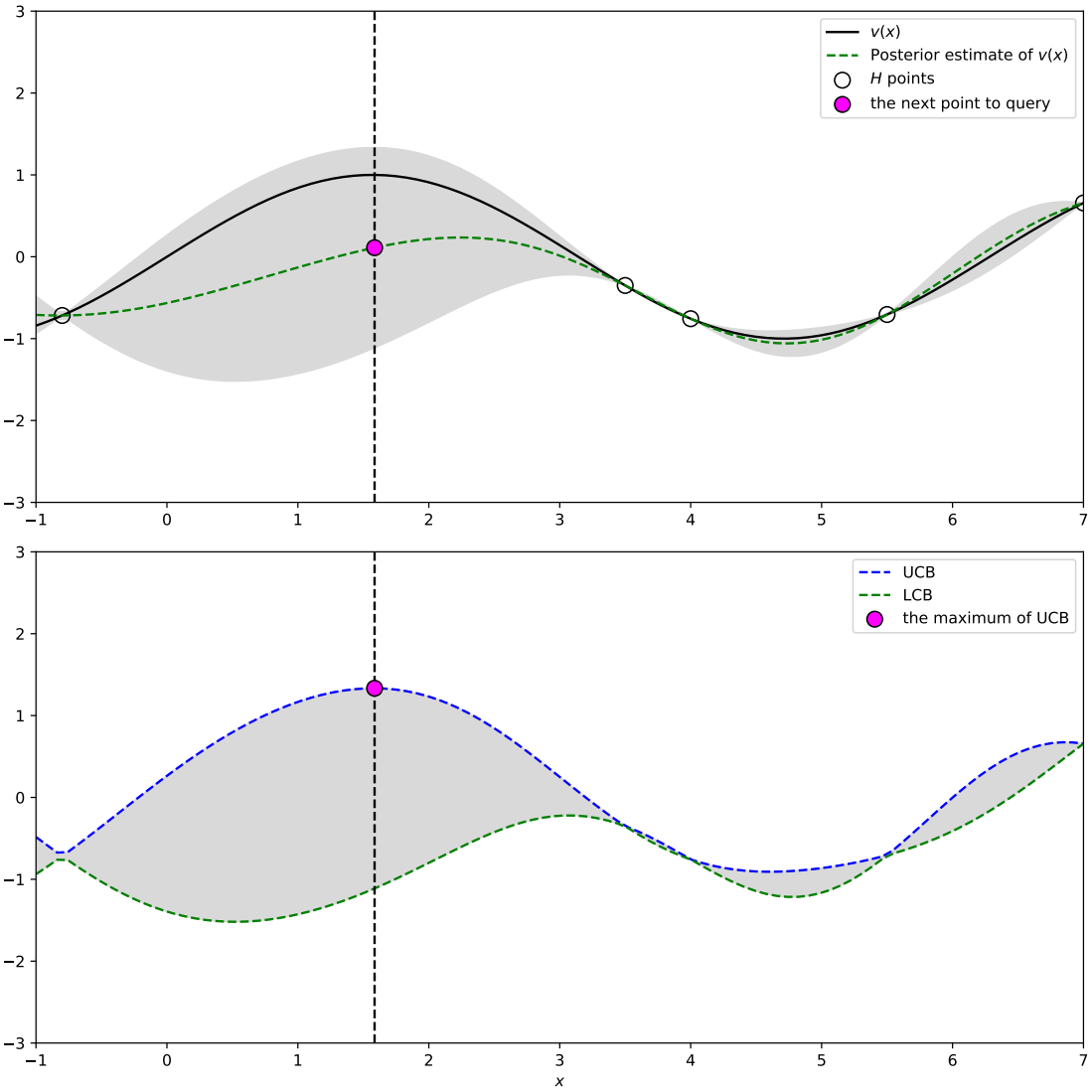

Figure 2.1: Top: Gaussian Process inference on $v(x) = \sin(x)$. Given the previously queried points, the unobserved function values are inferred, along with confidence estimates. Bottom: Given the Gaussian Process inference, a point is chosen which maximizes the upper confidence bound.

tion (Rasmussen and C. K. I. Williams 2006). A GP is fully specified by its *prior* mean $\mu(\theta)$ and covariance $k(\theta, \theta')$ for all $\theta, \theta' \in \Theta$, which are, respectively, assumed w.l.o.g. to be $\mu(\theta) = 0$ and $k(\theta, \theta') \leq 1$. Given a vector $\mathbf{y}_T \triangleq [y_t]^\top_{t=1,\dots,T}$ of noisy observations from evaluating v at input queries $\theta_1, \dots, \theta_T \in \Theta$ after T iterations, the GP posterior belief of v at some input $\theta \in \Theta$ is a Gaussian with the following

posterior mean $\mu_T^k(\theta)$ and variance $[\sigma_T^k]^2(\theta)$:

$$\mu_T^k(\theta) \triangleq \mathbf{k}_T^k(\theta)^\top (\mathbf{K}_T^k + \sigma^2 \mathbf{I})^{-1} \mathbf{y}_T, \quad \left[\sigma_T^k\right]^2(\theta) \triangleq k(\theta,\theta) - \mathbf{k}_T^k(\theta)^\top (\mathbf{K}_T^k + \sigma^2 \mathbf{I})^{-1} \mathbf{k}_T^k(\theta) \tag{2.2}$$

where $\mathbf{K}_T^k \triangleq [k(\theta_t, \theta_{t'})]_{t,t'=1,\ldots,T}$ and $\mathbf{k}_T^k(\theta) \triangleq [k(\theta_t, \theta)]_{t=1,\ldots,T}^\top$. We note that this inference rule is exactly the same as in Section 2.2 with the kernel k incorporated into the notational semantics.

In each iteration t of BO, an input query $\theta_t \in \Theta$ is selected to maximize an acquisition function such as, $\theta_t \triangleq \arg\max_{\theta \in \Theta} \mu_{t-1}^k(\theta) + \sqrt{\beta_t}\sigma_{t-1}^k(\theta)$ (Srinivas et al. 2010) where β_t follows a well-defined pattern. Effective approaches to performing BO rely on choosing θ_t in some intelligent manner to balance *exploration* of the unknown function v with *exploitation* of this learned information to find the maximizer of v. Several approaches exist to perform acquisition including expected-improvement (Jones, Schonlau, and Welch 1998), entropy search (Hennig and Schuler 2012), Thompson Sampling (Chowdhury and Gopalan 2017), among others. An illustrative example of both Gaussian Process inference and Bayesian Optimization is presented in Fig. 2.1.

2.3 Pruning

Pruning is a technique used to address the issue of overparameterization in deep neural networks (DNNs) by identifying and eliminating ineffective parameters while maintaining predictive performance (LeCun, Denker, and Solla 1989). This method

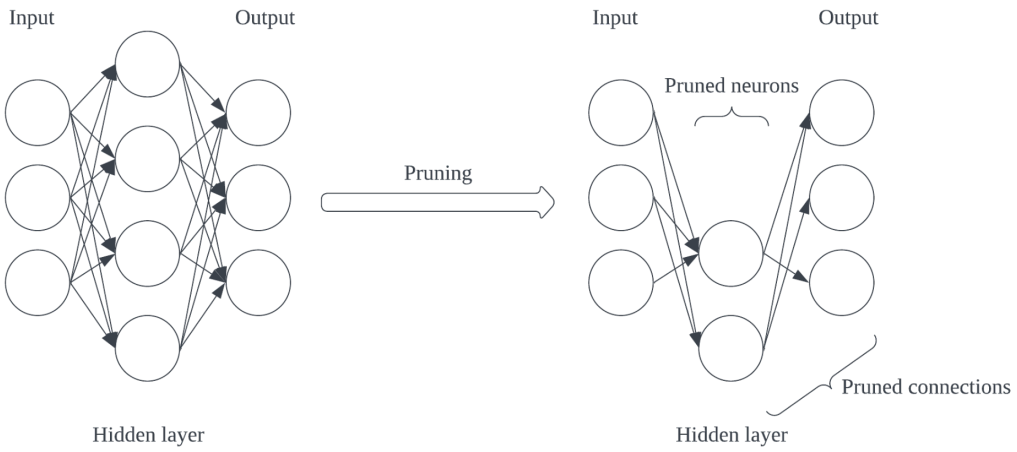

Figure 2.2: On the left, a neural network prior to pruning is presented. On the right, the network after pruning is presented where ineffectual network elements such as connections and neurons have been pruned.

is typically implemented after the DNN has been trained, with the primary objectives being to expedite test-time evaluation and to enable deployment on devices with limited resources, such as mobile phones and cameras. In standard image classification tasks using datasets like MNIST, CIFAR-10, and ImageNet, pruning has been shown to significantly reduce the number of learnable parameters without significantly affecting predictive performance (S. Han et al. 2015; H. Li et al. 2017; Molchanov et al. 2017; S. Lin et al. 2019). A visualizing figure of a neural network before and after pruning is presented in Fig. 2.2.

Pruning involves refining the neural network $\mathcal{N}\boldsymbol{v}$, parameterized by $\boldsymbol{v} = [v^a]a = 1..M$, where M represents pruneable elements (e.g., weights, neurons, filters). The loss function $\mathcal{L}(\mathcal{N}_{\boldsymbol{v}})$ assesses predictive efficacy. Pruning removes ineffective network elements within a sparsity budget B, preserving accuracy. The pruning optimization problem is formulated as:

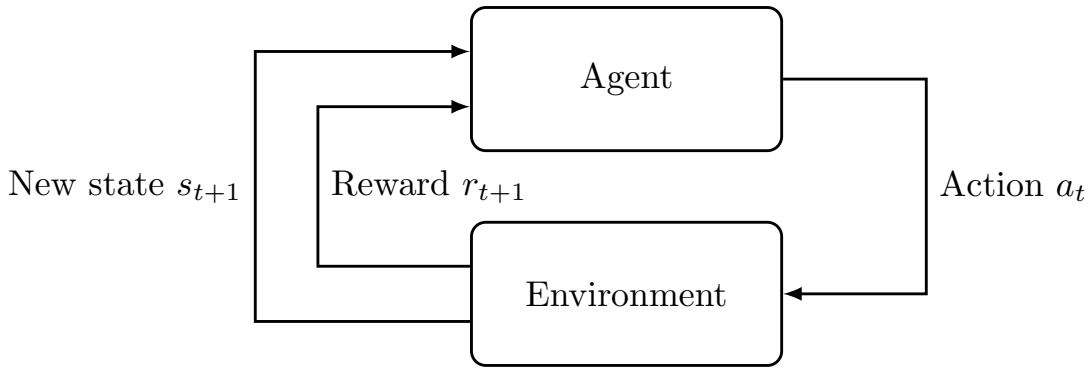

Figure 2.3: An agent interacting with an environment is presented. The agent interacts with the environment and receives reward as feedback. Reinforcement learning uses this feedback to improve the agent's decision making.

$$\min_{\boldsymbol{m}\in\{0,1\}^M} |\mathcal{L}(\mathcal{N}_{\boldsymbol{m}\odot\boldsymbol{v}}) - \mathcal{L}(\mathcal{N}_{\boldsymbol{v}})| \quad \text{s.t.} \quad \|\boldsymbol{m}\|_0 \leq B$$

where $\odot$ is the Hadamard product and $\boldsymbol{m} = [m^a]_{a=1..M}$ is a pruning mask. Here, $m^a \times v^a$ prunes v^a if $m^a = 0$, keeping it otherwise. Pruning involves zeroing network elements or their weight parameters, including those using the pruned element's output.

2.4 Reinforcement Learning

Reinforcement Learning (RL) is a framework for teaching agents to make sequential decisions by interacting with an environment (Kaelbling, Littman, and Moore 1996). In RL, an agent observes the current state of the environment, takes an action, and receives a reward based on that action and the resulting state transition. This relationship is presented in 2.3. The goal of the agent is to learn a policy—a mapping from states to actions—that maximizes the cumulative expected reward over time.

A popular approach to reinforcement learning is policy gradient. Policy gradi-

ent methods directly optimize the policy parameterization to maximize expected cumulative rewards. Let π_θ represent a policy parameterized by θ. The objective in policy gradient methods is to maximize the expected cumulative reward, denoted as $J(\theta)$, which is defined as:

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)}[R(\tau)]$$

where $\tau = (s_1, a_1, r_1, \ldots, s_T, a_T, r_T)$ is a trajectory, $p(\tau;\theta)$ is the trajectory distribution under policy π_θ, and $R(\tau)$ is the cumulative reward obtained along trajectory τ.

The policy gradient is computed as the gradient of the expected cumulative reward:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)} \left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) R(\tau) \right]$$

Policy gradient methods employ stochastic gradient ascent to update the policy parameters by sampling trajectories and estimating the gradient using these samples. By iteratively updating the policy parameters in the direction of the gradient, policy gradient methods aim to find policies that maximize expected cumulative rewards.

RL techniques, particularly policy gradient methods, have demonstrated remarkable success in various domains, including game playing, robotics, and natural language processing (Yuxi Li 2017).

Chapter 3

Related Works

In this chapter, we give an overview of relevant subareas and subfields. We discuss the subareas of deep learning, reinforcement learning, and Adversarial Machine Learning within these subareas. In addition, we introduce related works in the subfields of Deep Neural Network Pruning and Sequential Decision-Making in detail. We also focus on introducing related works in these subfields which are relevant for understanding our contributions.

3.1 Deep Learning

We give an introductory survey to modern deep learning approaches that are unified by a common thread of large, scalable, and data expensive models. In addition, we also give an introductory survey of Adversarial Machine Learning, showing some of the known drawbacks of these approaches.

Deep learning on the minute scale has not significantly changed since The Perceptron (Rosenblatt 1958). Modern deep learning architectures still consist of an affine function followed by a non-linearity. The non-linearity is often referred to as an activation function. Significant effort has been expended into discovering newer

and more effective forms of activation functions (Dubey, S. K. Singh, and Chaudhuri 2022). Often, the benefits of more effective forms of activation functions are better performance, training characteristics, robustness, and an improved loss landscape. Some popular activation functions include Softplus (Dugas et al. 2000), Rectified Linear Units (Nair and Geoffrey E. Hinton 2010), Exponential Linear Unit (Clevert, Unterthiner, and Hochreiter 2016), and Swish (Ramachandran, Zoph, and Q. V. Le 2018). The study of trainable activation functions with parameterization of the non-linearity is an ongoing field of study (Apicella et al. 2021).

On the larger scale, several architectures have emerged as popular and useful paradigms for accomplishing specific use cases. These can historically be categorized into vision and text architectures.

Convolutional Neural Networks (Z. Li et al. 2022) (CNNs) significantly improved the efficacy of neural networks at learning in the vision space. A convolution is an approach to simplify the hypothesis space of neural networks for vision tasks by parameterizing only a small receptive field (the filter) which is then repeatedly applied (i.e., convolved) over the image being processed. This inductive bias assumes that every small patch of the image can be similarly processed or interpreted by a neural network. Approaches using the convolution operation have demonstrated significant success. Significant architectural milestones in CNNs include AlexNet (Krizhevsky, Sutskever, and Geoffrey E Hinton 2012), VGG (Simonyan and Zisserman 2015a), ResNet (K. He et al. 2016a), and U-net (Ronneberger, P. Fischer, and Brox n.d.). Convolution continues to be an important and useful operation in computer vision tasks.

In the textual approaches, Recurrent Neural Networks (Rumelhart, Geoffrey E Hinton, R. J. Williams, et al. 1985) (RNNs) form the architectural basis of subsequent works. RNNs are autoregressive approaches which decompose a learning task into a task considering both the sequential history of previously seen data, and the currently seen data. A followup architecture introduced long short-term memory (Hochreiter and Schmidhuber 1997) (LSTM) to aid in remembering important parts of the previously seen data while performing inference. More recent deep learning models are built on architectures using LSTM as the basic building block. These include Sequence-to-Sequence models (Sutskever, Vinyals, and Q. V. Le 2014), Sequence-to-Sequence with Attention (Bahdanau, Cho, and Bengio 2015), Pointer Networks (Vinyals, M. Fortunato, and Jaitly 2015), among others. Much of this architectural work has been subsumed by Transformers (Vaswani et al. 2017) using attention as a key computation mechanism. Textual approaches such as the above have achieved considerable success in several language based tasks such as Statistical Machine Translation (Sutskever, Vinyals, and Q. V. Le 2014). More recently, Vision Transformers (K. Han et al. 2023) have shown superior performance to CNNs in the computer vision field.

Other important architectures of note are Graph Neural Networks (Z. Wu et al. 2021) (GNNs), Physics-Informed Neural Networks (Karniadakis et al. 2021) (PINNs) and Large Language Models (Chang et al. 2023) (LLMs). GNNs integrate approaches from Graphical Models where several "nodes" of neural networks utilize message passing as a communication mechanism to iteratively evolve or refine inference expressed in a graphical format. PINNs utilize stronger inductive biases

inspired from physics to aid in learning with less data than traditional deep learning approaches. LLMs utilize significant scalability in both data and computational resources to train very large language models capable of many language based tasks. These architectural approaches have found success in a diverse number of fields.

The above introductory survey does not cover several key areas associated with or related to deep learning such as optimization (Mittal and Vaishay 2019), differentiable loss functions (Q. Wang et al. 2020), meta-learning (Hospedales et al. 2022), Generative Adversarial Networks (Jabbar, X. Li, and Omar 2022), diffusion models (Ling Yang et al. 2023), hyperparameter optimization (Li Yang and Shami 2020), Neural Architecture Search (Elsken, Metzen, and Hutter 2019), Federated Learning (C. Zhang et al. 2021), distributed training (Verbraeken et al. 2021), and several others. Deep learning has been and continues to be an evolving field.

Despite its numerous strengths, the drawbacks of deep learning been highlighted by Adversarial Machine Learning (Machado, Silva, and Goldschmidt 2023). Although deep learning approaches have demonstrated excellent empirical performance, a growing concern with these approaches is their susceptibility to adversarial attacks. Adversarial Machine Learning often demonstrates the fragility of deep learning models. Several attack vectors exist in Adversarial Machine Learning, including training time attacks (e.g., data poisoning), deployment time attacks (e.g., weight perturbations), and inference time attacks (e.g., adversarially crafted example). These attacks demonstrate, that although the empirical performance of deep learning approaches has shown tremendous improvement, this has come at the cost of the fragility of deep learning models.

Several research works exist in Adversarial Machine Learning highlighting the weaknesses under adversarial conditions of CNNs (Akhtar et al. 2021), RNNs (Papernot et al. 2016), Transformers (C. Guo et al. 2021; L. Li et al. 2020; K. Mahmood, R. Mahmood, and Dijk 2021), GNNs (Zügner et al. 2020), and LLMs (Zou et al. 2023). PINNs (Yao Li et al. 2023; Shekarpaz, Azizmalayeri, and Rohban 2022) also show susceptibility to adversarial attack, despite being a nascent research direction. Building deep learning models which are robust to Adversarial Machine Learning is an ongoing field of study.

3.2 Reinforcement Learning

We give a brief survey on classical reinforcement learning (Kaelbling, Littman, and Moore 1996) and its evolution into deep reinforcement learning (Yuxi Li 2017). In this work, reinforcement learning used without additional qualifiers refers to deep reinforcement learning (RL). This is followed by a survey into recent developments in RL as well as its drawbacks, similar to our review of deep learning.

RL optimizes an agent either in the Markov Decision Process (MDP) or Partially Observable Markov Decision Process (POMDP) setting. In either setting, an *agent* is in some environment where it can observe the environment state, take some action, and as a consequence of the action it moves to another state and receives some reward (Krishnamurthy 2016). In the POMDP setting, the current state of the agent is not known and must be inferred given observations about the current state.

The goal of RL is to learn the optimal policy which decides what actions to take given the observation or state which maximizes the cumulative reward of

the agent. The two widespread schools of thought in historical treatments of RL are model-based (Moerland et al. 2023) and model-free (Çalisir and PehlIvanoõlu 2019) methods. In model-based methods, access to the reward function, as well as transition probabilities for the next state are thought to be known. In model-free methods, these values are unknown and require modeling to reason about. Generally, the model-free setting is considered more difficult to solve, but can be practically applied in more scenarios.

Within model-based methods, two well known approaches are policy iteration (Bertsekas 2011) and value iteration (Ernst et al. 2005). In policy iteration, a policy is randomly initialized and allowed to make decisions in the environment. The valuable policy behaviors are then reinforced, allowing the policy to become more optimal and collect more reward. In value iteration, instead of initializing and iterating upon a policy, the reward function corresponding to state-action pairs is learned. Decision-making in an environment requires querying this learned reward function to reason about the best action given the current state.

In model-free methods, a similar dichotomy exists. The analogous approach to policy iteration is referred to as policy gradient (Silver, Lever, et al. 2014). Here, the agent's policy is directly parameterized and allowed to interact with the environment. Computing a gradient of the reward with respect to the policy parameters determines which behaviors to reinforce. Q-learning (Watkins and Dayan 1992) is analogous to value iteration in the model-free setting. A Q-function is parameterized as a model which encodes the value of taking specific actions in states. Similarly, decision-making depends on querying this learned Q-function. A hybrid approach combining

the two is known as Actor-Critic (Konda and Tsitsiklis 1999). Here the actor encodes the policy, and the critic encodes the model of the environment, providing a guiding signal to reinforce rewarding behaviors in the actor.

Transitioning to deep reinforcement learning takes these classical approaches and uses deep learning models to represent the policy, the value function, or the critic (Yuxi Li 2017). Although such an approach has led to immense improvement in empirical performance, the weaknesses of deep learning highlighted earlier are also inherited. This is evidenced in the growing field of Adversarial Reinforcement Learning (Gleave et al. 2020). Models when represented by DNNs in reinforcement learning suffer from a lack of robustness, which can be exploited to make a learned policy perform suboptimal actions.

In addition to the above weakness, another known weakness of deep reinforcement learning is its failure in learning an effective policy in the scenario of sparse reward (Aubret, Matignon, and Hassas 2019). Sparse reward occurs in environments where the correlation between action and reward is difficult to ascertain. Due to this difficulty, learning an effective policy through reinforcement learning remains a difficult challenge and is an ongoing topic of study.

The above survey serves as an introduction to reinforcement learning, and deep reinforcement learning. Reinforcement learning continues to be an ongoing field of study.

3.3 Pruning of Deep Neural Networks

In this section, we cover the related work for Chapter 4. Chapter 4 concerns scaling up Bayesian approaches for early pruning of Deep Neural Networks (DNNs). Pruning is an approach to improve model performance by zeroing out, or making redundant network elements with little to no efficacy at a given inference task. Early pruning is an approach to improve model training time by performing pruning during the training process. This allows improvement in model training time by pruning ineffective elements early in the training process.

3.3.1 Pruning and Related Techniques

Initial works in DNN pruning center around saliency-based pruning after training including Skeletonization (Mozer and Smolensky 1988), Optimal Brain Damage, and other followup works (Hassibi and Stork 1992; LeCun, Denker, and Solla 1989) as well as sensitivity-based pruning (Karnin 1990). Saliency and sensitivity measure the efficacy of a network element at the given inference task. More recently, saliency functions been adapted to pruning neurons or convolutional filters. H. Li et al. 2017 defined a saliency function on convolutional filters by using the L_1 norm. Molchanov et al. 2017 proposed using a first-order Taylor series approximation on the objective function as a saliency measure. X. Dong, S. Chen, and Pan 2017 proposed layer-wise pruning of weight parameters using a Hessian-based saliency measure. Several variants of pruning after training exist. S. Han et al. 2015 proposed *iterative pruning* where pruning is performed in stages alternating with fine-tune training. Y. Guo, Yao, and Y. Chen 2016 suggested dynamic network surgery where

pruning is performed on the fly during evaluation time. H. Li et al. 2017 and Y. He et al. 2018 proposed reinforcement learning for pruning decisions. A comprehensive overview can be found in (Gale, Elsen, and Hooker 2019).

Knowledge distillation (Geoffrey E. Hinton, Vinyals, and Dean 2015; Lu, M. Guo, and Renals 2017; Tung and Mori 2019; Yim et al. 2017) aims to transfer the capabilities of a trained network into a smaller network. Weight sharing (Nowlan and Geoffrey E. Hinton 1992; Ullrich, Meeds, and Welling 2017) and low-rank matrix factorization (Denton et al. 2014; Jaderberg, Vedaldi, and Zisserman 2014) compress the number of parameters of neural networks. Network quantization (Courbariaux, Bengio, and David 2015; Hubara et al. 2017; Micikevicius et al. 2018) uses lower-fidelity representation of network elements (e.g., 16 bits) to speed up training and evaluation. Our work is orthogonal to network quantization as we reduce overall training floating point operations during the training process to demonstrate empirical results.

3.3.2 Initialization Time or Training-Time Pruning

Frankle and Carbin 2019 showed that a randomly initialized DNN contains a small subnetwork which, if trained by itself, yields equivalent performance to the original network. This foundational work led to the field of pruning at initialization and pruning during training. Both fields attempt to reduce the training time of neural networks by early pruning ineffective network elements. SNIP (N. Lee, Ajanthan, and Torr 2019) and GraSP (Chaoqi Wang, G. Zhang, and Grosse 2020) perform pruning of connection weights at initialization through a first-order and second-order saliency

function, respectively. This technique was improved upon with IterSnip (Jorge et al. 2021) and SynFlow (Tanaka et al. 2020). PruneFromScratch (Yulong Wang et al. 2020) performs pruning at initialization to reduce training cost.

A somewhat related field is dynamic sparse reparameterization which performs pruning and regrowing parameter weights during the training process (Bellec et al. 2018; Dettmers and Zettlemoyer 2019; J. Liu et al. 2020; Mostafa and Xin Wang 2019). Sparse evolutionary training (Mocanu et al. 2018) initializes networks with sparse topology prior to training. Dai, Yin, and Jha 2019 proposed a grow-and-prune approach to learn network architecture and connection layout. Other works (Louizos, Welling, and Kingma 2018; Narang et al. 2017) have proposed pruning using heuristics such as L_0 norm regularization for DNNs and recurrent neural networks.

Other works have also proposed model compression during training as a constrained optimization problem as part of a general framework encompassing pruning, quantization, and low-rank decomposition (Idelbayev and Carreira-Perpiñán 2021a; Idelbayev and Carreira-Perpiñán 2021b; Lym et al. 2019). Within the context of our work in early pruning, we focus on minimizing test-time accuracy loss while reducing training cost through pruning during training. We also offer a mechanism to trade off between the metrics of test-time accuracy loss and training cost.

3.4 Policy Optimization

In this section, we cover the related work for Chapter 5. Chapter 5 concerns scaling up High-Dimensional Bayesian Optimization for policy optimization problems. Bayesian

Optimization is an approach to optimize black-box objective functions; however, it suffers from the curse of dimensionality. Works in High-Dimensional Bayesian Optimization focus on addressing this curse of dimensionality.

3.4.1 Approaches to Policy Optimization

Policy optimization focuses on optimizing a policy, which determines actions taken by an agent or agents to achieve a goal (Rizk, Awad, and Tunstel 2018; Roijers et al. 2013). A policy serves as a set of decision-making rules for an agent in some environment. In such an environment, there are predefined states that give the agent some reward. The study of policy optimization spans broad fields of study such as Game Theory (Condon 1992; Hu and Wellman 2003) and Swarm Intelligence (Barca and Sekercioglu 2013; Karaboga and Akay 2009). Subareas of note within policy optimization are the Markov Decision Process (MDP) and Partially Observable Markov Decision Process (POMDP) settings (Kaelbling, Littman, and Cassandra 1998). In both settings, a policy is optimized to accumulate maximum reward while interacting with an environment (Shani, Pineau, and Kaplow 2013). In both MDP and POMDP, approaches can be broadly categorized into direct policy search and reinforcement learning methods. Direct policy search (Heidrich-Meisner and Igel 2008; Lizotte et al. 2007; Martinez-Cantin 2017; Papavasileiou, Cornelis, and Jansen 2021; Wierstra et al. 2008) searches the policy space in some efficient manner. Reinforcement learning (Arulkumaran et al. 2017; Fujimoto, Hoof, and Meger 2018; Haarnoja et al. 2018; Lillicrap et al. 2015; Lowe et al. 2017; Mnih, Kavukcuoglu, Silver, Rusu, et al. 2015; Schulman et al. 2017) starts with a randomly initialized

policy and *reinforces* rewarding behavior patterns to improve the policy.

3.4.2 Bayesian Optimization

Bayesian Optimization (BO) has been utilized for direct policy search in the low-dimensional setting (Lizotte et al. 2007; A. Wilson, Fern, and Tadepalli 2014; Marco et al. 2016; Martinez-Cantin 2017; Rohr et al. 2018). However, these approaches have not scaled to the high-dimensional setting. Applying BO to policy optimization is a nascent field. In more recent works, BO has been utilized to aid in local search methods similar to reinforcement learning (Akrour et al. 2017; Eriksson et al. 2019a; L. Wang, Fonseca, and Tian 2020; Fröhlich, Zeilinger, and Klenske 2021; Müller, Rohr, and Trimpe 2021). BO for local search optimizes a black-box function by repeatedly searching for a nearby policy that improves upon the current policy. Here, black-box means the gradient of the reward with respect to the policy is not available. The search for nearby policies is done through black-box queries of nearby or similar policies. Recently, combinations of local and global search methods have been proposed (McLeod, Roberts, and Osborne 2018; Shekhar and Javidi 2021a). Typically, these approaches do not scale well to higher dimensions and lack thorough validation on common reinforcement learning benchmarks.

Somewhat related to our work is the usage of higher-order feedback (e.g., gradients) from the function being optimized by BO. Several recent works have considered BO under this additional constraint (Shekhar and Javidi 2021b; Ament and Gomes 2022a; Prakash et al. 2024). The key difference when compared to our work in BO is we utilize higher-order feedback to learn the dependency structure, as opposed

to using it directly for optimization. Furthermore, in practice, our approach only requires the surrogate Hessian, which is readily available in the policy optimization setting.

3.4.3 Multi-Agent Policy Optimization

Multi-Agent Policy Optimization is an intersectional field combining the field of Multi-Agent Systems with Policy Optimization (Dorri, Kanhere, and Jurdak 2018). Recently, this field has received significant interest from the machine learning community. We give a brief and introductory survey of these works.

Multi-Agent Reinforcement Learning (MARL) is delineated into centralized training and decentralized execution (CTDE) and centralized training and centralized execution (Comm-MARL) (C. Zhu, Dastani, and S. Wang 2022). The difference between these two approaches is whether agents are allowed to collaborate (i.e., in Comm-MARL) during execution. In both CTDE and Comm-MARL, agents are trained to learn a joint policy which considers the policy of other agents. However, during execution, CTDE does not allow communication or information sharing between agents for decision-making. The multi-agent interactions of CTDE methods can be implicitly captured by learning approximate models of other agents (Lowe et al. 2017; Foerster et al. 2018) or decomposing global rewards (Sunehag et al. 2017; Rashid et al. 2018; Son et al. 2019). These approaches allow for better informed decision-making during execution by modeling or inferring the behavior of other agents without requiring communication. Several approaches exist in Comm-MARL focusing on communication and interaction models between agents to allow for

collaborative decision-making during execution. However, these methods do not focus on how interactions are performed between agents (Shao et al. 2022; Y. Liu et al. 2020; P. Peng et al. 2017; Das et al. 2019; A. Singh, Jain, and Sukhbaatar 2019). In MARL, the concept of *role* is often leveraged to enhance the flexibility of behavioral representation while controlling the complexity of the design of agents (Hoang M Le et al. 2017; Lhaksmana, Murakami, and Ishida 2018; T. Wang, H. Dong, et al. 2020; T. Wang, Gupta, et al. 2021; C. Li et al. 2021). In role-based MARL, agents take decisions based on their currently assigned role. This approach allows for diversity in agent behavior on a per-role basis, while limiting the complexity of the model by having fewer roles than agents.

Scalable MARL is a nascent field tackling the challenges of exponential increase in the state and action space due to the multi-agent setting. A set of related approaches is to assume independence between the actions of each agent and their rewards (Claus and Boutilier 1998; Matignon, Laurent, and Fort-Piat 2012). Another related technique is to utilize mean field approximation techniques, assuming each agent interacts with the aggregated "mean" behavior of other agents (Zaman et al. 2020; Y. Yang et al. 2018). Another related approach is utilizing a predefined graph structures to allow interaction between nearby agents (Qu, Y. Lin, et al. 2020; Qu, Wierman, and N. Li 2020; Y. Lin et al. 2017), however, this approach does not extend to time-varying, distant interactions between unrelated agents.

Chapter 4

Pruning During Training by Network Efficacy Modeling

4.1 Introduction

Deep neural networks (DNNs) are known to be overparameterized (Allen-Zhu, Yuanzhi Li, and Liang 2019) as they usually have more learnable parameters than needed for a given learning task. Consequently, a trained DNN contains many ineffectual parameters that can be safely pruned or zeroed out with little effect on its performance.

Pruning (LeCun, Denker, and Solla 1989) is an approach to alleviate overparameterization of a DNN by identifying and removing its ineffectual parameters while preserving its predictive performance. Generally, pruning is applied to the DNN *after training* to speed up *test-time evaluation*, or for deployment of the model on resource-constrained devices (e.g., mobile phones, cameras, etc.). For standard image classification tasks with MNIST, CIFAR-10, and ImageNet datasets, it can reduce the number of learnable parameters by up to 50% or more with little to no reduction in predictive performance (S. Han et al. 2015; H. Li et al. 2017; Molchanov

et al. 2017; S. Lin et al. 2019).

However, applying pruning after training leads to considerable training cost being wasted on training ineffectual DNN elements (e.g., connection weights, neurons, or convolutional filters) which are eventually pruned. This problem is further compounded by the development of larger network architectures, which are expensive to train. These observations motivate the need of *early pruning*.

The objective of early pruning is to perform pruning *during training* for reducing training cost while maintaining predictive performance given a fixed final network sparsity (e.g., determined by the resource constraints of the deployment devices). This necessitates consideration for both the test-time efficacy and training cost as both metrics are highly desirable to users.

Related works in pruning during training (Lym et al. 2019) as well as prune and regrow (Bellec et al. 2018; Dettmers and Zettlemoyer 2019; Mostafa and Xin Wang 2019) approaches do not jointly consider both competing metrics while offering a fixed final network sparsity. Similarly, pruning at initialization (Jorge et al. 2021; N. Lee, Ajanthan, and Torr 2019; Tanaka et al. 2020; Chaoqi Wang, G. Zhang, and Grosse 2020) offers a method of reducing training cost, but does not consider maintaining predictive performance. These approaches overly sacrifice test-time efficacy by pruning *at initialization* when *test-time* network element efficacy is not well known. Therefore, previous works do not offer a mechanism to *trade off* between training cost and test-time efficacy according to user-defined sparsity objectives.

To offer a mechanism to trade off between training cost and test-time efficacy, a number of technical challenges must be addressed by early pruning. Early pruning

needs to minimize test-time efficacy, but pruning occurs during training when the test-time element efficacy is *unknown*. Therefore, pruning decisions need to be based on the *inferred* test-time network element efficacy. *How to infer test-time network element efficacy for making accurate early pruning decisions* is an important question that has not been addressed by related work. Secondly, building an inference model requires collecting network efficacy observations during training which has its own challenges. A more accurate model requires more observations to be collected, which increases the training cost. Conversely, pruning with an inaccurate model incurs lower DNN training cost but sacrifices predictive performance after training due to the inaccurate model inference. Given this trade-off, *when should an element that is predicted to perform poorly be pruned?* Finally, as both training cost and test-time efficacy are important metrics to end users, *how should these metrics be balanced while addressing the above challenges?*

To answer these questions, our work considers modeling the network element efficacy during training and performing early pruning based on the inferred element efficacy. In addition, our pruning decisions also depend on the *confidence* or certainty of the inferences. A network element is pruned when a sufficiently high degree of confidence is reached regarding its poor performance. To naturally trade off between training cost and predictive performance, we formulate early pruning as a constrained optimization problem and propose an efficient algorithm for solving it. The trade-off is achieved by modulating the degree of confidence necessary before a poorly performing element is pruned. Pruning with a high degree of confidence makes fewer mistakes, yet requires collecting more observations. Conversely, pruning with

a low degree of confidence makes more mistakes, yet requires fewer observations. The training cost is correlated with the number of observations collected, therefore reducing the confidence reduces the training cost. Within this approach, the specific contributions of our work in this paper include:

- We pose early pruning as a constrained optimization problem to minimize training cost while maintaining predictive efficacy by pruning during training under a fixed final network sparsity (Section 4.3.1).

- We propose inferring the efficacy of DNN elements using a *Multi-Output Gaussian Process* (MOGP) model which represents the *belief* of element efficacy conditioned on efficacy measurements collected during training. This approach not only identifies poorly performing elements but also provides a measure of confidence by assigning a probabilistic belief to its prediction (Section 4.3.2).

- We design a *Bayesian early pruning* (BEP) algorithm to allow trading off between training cost and predictive efficacy (Section 4.3.3). To the best of our knowledge, this is the first algorithm that can naturally satisfy a fixed final network sparsity while achieving the trade-off between training cost vs. predictive efficacy. Existing works either require fixed scheduled pruning strategy or cannot achieve the fixed final network sparsity without tuning parameters.

- We demonstrate strong performance improvements of our BEP algorithm when

a large percentage of the network is pruned. This improvement is significant as DNNs continue to grow in size and may require significant pruning to allow training in a short time frame (Section 4.4).

Pruning typically relies on a measure of network element efficacy, which is termed saliency (LeCun, Denker, and Solla 1989). The development of saliency functions is an active area of research with no clear optimal choice. To accommodate this, our algorithm is agnostic (and therefore flexible) to changes in saliency function. We use BEP to prune neurons and convolutional filters and demonstrate its ability to capture the trade-off between *training cost vs. predictive efficacy.*[1]

4.2 Preliminaries of Pruning

We define a dataset of D training examples with inputs $\mathcal{X} \triangleq \{\mathbf{x}_1, \ldots, \mathbf{x}_D\}$ and corresponding outputs $\mathcal{Y} \triangleq \{y_1, \ldots, y_D\}$. The neural network being trained is denoted with $\mathcal{N}_{\boldsymbol{v}_t}$ which is parameterized at time $t \leq T$ by a vector $\boldsymbol{v}_t \triangleq [v_t^a]_{a=1..M}$ of M *pruneable* network elements (e.g., weight parameters, neurons, or convolutional filters). The neural network undergoes a training process consisting of T iterations of *stochastic gradient descent* (SGD). Let $\mathcal{L}(\mathcal{X}, \mathcal{Y}; \mathcal{N}_{\boldsymbol{v}_t})$ be the loss function for the neural network $\mathcal{N}_{\boldsymbol{v}_t}$. The loss function behaves as a surrogate measure for its predictive efficacy as network elements which reduce loss improve the predictive efficacy. Pruning refines the network elements $\boldsymbol{v}_t$ given some user-specified sparsity budget B while preserving the accuracy of the neural network after convergence (i.e.,

[1] We have not considered pruning network connections since it cannot be easily capitalized upon with performance improvements due to the difficulty of accelerating sparse matrix operations with existing deep learning libraries (Buluç and Gilbert 2008; C. Yang, Buluç, and Owens 2018).

$\mathcal{N}_{\boldsymbol{v}_T}$). As shown by Molchanov et al. 2017, pruning can be stated as a constrained optimization problem:

$$\min_{\boldsymbol{m}\in\{0,1\}^M} |\mathcal{L}(\mathcal{X},\mathcal{Y};\mathcal{N}_{\boldsymbol{m}\odot\boldsymbol{v}_T}) - \mathcal{L}(\mathcal{X},\mathcal{Y};\mathcal{N}_{\boldsymbol{v}_T})| \quad \text{s.t.} \quad \|\boldsymbol{m}\|_0 \leq B$$

where $\odot$ is the Hadamard product and $\boldsymbol{m} \triangleq [m^a]_{a=1..M}$ is a pruning mask. Note that we abuse the Hadamard product to ease notations. For $a = 1, \ldots, M$, $m^a \times v_T^a$ corresponds to pruning v_T^a if $m^a = 0$, and keeping v_T^a otherwise. Pruning a network element means zeroing the network element or the network element weight parameters. Any weight parameters that use the output of the pruned network element are also zeroed.

In the deep learning community, a common approach is to solve the above combinatorial using a saliency function. A saliency function serves as a surrogate measure for the efficacy of network elements at minimizing the loss function. A network element with small saliency can be pruned since it is not *salient/important* in minimizing the loss function. Consequently, pruning is performed by maximizing the saliency of the network elements given the user-specified sparsity budget B:

$$\max_{\boldsymbol{m}\in\{0,1\}^M} \sum_{a=1}^{M} m^a s(a;\mathcal{X},\mathcal{Y},\mathcal{N}_{\boldsymbol{v}_T},\mathcal{L}) \quad \text{s.t.} \quad \|\boldsymbol{m}\|_0 \leq B \tag{4.1}$$

where $s(a;\mathcal{X},\mathcal{Y},\mathcal{N}_{\boldsymbol{v}_T},\mathcal{L})$ measures the saliency of v_T^a at minimizing $\mathcal{L}$ after convergence through T SGD iterations. The above optimization problem (Eq. 4.1) can be efficiently solved by greedily selecting the B most *salient* network elements in $\boldsymbol{v}_T$.

The construction of the saliency function has been discussed in many existing works. Some works have derived the saliency function from first-order (LeCun, Denker, and Solla 1989; Molchanov et al. 2017) and second-order (Hassibi and Stork

Table 4.1: Summary of key notations.

Notation	Description
M	Total number of network elements in a neural network
T	Total number of SGD iterations in the training procedure
s_t^a	Random variable denoting saliency of network element v_t^a at iteration t
$\boldsymbol{s}_t$	Random vector $[s_t^a]_{a=1}^M$ denoting saliency of network elements $\boldsymbol{v}_t$ at iteration t
$\boldsymbol{s}_{\tau_1:\tau_2}$	Random matrix $[\boldsymbol{s}_t]_{t=\tau_1}^{\tau_2}$ denoting saliency of network elements from iterations τ_1 to τ_2
$\tilde{\mathbf{s}}_{1:t}$	Realization of $\boldsymbol{s}_{1:t}$ denoting observed saliency measurements of network elements from iterations 1 to t
$\boldsymbol{m}_t$	Vector of pruning mask at iteration t
B_s	User-specified sparsity budget of the trained network
$B_{t,c}$	User-specified training cost budget at iteration t
$\mu_{t'\|1:t}^a$	Predictive mean of the saliency $s_{t'}^a$ given observed saliency measurements $\tilde{\mathbf{s}}_{1:t}$ from iterations 1 to t
$\sigma_{t'\|1:t}^{aa'}$	Predictive covariance of saliencies $s_{t'}^a$ and $s_{t'}^{a'}$ given observed saliency measurements $\tilde{\mathbf{s}}_{1:t}$ from iterations 1 to t

1992; Chaoqi Wang, G. Zhang, and Grosse 2020) Taylor series approximations of $\mathcal{L}$. Other common saliency functions include L_1 (H. Li et al. 2017) or L_2 (Wen et al. 2016) norm of the network element weights, as well as mean activation (Polyak and Wolf 2015). Our work here uses a first-order Taylor series approximation-based saliency function defined for neurons and convolutional filters (Molchanov et al. 2017).[2] Due to the first-order (i.e., gradient-based) approximation, this saliency function has minimal memory and training cost overhead during DNN training. However, our approach remains flexible to an arbitrary choice of saliency function. For ease of reference, Table 5.1 summarizes the notations that will be used frequently in the remaining sections of this paper.

4.3 Bayesian Early Pruning

4.3.1 Problem Formulation

Existing pruning works based on saliency function typically perform pruning after training completion (i.e., Eq. 4.1) to speed up the test-time evaluation on resource-

[2]Appendix A.1 gives the implementation details of this saliency function.

constrained devices. However, this approach wastes considerable time on training network elements which are eventually pruned. To resolve this issue, we extend the definition of the pruning problem (Eq. 4.1) along the temporal dimension to allow network elements to be pruned during the training process consisting of T SGD iterations.

Let the random variable $s_t^a \triangleq s(a; \mathcal{X}, \mathcal{Y}, \mathcal{N}_{\boldsymbol{v}_t}, \mathcal{L})$ represent the saliency of network element v_t^a after t SGD iterations. For $t = 1, \ldots, T$, the random vector $\boldsymbol{s}_t \triangleq [s_t^a]_{a=1..M}$ represents the saliency of network elements $\boldsymbol{v}_t$ at SGD iteration t, and the random matrix $\boldsymbol{s}_{\tau_1:\tau_2} \triangleq [\boldsymbol{s}_t]_{t=\tau_1..\tau_2}$ represents the saliency of all network elements from SGD iterations τ_1 to τ_2. Our early pruning problem is formulated with the goal of maximizing the saliency of the *unpruned* network elements after iteration T of SGD, yet allowing for pruning at each iteration t given some training cost budget $B_{t,c}$ for $t = 1, \ldots, T$:

$$\rho_T(\boldsymbol{m}_{T-1}, B_{T,c}, B_s) \triangleq \max_{\boldsymbol{m}_T} \ \boldsymbol{m}_T \cdot \boldsymbol{s}_T \quad (4.2a)$$

$$\text{s.t. } \|\boldsymbol{m}_T\|_0 \leq B_s \ (4.2b), \boldsymbol{m}_T \dot{\leq} \boldsymbol{m}_{T-1} \ (4.2c) \ , B_{T,c} \geq 0 \qquad (4.2d)$$

$$\rho_t(\boldsymbol{m}_{t-1}, B_{t,c}, B_s) \triangleq \max_{\boldsymbol{m}_t} \ \mathbb{E}_{p(\boldsymbol{s}_{t+1}|\tilde{\mathbf{s}}_{1:t})} \left[\rho_{t+1}(\boldsymbol{m}_t, B_{t,c} - \|\boldsymbol{m}_t\|_0, B_s)\right] \qquad (4.3a)$$

$$\text{s.t.} \quad \boldsymbol{m}_t \dot{\leq} \boldsymbol{m}_{t-1} \qquad (4.3b)$$

where B_s is some user-specified *sparsity* budget of the trained network, $\tilde{\mathbf{s}}_{1:t}$ is a realization of $\boldsymbol{s}_{1:t}$ representing the observed saliency measurements of all network elements from iterations 1 to t. Furthermore, $\boldsymbol{m}_0$ is a M-dimensional vector of 1's representing all network elements being *unpruned* at initialization, and $\boldsymbol{m}_t \dot{\leq} \boldsymbol{m}_{t-1}$

represents an element-wise inequality between $\boldsymbol{m}_t$ and $\boldsymbol{m}_{t-1}$: $m_t^a \leq m_{t-1}^a$ for $a = 1, \ldots, M$. At each iteration t, the saliency $\boldsymbol{s}_t$ is observed and $\boldsymbol{m}_t \in \{0,1\}^M$ in Eq. 4.3a represents a pruning decision performed to maximize the *expected value* of ρ_{t+1} with respect to the predictive belief of $\boldsymbol{s}_{t+1}$ conditioned on observed saliency measurements $\tilde{\mathbf{s}}_{1:t}$ collected up to and including SGD iteration t. This recursive structure terminates with the base case ρ_T where the saliency of the *unpruned* network elements is maximized after T SGD iterations.

In the above formulation, constraints (4.2c) and 4.3b ensure that pruning is performed in a practical manner whereby once a network element is pruned, it can no longer be recovered in a later SGD iteration. The user specifies a sparsity budget B_s (4.2b) which indicates the desired network size after training completion. This constraint is important as training is often performed on GPUs for resource-constrained devices (e.g., IoT devices or mobile phones) which can only support networks of limited size. The user also specifies the total training cost budget $B_{t,c}$ for training from SGD iterations t to T, which is reduced by the number $\|\boldsymbol{m}_t\|_0$ of unpruned network elements per SGD iteration. The constraint $B_{T,c} \geq 0$ (4.2d) ensures training completion within the specified training cost budget.

Here, we assume that a more sparse pruning mask $\boldsymbol{m}_t$ corresponds to a lower training cost effort at SGD iteration t due to fewer network elements being updated. Finally, Eq. 4.2a maximizes the saliency with a pruning mask $\boldsymbol{m}_T$ constrained by the *sparsity* budget B_s (4.2b). Our formulation of the early pruning problem balances the saliency of network elements after convergence (i.e., $\boldsymbol{m}_T \cdot \boldsymbol{s}_T$) against the total training cost for training the network (i.e., $\sum_{t=1}^{T} \|\boldsymbol{m}_t\|_0$). This appropriately captures

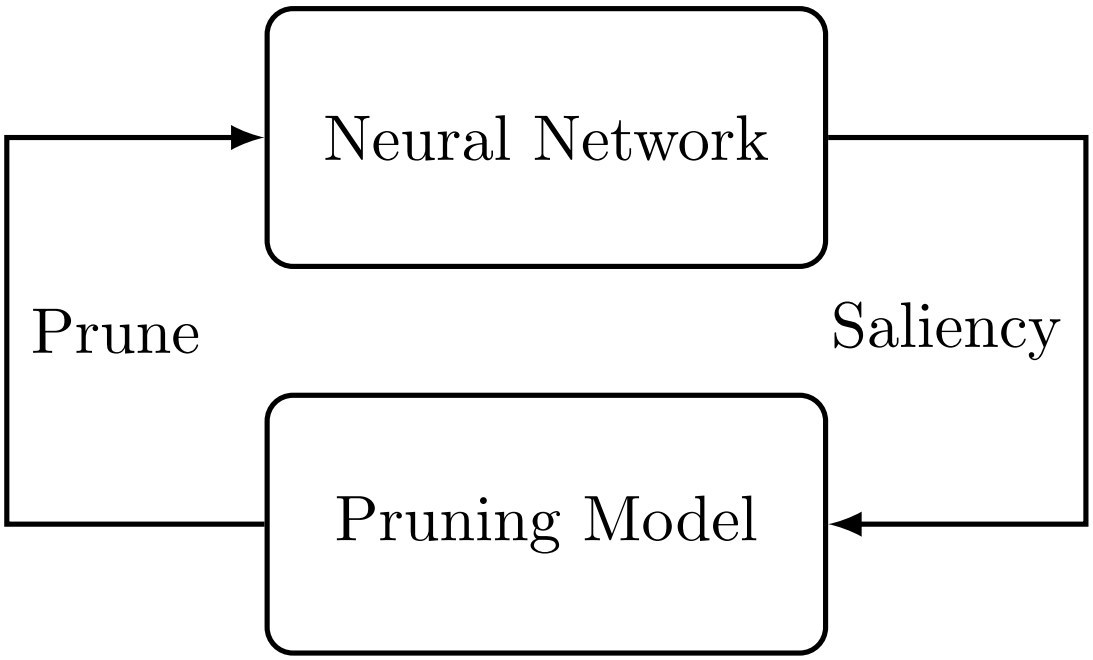

Figure 4.1: A visualization of the train-prune-train loop. A neural network is trained while collecting network element saliency data. A pruning model is constructed based on the collected saliency data and is used to prune the neural network. This process then repeats until the desired neural network size is reached.

the balancing act of training-time early pruning whereby training cost is reduced by *early pruning* network elements while preserving the saliency of the remaining network elements after training completion. We visualize this pruning and training loop in Fig. 4.1.

4.3.2 Modeling Saliency with Multi-Output Gaussian Process

To solve the above early pruning problem, we model the belief $p(\boldsymbol{s}_{1:T})$ such that the predictive belief $p(\boldsymbol{s}_{t+1:T} \mid \tilde{\mathbf{s}}_{1:t})$ of the future saliency $\boldsymbol{s}_{t+1:T}$ in Eq. 4.3a can be inferred. A naive approach is to decompose the belief $p(\boldsymbol{s}_{1:T}) \triangleq \prod_{a=1}^{M} p(\boldsymbol{s}^a_{1:T})$ and model the belief of the saliency $\boldsymbol{s}^a_{1:T} \triangleq [s^a_t]_{t=1}^T$ of each network element independently. Independent modeling, however, ignores the *co-adaptation* and *co-evolution* of

the network elements, which have been shown to be a common occurrence in DNNs (Geoffrey E. Hinton, Srivastava, et al. 2012; Srivastava et al. 2014; Chaoqi Wang, G. Zhang, and Grosse 2020). In addition, *explicitly* modeling the correlation of the saliency between different network elements is non-trivial since considerable feature engineering is needed to represent diverse network elements such as neurons, connections, or convolutional filters.

To resolve such issues, we use a *Multi-Output Gaussian Process* (MOGP) model to represent the belief $p(\boldsymbol{s}_{1:T})$ of the saliency of all network elements from SGD iterations 1 to T. Specifically, we assume that the saliency s_t^a of network element v_t^a at iteration t is a linear mixture of Q independent latent functions $[u_q(t)]_{q=1..Q}$: $s_t^a \triangleq \sum_{q=1}^{Q} \gamma_q^a \; u_q(t)$.[3] If each $u_q(t)$ is an independent GP with *prior* mean of zero and covariance $k_q(t,t')$, then the resulting belief $p(\boldsymbol{s}_{1:T})$ is a Multivariate Gaussian with prior mean of zero and covariance determined by the mixing covariance $\text{cov}[s_t^a, s_{t'}^{a'}] = \sum_{q=1}^{Q} \gamma_q^a \; \gamma_q^{a'} \; k_q(t,t')$. This explicit covariance between s_t^a and $s_{t'}^{a'}$ helps to capture the co-evolution and co-adaptation of the network elements within the DNN.

To capture the asymptotic trend of $s_1^a, \ldots, s_T^a$ visualized in Fig. 4.2, we turn to a kernel used for modeling decaying exponential curves known as the "exponential kernel" (Swersky, Snoek, and Ryan Prescott Adams 2014) and set $k_q(t,t') \triangleq {\beta_q}^{\alpha_q}/(t+t'+\beta_q)^{\alpha_q}$ where the MOGP hyperparameters α_q and β_q can be learned using maximum likelihood estimation (Álvarez and Lawrence 2011).

[3]Among the various types of MOGP models (see (Álvarez and Lawrence 2011) for a detailed review), we have chosen such a linear model as its covariance between s_t^a and $s_{t'}^{a'}$ can be computed analytically.

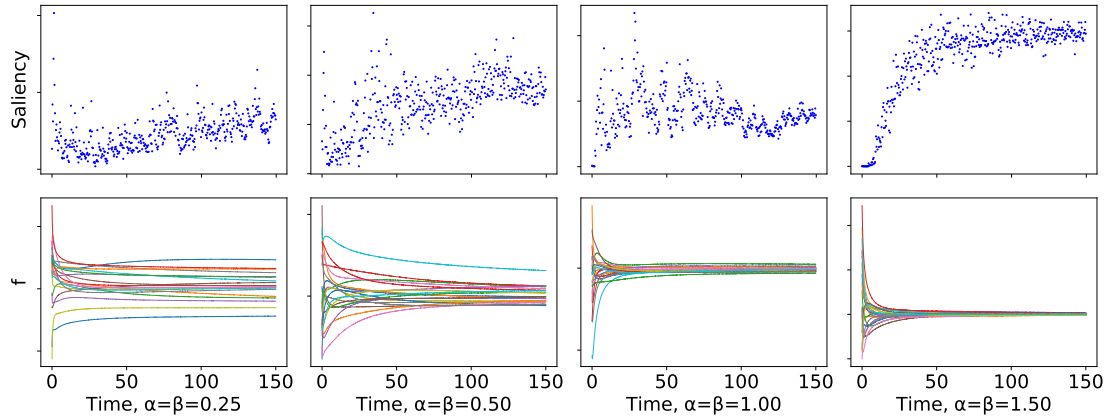

Figure 4.2: (Top) Saliency of different convolutional filters over 150 SGD epochs for a convolutional neural network trained on CIFAR-10 dataset. (Bottom) Function samples drawn from a GP using the exponential kernel with varying hyperparameter values. Both the saliency of different convolutional filters and the function samples from the GP follow an asymptotic, exponentially decaying behavior.

Let the *prior* covariance matrix be $\boldsymbol{K}_{\tau_1:\tau_2} \triangleq [\mathrm{cov}[s_t^a, s_{t'}^{a'}]]_{t,t'=\tau_1,\ldots,\tau_2}^{a,a'=1,\ldots,M}$ for any $1 \leq \tau_1 \leq \tau_2 \leq T$. Then, given a matrix $\tilde{\mathbf{s}}_{1:t}$ of observed saliency measurements (i.e., a realization of $\boldsymbol{s}_{1:t}$), the MOGP regression model can provide a Gaussian predictive belief $p(\boldsymbol{s}_{t'} \mid \tilde{\mathbf{s}}_{1:t}) = \mathcal{N}(\boldsymbol{\mu}_{t'|1:t}, \boldsymbol{K}_{t'|1:t})$ of any future saliency $\boldsymbol{s}_{t'}$ with the following *posterior* mean vector and covariance matrix:

$$\boldsymbol{\mu}_{t'|1:t} \triangleq \boldsymbol{K}_{[t't]}\boldsymbol{K}_{1:t}^{-1}\tilde{\mathbf{s}}_{1:t} \ , \quad \boldsymbol{K}_{t'|1:t} \triangleq \boldsymbol{K}_{t':t'} - \boldsymbol{K}_{[t't]}\boldsymbol{K}_{1:t}^{-1}\boldsymbol{K}_{[t't]}^{\top}$$

where $\boldsymbol{K}_{[t't]} \triangleq [\mathrm{cov}[s_{t'}^a, s_\tau^{a'}]]_{\tau=1,\ldots,t}^{a,a'=1,\ldots,M}$. The a-th element $\mu_{t'|1:t}^a$ of $\boldsymbol{\mu}_{t'|1:t}$ denotes the predictive mean of the saliency $s_{t'}^a$ for $a = 1, \ldots, M$ (i.e., $\boldsymbol{\mu}_{t'|1:t} \triangleq [\mu_{t'|1:t}^a]_{a=1}^M$), while the $[a, a']$-th element $\sigma_{t'|1:t}^{aa'}$ of $\boldsymbol{K}_{t'|1:t}$ denotes the predictive covariance between the saliencies $s_{t'}^a$ and $s_{t'}^{a'}$ for $a, a' = 1, \ldots, M$ (i.e., $\boldsymbol{K}_{t'|1:t} \triangleq [\sigma_{t'|1:t}^{aa'}]_{a,a'=1}^M$).

4.3.2.1 On the Choice of the "Exponential Kernel"

We justify our choice of the exponential kernel as a modeling mechanism by presenting visualizations of saliency measurements collected during training, and com-

paring these to samples drawn from a GP using the exponential kernel $k_q(t, t') \triangleq \beta^\alpha/(t + t' + \beta)^\alpha$. As shown in Fig. 4.2, both the saliency of various convolutional filters and the function samples from the GP exhibit exponentially decaying behavior, which makes the exponential kernel a suitable choice for modeling saliency evolution over time.

We note that the exponential kernel was previously used by Swersky, Snoek, and Ryan Prescott Adams 2014 to model loss curves. Similar to the saliency measurement curves, loss curves also exhibit an asymptotic, exponentially decaying behavior, which further justifies the exponential kernel to be a suitable choice for our saliency modeling task.

4.3.3 Bayesian Early Pruning (BEP) Algorithm

Solving the above optimization problem (Eq. 4.2 and Eq. 4.3) is challenging due to the interplay between $[\boldsymbol{m}_{t'}]_{t'=t}^{T}$, $[B_{t',c}]_{t'=t}^{T}$, and $\boldsymbol{m}_T \cdot \boldsymbol{s}_T$. To tackle this challenge, we instead analyze a lower bound of $\rho_t(\cdot)$:

$$\begin{aligned}\rho_t(\boldsymbol{m}_{t-1}, B_{t,c}, B_s) &= \max_{\boldsymbol{m}_t} \ \mathbb{E}_{p(\boldsymbol{s}_{t+1}|\tilde{\mathbf{s}}_{1:t})}\left[\rho_{t+1}(\boldsymbol{m}_t, B_{t,c} - \|\boldsymbol{m}_t\|_0, B_s)\right] \\ &\geq \max_{\boldsymbol{m}_t} \ \mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\mathbf{s}}_{1:t})}\left[\rho_T(\boldsymbol{m}_t, B_{t,c} - (T-t)\|\boldsymbol{m}_t\|_0, B_s)\right].\end{aligned} \tag{4.4}$$

We prove this lower bound in Appendix A.2. Substituting this lower bound,[4] we define $\hat{\rho}(\cdot)$:

$$\hat{\rho}_t(\boldsymbol{m}_{t-1}, B_{t,c}, B_s) \triangleq \max_{\boldsymbol{m}_t} \mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\mathbf{s}}_{1:t})}[\rho_T(\boldsymbol{m}_t, B_{t,c} - (T-t)\|\boldsymbol{m}_t\|_0, B_s)] \ . \tag{4.5}$$

[4]We omit Eq. 4.3b as it is automatically satisfied due to our lower bound.

This approach allows us to lift Eq. 4.2d from Eq. 4.2, to which we add a Lagrange multiplier and achieve:

$$\hat{\rho}_t(\boldsymbol{m}_{t-1}, B_{t,c}, B_s) \triangleq \max_{\boldsymbol{m}_t} \mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\mathbf{s}}_{1:t})} \left[\hat{\rho}_T(\boldsymbol{m}_t, B_s)\right] + \lambda_t \left(B_{t,c} - (T-t)\|\boldsymbol{m}_t\|_0\right) \quad (4.6)$$

for $t = 1, \ldots, T-1$ and $\hat{\rho}_T$ is defined as ρ_T without constraint 4.2d. Consequently, such a $\hat{\rho}_T$ can be solved in a greedy manner as in Eq. 4.1. Afterwards, we will omit $B_{t,c}$ as a parameter of $\hat{\rho}_T$ as it no longer constrains the solution of $\hat{\rho}_T$. Note that the presence of an *additive* penalty in a *maximization* problem is due to the constraint $B_{T,c} \geq 0 \Leftrightarrow -B_{T,c} \leq 0$, which is typically expected prior to Lagrangian reformulation.

To proceed with the analysis, we show the above optimization problem is submodular in $\boldsymbol{m}_t$. In Eq. 4.6, the problem of choosing $\boldsymbol{m}$ from $\{0,1\}^M$ can be considered as selecting a subset A of indexes from $\{1, \ldots, M\}$ such that $m_t^a = 1$ for $a \in A$, and $m_t^a = 0$ otherwise. Therefore, $P(\boldsymbol{m}) \triangleq \mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\boldsymbol{s}}_{1:t})}[\hat{\rho}_T(\boldsymbol{m}, B_s)]$ can be considered as a set function which we show to be submodular.

Lemma 4.1. *Let* $\boldsymbol{m}'$, $\boldsymbol{m}'' \in \{0,1\}^M$, *and* $e^{(a)}$ *be an arbitrary* M*-dimensional one hot vector with* $1 \leq a \leq M$ *with* $P(\boldsymbol{m}) \triangleq \mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\boldsymbol{s}}_{1:t})}[\hat{\rho}_T(\boldsymbol{m}, B_s)]$. *We have* $P(\boldsymbol{m}' \vee e^{(a)}) - P(\boldsymbol{m}') \geq P(\boldsymbol{m}'' \vee e^{(a)}) - P(\boldsymbol{m}'')$ *for any* $\boldsymbol{m}' \dot{\leq} \boldsymbol{m}''$ *when* $\boldsymbol{m}' \wedge e^{(a)} = 0^M$, *and* $\boldsymbol{m}'' \wedge e^{(a)} = 0^M$.

Here, '$\vee$' and '$\wedge$' represent bitwise OR and AND operations, respectively. The bitwise OR operation is used to denote the *inclusion* of $\boldsymbol{e}^{(a)}$ in $\boldsymbol{m}_t$. The proof for Lemma 4.1 is presented in Appendix A.3. Greedy approximations for submodular optimization incur $O(\|\boldsymbol{m}_{t-1}\|_0^2)$ time, which remains far too slow due to the large

number of network elements in DNNs. To overcome this, we leverage the strong tail decay of multivariate Gaussian density $p(\boldsymbol{s}_T \mid \tilde{\boldsymbol{s}}_{1:t})$ to deliver an efficient approximation procedure. Our approach relies on the following lemma (its proof is in Appendix A.4.):

Lemma 4.2. *Let $\boldsymbol{e}^{(i)}$ be a M-dimensional one-hot vectors with the i^{th} element being 1. $\forall\ 1 \leq a, b \leq M, \boldsymbol{m} \in \{0,1\}^M$ s.t. $\boldsymbol{m} \wedge (\boldsymbol{e}^{(a)} \vee \boldsymbol{e}^{(b)}) = 0^M$. Given a matrix $\tilde{\mathbf{s}}_{1:t}$ of observed saliency measurements, if $\mu^a_{T|1:t} \geq \mu^b_{T|1:t}$ and $\mu^a_{T|1:t} \geq 0$, then*

$$\mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\mathbf{s}}_{1:t})}[\hat{\rho}_T(\boldsymbol{m} \vee \boldsymbol{e}^{(b)})] - \mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\mathbf{s}}_{1:t})}[\hat{\rho}_T(\boldsymbol{m} \vee \boldsymbol{e}^{(a)})] \leq \mu^b_{T|1:t}\ \Phi(\nu/\theta) + \theta\ \phi(\nu/\theta)$$

where $\theta \triangleq \sqrt{\sigma^{aa}_{T|1:t} + \sigma^{bb}_{T|1:t} - 2\sigma^{ab}_{T|1:t}}$, $\nu \triangleq \mu^b_{T|1:t} - \mu^a_{T|1:t}$, and Φ and ϕ are standard normal CDF and PDF, respectively.

Lemma 4.2 indicates that, with high probability, opting for $\boldsymbol{m}_t = \boldsymbol{m} \vee e^{(a)}$ is not a much worse choice than $\boldsymbol{m}_t = \boldsymbol{m} \vee e^{(b)}$ given $\mu^a_{T|1:t} \geq \mu^b_{T|1:t}$ due to the strong tail decay[5] of ϕ and Φ. This admits the following approach to optimize $\hat{\rho}_t$: Starting with $\boldsymbol{m}_t = 0^M$, we consider the inclusion of network elements in $\boldsymbol{m}_t$ by the *descending* order of $\{\mu^a_{T|1:t}\} a = 1..M$ which can be computed analytically using MOGP. A network element denoted by $\boldsymbol{e}^{(a)}$ is included in $\boldsymbol{m}_t$ if it improves the objective in Eq. 4.5. The algorithm terminates once the highest not-yet-included element does not improve the objective function as a consequence of the penalty term outweighing the improvement in $\mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\mathbf{s}}_{1:t})}[\hat{\rho}_T]$. The remaining excluded elements are then pruned.

Following the algorithm sketch above, we define the utility of network element v^a_t with respect to candidate pruning mask $\boldsymbol{m}_t \dot{\leq} \boldsymbol{m}_{t-1}$ which measures the improvement

[5]Note as $\mu^a_{T|1:t} \geq \mu^b_{T|1:t}$, $\Phi(\cdot) \leq 0.5$ and experiences tail decay proportional to $\mu^a_{T|1:t} - \mu^b_{T|1:t}$.

in $\mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\mathbf{s}}_{1:t})}[\hat{\rho}_T]$ as a consequence of inclusion of $\boldsymbol{e}^{(a)}$ in $\boldsymbol{m}_t$:

$$\Delta(a, \boldsymbol{m}_t, \tilde{\mathbf{s}}_{1:t}, B_s) \triangleq \mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\mathbf{s}}_{1:t})}[\hat{\rho}_T(\boldsymbol{e}^{(a)} \vee \boldsymbol{m}_t, B_s) - \hat{\rho}_T(\boldsymbol{m}_t, B_s)]. \tag{4.7}$$

In computing $\Delta(\cdot)$, we take the expectation over the distribution $p(\boldsymbol{s}_T \mid \tilde{\mathbf{s}}_{1:t})$, which utilizes both the predictive mean and variance of the network element. Consquently, the confidence of the MOGP prediction is considered prior to pruning. We can now take a Lagrangian approach to make pruning decisions during iteration t by balancing the utility of network element v_t^a against the change of the penalty (i.e., $\lambda_t(T-t)$), as shown in Algorithm 1. Finally, we show how λ_t offers a probabilistic guarantee in the poor performance of a pruned network element:

Lemma 4.3. *Let $\boldsymbol{e}^{(*)}$ represent a pruned element at time t with the highest predictive mean $\mu^*_{T|1:t} \geq 0$. Given an arbitrary pruned element $\boldsymbol{e}^{(a)}$ at time t, then for all $\delta \in (0,1)$, the following holds:*

$$p\left(\hat{\rho}_T(\boldsymbol{e}^{(a)} \vee \boldsymbol{m}_t, B_s) - \hat{\rho}_T(\boldsymbol{m}_t, B_s) < \frac{\lambda_t}{\delta}(T - t + \epsilon)\right) > 1 - \delta$$

where $\epsilon \triangleq \lambda_t^{-1}\left[\mu^a_{T|1:t}\Phi(\nu/\theta) + \theta\ \phi(\nu/\theta)\right]$ with $\theta \triangleq \left(\sigma^{**}_{T|1:t} + \sigma^{aa}_{T|1:t} - 2\sigma^{*a}_{T|1:t}\right)^{1/2}$, and $\nu \triangleq \mu^a_{T|1:t} - \mu^*_{T|1:t}$.

The proof of the above is in Appendix A.5. The above lemma shows that λ_t acts as a probabilistic guarantee of the poor performance of a pruned network element. A smaller λ_t offers a higher probability in the poor performance of an element prior to pruning. Consequently, λ_t is inversely correlated with training cost where a lower λ_t requires more training cost as fewer network elements are pruned. This offers a

trade-off between training cost and test-time efficacy using the penalty parameter λ_t.

Due to the relatively expensive cost of performing early pruning, we chose to early prune every T_{step} SGD iterations. Typically T_{step} was chosen to correspond to 10-20 epochs of training. To compute $\Delta(\cdot)$ we sampled from $p(\boldsymbol{s}_T \mid \tilde{\boldsymbol{s}}_{1:t})$ and used a greedy selection algorithm per sample as in Eq. 4.1. During implementation, we also enforced an additional hard constraint $\|\boldsymbol{m}_t\|_0 \geq B_s$ which we believe is desirable for practicality reasons. We used a fixed value of $B_{1,c} = \|\boldsymbol{m}_0\|_0 T_0 + B_s(T - T_0)$ in all our experiments.

Algorithm 1 Bayesian Early Pruning

Require: $\mathcal{N}$, $\boldsymbol{v}_1$, T, $B_{1,c}$, B_s, λ, (LITE, B_0) ▷ DNN $\mathcal{N}$, Lagrangian penalties λ

if LITE **then** ▷ See Section 4.3.4.
  $\boldsymbol{s}_0 \leftarrow \text{evaluate}(\mathcal{N}_{\boldsymbol{v}_1})$ ▷ Evaluate saliency at initialization.
  $m_0 \leftarrow \arg\max_{\boldsymbol{m}_0 \in \{0,1\}^M} \sum_{a=1}^{M} \boldsymbol{m}_0 \cdot \boldsymbol{s}_0$ subject to $\|\boldsymbol{m}_0\|_0 \leq B_0$
  $prune(\boldsymbol{v}_1, \boldsymbol{m}_0)$
$\tilde{\mathbf{s}}_{1:T_0} \leftarrow \text{train}(\mathcal{N}_{\boldsymbol{v}_1}, T_0)$ ▷ Train for T_0 iterations to create seed dataset.
$B_{T_0,c} \leftarrow B_{1,c} - T_0\, dim(\boldsymbol{v}_1)$ ▷ Track training cost expenditure.
for $k \leftarrow 0, \ldots, \frac{T-T_0}{T_{step}}$; $t \leftarrow T_0 + kT_{step}$ **do**
  $\boldsymbol{\mu}_{T|1:t}, \sigma_{T|1:t} \leftarrow MOGP(\tilde{\mathbf{s}}_{1:t})$ ▷ Train and perform inference.
  $\boldsymbol{s}_T \leftarrow argsort(-\boldsymbol{\mu}_{T|1:t})$ ▷ Sort descending.
  $\boldsymbol{m}_t \leftarrow 0^{dim(\boldsymbol{v}_t)}$ ▷ Initial pruning mask.
  for $a \leftarrow \boldsymbol{s}_T^1, \ldots, \boldsymbol{s}_T^{dim(\boldsymbol{v}_t)}$ **do** ▷ Consider each network element.
    if $B_{t,c} - (T-t)\|\boldsymbol{m}_t\|_0 > 0$ **then**
      $\boldsymbol{m}_t = \boldsymbol{m}_t \vee \boldsymbol{e}^{(a)}$
    else if $\Delta(a, \boldsymbol{m}_t, \tilde{\mathbf{s}}_{1:t}, B_s) \geq \lambda_t(T-t)$ **then** ▷ *Utility vs. penalty.*
      $\boldsymbol{m}_t = \boldsymbol{m}_t \vee \boldsymbol{e}^{(a)}$
    else
      break
  $prune(\boldsymbol{v}_t, \boldsymbol{m}_t)$ ▷ $dim(\boldsymbol{v}_t)$ is reduced here.
  $B_{t+T_{step},c} \leftarrow B_{t,c} - T_{step}\|\boldsymbol{m}_t\|_0$
  $\tilde{\mathbf{s}}_{t+1:t+T_{step}} \leftarrow train(\mathcal{N}_{\boldsymbol{v}_t}, T_{step})$ ▷ Continue training.
return $\mathcal{N}$

4.3.4 BEP-LITE

We further reduce the training cost of BEP by combining with pruning at initialization. This combination is motivated by noticing that pruning at initialization methods (N. Lee, Ajanthan, and Torr 2019; Chaoqi Wang, G. Zhang, and Grosse 2020) implicitly utilize the following predictive model of saliency:

$$p(\boldsymbol{s}_T) \triangleq \delta(\boldsymbol{s}_0) \tag{4.8}$$

where δ represents the Dirac delta function. We observe in validation that this predictive model is effective to identify the *poorest performing elements*.[6] Thus, we may prune at initialization by solving the following optimization problem:

$$\max_{\boldsymbol{m}_0 \in \{0,1\}^M} \sum_{a=1}^{M} \boldsymbol{m}_0 \cdot \boldsymbol{s}_0 \quad \text{subject to} \quad \|\boldsymbol{m}_0\|_0 \leq B_0 \tag{4.9}$$

where $B_0 \geq B_s$ and is chosen to yield 10%-20% training cost overhead over pruning at initialization. Consequently, pruning at initialization is used as a permissive heuristic to determine $\boldsymbol{m}_0$, with the remainder of the pruning decisions made using BEP as in Algorithm 1. This fusion of techniques, which we term BEP-LITE, significantly reduces training cost without adversely affecting test performance (Section 4.4).

4.3.5 Dynamic Penalty Scaling

In BEP, each optimization problem $\hat{\rho}_t(\cdot)$ has a corresponding Lagrange multiplier λ_t. This requires several hyperparameters, one for each pruning iteration. Optimizing these hyperparameters is costly, and thus undesirable. Due to this, we propose

[6] See Section 4.4.1.3 for verification.

determining λ_t dynamically using a feedback loop utilizing a singular Lagrange multiplier λ.

A proportional feedback loop can be defined as follows[7]:

$$\lambda_t \triangleq \lambda + K_p \times e(t) \tag{4.10}$$

where $K_p \geq 0$ is a proportional constant which modulates λ_t according to a signed measure of error $e(\cdot)$ at time t. Note that $\lambda_t \geq \lambda$ as $e(t) \geq 0$, and the opposite occurs if $e(t) \leq 0$, which allows the *error* to serve as *feedback* to determine λ_t. Implicitly, λ_t asserts some control over $e(t+1)$ and closes the feedback loop.

Traditional approaches to determine K_p do not work in our case as λ may vary over several orders of magnitude. Consequently, a natural choice for K_p is λ itself which preserves the same order of magnitude between K_p and λ:

$$\lambda_t = \lambda + \lambda \times e(t) = \lambda(1 + e(t)). \tag{4.11}$$

Here we make two decisions to adapt the above to our task. First, as λ is likely to be extremely small, we use exponentiation, as opposed to multiplication. Secondly as $\lambda \leq 1$ in practice, we use $1 - e(t)$ as an exponent:

$$\lambda_t = \lambda^{\wedge}\left[1 - e(t)\right] = \lambda\left[(1/\lambda)^{\wedge} e(t)\right]. \tag{4.12}$$

The above equation is complete with our definition of $e(t)$:

$$e(t) \triangleq (T - t)\|\boldsymbol{m}_t\|_0 / B_{t,c} - 1. \tag{4.13}$$

The signed error is determined by the discrepancy between the anticipated compute required to complete training $(T - t)\|\boldsymbol{m}_t\|_0$, vs. the remaining budget $B_{t,c}$ with

[7]This approach is inspired from proportional-integral-derivative (PID) controllers (Bellman 2015), see Åström et al. 1993 for an introductory survey.

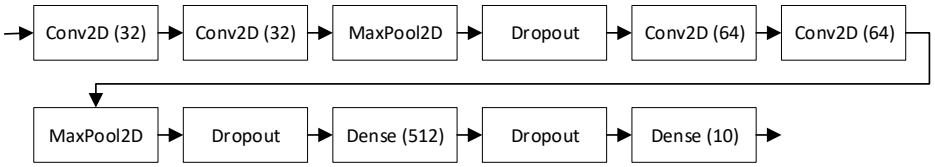

Figure 4.3: Small scale model neural network architecture for CIFAR-10. Parentheticals indicate the number of convolutional filters, or neurons in a layer. The receptive field size for convolution is $(3, 3)$, Max Pooling is done with receptive field size $(2, 2)$.

$e(t) = 0$ if the two are equal. This is a natural measure of feedback for λ as we expect the two to be equal if λ is serving well to *early prune* the network.

4.4 Experiments

This section empirically validates the efficacy of our proposed methods. In particular, we will demonstrate: (a) The effectiveness of our MOGP modeling approach at inferring future saliency measurements; (b) The early pruning performance of BEP compared to related works and the trade-off between training cost vs. test-time efficacy; and (c) The robustness of the BEP performance in its hyperparameter tuning.

To avoid the large cost in validating the above contributions in various settings, we first evaluate our saliency modeling approach as well as our BEP and BEP-LITE algorithms using a small-scale network: a CNN model trained on the CIFAR-10, and CIFAR-100 dataset. The model architecture is presented in Fig. 4.3 and consists of 4 convolutional layers followed by a fully connected layer. MaxPooling and Dropout is also utilized in the architecture, similar to VGG-16 (Simonyan and Zisserman 2015b).

The proposed BEP algorithm is compared with several pruning methods applied at initialization stage: (a) *Random*: Random pruning; (b) *SNIP* (N. Lee, Ajanthan, and Torr 2019); (c) *GraSP* (Chaoqi Wang, G. Zhang, and Grosse 2020); (d) *PFS*: PruneFromScratch (Yulong Wang et al. 2020); and (e) *EagleEye* (B. Li et al. 2020): A pruning-after-training approach which is applied to the initialization stage for comparison.

Following the small scale experiments, we apply BEP and BEP-LITE to prune ResNet-50 on the ImageNet dataset and compare against related works. For our ResNet-50 we compare against (a) *IterSnip* (Jorge et al. 2021); and (b): *SynFlow* (Tanaka et al. 2020) as well as previously mentioned related works. Our ResNet-50 validation shows that BEP is able to train networks with higher accuracy when compared to related work. Furthermore, utilizing the BEP-LITE heuristic, these networks can be trained with only a small amount of training cost overhead when compared to pruning at initialization.

In this work, we use training FLOPs to measure training cost. Due to the continued growth in training cost of DNNs, we focus on the task of pruning a large percentage of the DNN. Due to the cubic time complexity of fitting MOGPs, we used a variational approximation (Hensman, A. Matthews, and Ghahramani 2015). In all of our models, we used 60 variational inducing points per latent function. The GPflow library (A. Matthews et al. 2017) is used to build our MOGP models.

Table 4.2: Comparing log-likelihood (standard error) of test data for independent GPs (GP) vs. MOGP with n latent functions (n-MOGP) on different size of collected saliency measurements from CIFAR-10 and CIFAR-100 training. The log-likelihoods are given as a multiple of -10^4 (lower is better). MOGP outperforms GP, particularly on the small dataset. Using additional latent functions improves MOGP modeling with diminishing returns. The large dataset is easier to model due to an overabundance of data, thus MOGP may show limited improvement due to task simplicity (e.g., see Layer (Lyr) 3, Large dataset). Results are averaged over 20 runs. Extremely large values are due to the GP model being unable to fit the data.

CIFAR-10									
	Small dataset			Medium dataset			Large dataset		
	Lyr 1	Lyr 2	Lyr 3	Lyr 1	Lyr 2	Lyr 3	Lyr 1	Lyr 2	Lyr 3
GP	1.19(0.5)	1.08(0.06)	1.07(1.07)e5	0.96(0.04)	0.93(0.03)	2.47(0.04)	0.49(0.01)	0.48(0.01)	1.33(0.02)
4-MOGP	1.15(0.05)	0.89(0.06)	2.44(0.05)	0.91(0.02)	0.80(0.03)	2.20(0.03)	0.38(0.02)	0.39(0.02)	1.25(0.02)
8-MOGP	1.09(0.04)	0.86(0.05)	2.38(0.04)	0.84(0.03)	0.78(0.03)	2.16(0.03)	0.32(0.01)	0.35(0.02)	1.20(0.02)
18-MOGP	0.97(0.04)	**0.80**(0.05)	2.33(0.04)	0.89(0.03)	0.76(0.03)	2.13(0.03)	0.31(0.01)	0.35(0.02)	1.20(0.02)
32-MOGP	**0.96**(0.06)	0.81(0.06)	**2.32**(0.04)	**0.79**(0.03)	**0.74**(0.03)	**2.13**(0.03)	**0.31**(0.01)	**0.34**(0.02)	**1.20**(0.02)
CIFAR-100									
	Small dataset			Medium dataset			Large dataset		
	Lyr 1	Lyr 2	Lyr 3	Lyr 1	Lyr 2	Lyr 3	Lyr 1	Lyr 2	Lyr 3
GP	0.75(0.06)	5.7(5.7)e4	5.6(5.6)e4	0.64(0.04)	0.70(0.04)	**2.13**(**0.05**)	3.4(3.4)e3	0.31(0.02)	1.06(0.02)
4-MOGP	0.79(0.05)	0.98(0.12)	3.13(0.10)	0.44(0.04)	0.60(0.10)	2.29(0.06)	0.12(0.01)	0.24(0.03)	1.07(0.03)
8-MOGP	0.65(0.05)	0.89(0.11)	3.00(0.09)	0.38(0.04)	0.60(0.10)	2.20(0.06)	0.10(0.01)	0.18(0.01)	1.02(0.03)
18-MOGP	**0.62**(0.05)	**0.84**(0.11)	2.93(0.10)	0.36(0.03)	**0.56**(0.10)	2.22(0.07)	0.09(0.01)	0.18(0.01)	1.01(0.03)
32-MOGP	0.65(0.05)	0.85(0.09)	**2.89**(0.10)	**0.36**(0.03)	0.59(0.10)	2.16(0.06)	**0.09**(0.02)	**0.18**(0.01)	**1.00**(0.03)

4.4.1 Small-Scale Experiments

4.4.1.1 Saliency Modeling Evaluation

A key assertion in our approach is the importance of inferring the future saliency of network elements given a dataset of past saliency measurements. This inference process is important because it underpins the pruning decisions made by BEP. To verify that our MOGP approach demonstrates strong performance at this task, we compare MOGP vs. GP belief modeling with GP assuming independence in saliency measurements across network elements (i.e., $p(\boldsymbol{s}_{1:T}) \triangleq \prod_{a=1}^{M} p(\boldsymbol{s}^a_{1:T})$). To validate this assertion, we collect saliency measurements of convolutional filters and neurons (network elements) by instrumenting the training process of our Small scale CNN model on the CIFAR-10/CIFAR-100 dataset.[8] We train the belief models

[8]Complete experimental setups are detailed in Appendix A.7.1.

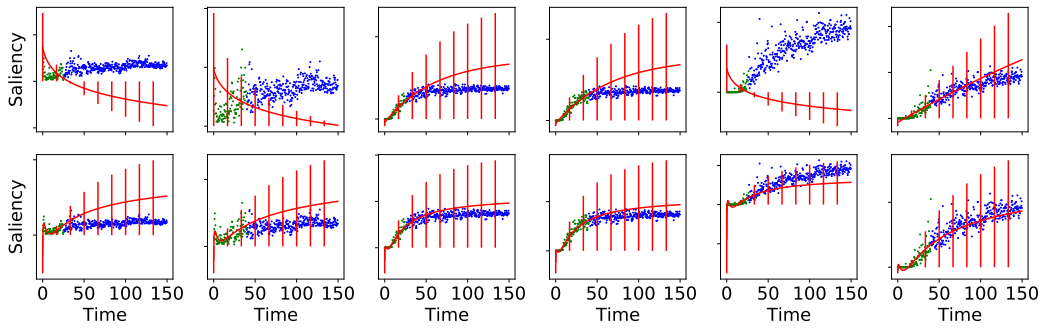

Figure 4.4: Visualization of qualitative differences between GP and MOGP prediction. Top: GP. Bottom: 18-MOGP. Dataset is separated into training (green) set of observations and future saliency forms the validation (blue) set. Posterior belief of the saliency is visualized as predictive mean (red line), and 95% confidence interval (error bar).In many cases (e.g., top left graph), GP is unable to predict the long term trends of the data due to irregular element saliency observations. However, MOGP is able to overcome data irregularities by utilizing correlations between saliency of the network elements.

with a small ($t = [0, 26]$ epochs), medium ($t = [0, 40]$ epochs), and large ($t = [0, 75]$ epochs) training dataset of saliency measurements. For GPs, a separate model was trained per network element (convolutional filter, or neuron). For MOGP, all network elements in a single layer shared one MOGP model. We measure these models' performance at inferring the future (unobserved) saliency measurements using log-likelihood and present the results in Table 4.2 for CIFAR-10 and CIFAR-100. Our MOGP approach better approximates the future saliency of network elements than a GP approach. In addition, increasing the size of the training dataset (e.g., from small to large) significantly improves the log-likelihood. A larger dataset is more accurate for pruning decisions, but collecting a larger dataset incurs more training cost. This is consistent with our focus on the trade-off between training cost vs. test-time efficacy.

We visualize the qualitative differences between GP and MOGP prediction in

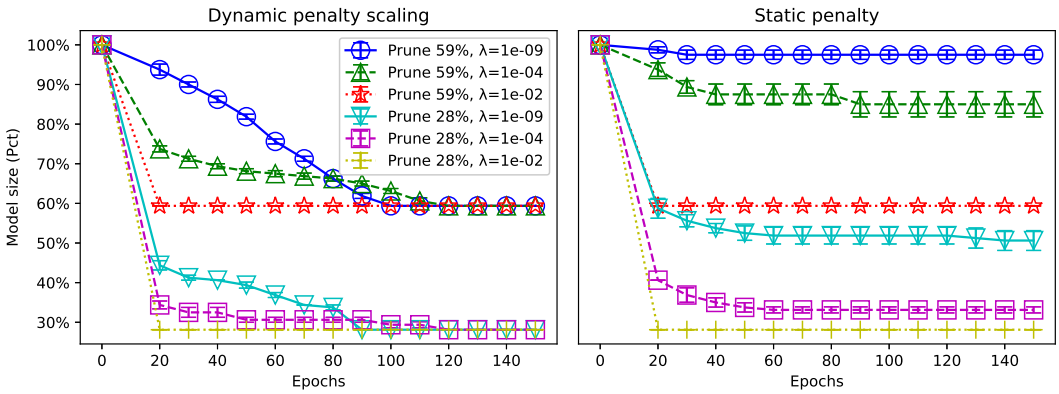

Figure 4.5: Comparing dynamic penalty scaling vs. static on pruning tasks on CIFAR-100 CNN training. Dynamic penalty scaling encourages gradual pruning across a wide variety of settings of λ.

Fig. 4.4. We observe that MOGP is able to capture the long term trend of saliency curves with significantly less data than GP. In many cases, GP is unable to predict the long term trends of the data due to irregular element saliency observations. However, MOGP is able to overcome data irregularities by utilizing correlations between saliency of the network elements.

4.4.1.2 Dynamic Penalty Scaling

We applied the early pruning algorithm on the aforementioned architecture, and training regimen. We investigated the behavior of the penalty parameter, λ. We compare dynamic penalty scaling, and penalty without scaling in Fig. 4.5 using $T_0 = 20$ epochs, $T_{step} = 10$ epochs for two pruning tasks. Dynamic penalty scaling encourages gradual pruning across a wide variety of settings of λ. We use dynamic penalty scaling in the remainder of our validation.

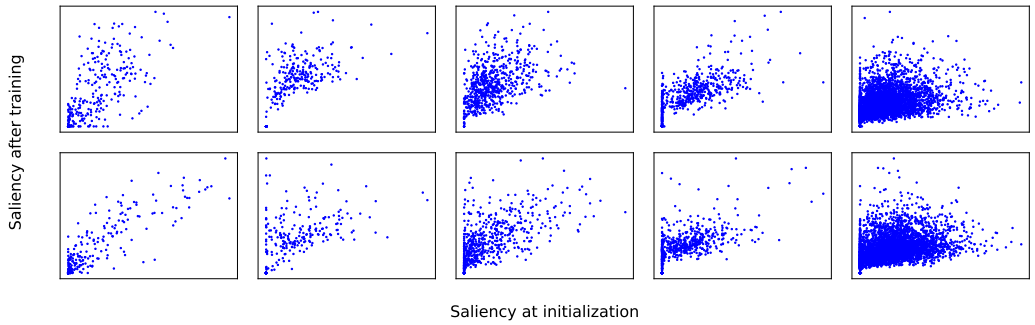

Figure 4.6: Correlation between saliency of network elements at initialization, and saliency of network elements after training. Top: CIFAR-10, Bottom: CIFAR-100. Left-to-right: Layers 1 through 5 of the convolutional neural network.

4.4.1.3 BEP-LITE Heuristic

For BEP-LITE we utilize the following predictive model of saliency

$$p(\boldsymbol{s}_T) \triangleq \delta(\boldsymbol{s}_0) \tag{4.14}$$

where δ represents the Dirac delta function. To verify the effectiveness of this model as a *permissive heuristic* for BEP, we plot the relation between saliency at initialization and after training completion.

Using the same experimental setup as Section 4.4.1.1, we plot the saliency measurements collected at initialization and after training completion. This is presented in Fig. 4.6. Saliency at initialization well correlates with saliency after training, hence demonstrating the validity of our heuristic. Following this observation, we utilize the above predictive model as a permissive heuristic applied at initialization to speed up the BEP algorithm. We utilize the GraSP (Chaoqi Wang, G. Zhang, and Grosse 2020) heuristic for initialization pruning in BEP-LITE due to its strong empirical performance.

Table 4.3: Performance (standard error) of the tested algorithms with varying inference FLOPs (percentage of pruned FLOPs) for CIFAR-10 and CIFAR-100 on the small scale CNN model. The BEP is tested with varying penalty: λ = 1e−2, 1e−4, and 1e−7. Unpruned baseline train and inference FLOPs are 1.9e13 and 2.5M, respectively.

	CIFAR-10 (Small scale CNN model)							
	223K Inference FLOPs (91%)		95K Inference FLOPs (96.2%)		24K Inference FLOPs (99.0%)		6K Inference FLOPs (99.8%)	
	Val Acc	Train FLOPs	Val Acc	Train FLOPs	Val Acc	Train FLOPs	Val Acc	Train FLOPs
Random	72.3(0.6)%	1.7e12	66.5(0.7)%	7.1e11	41.4(7.9)%	1.8e11	14.5(4.5)%	4.6e10
SNIP	75.4(4.7)%	1.7e12	67.7(0.7)%	7.1e11	50.8(0.8)%	1.8e11	29.4(4.9)%	4.6e10
GraSP	74.6(0.6)%	1.7e12	66.5(0.9)%	7.1e11	50.7(0.6)%	1.8e11	32.9(1.0)%	4.6e10
PFS	71.3(2.0)%	1.7e12	59.9(5.8)%	7.1e11	43.3(2.5)%	1.8e11	31.7(1.7)%	4.6e10
EagleEye	60.9(7.5)%	1.7e12	44.6(11.3)%	7.1e11	47.8(7.8)%	1.8e11	28.7(4.5)%	4.6e10
BEP 1×10^{-2}	75.9(0.3)%	4.0e12(9.5e10)	69.7(0.4)%	3.1e12(2.1e9)	54.8(1.0)%	2.6e12(4.1e9)	18.9(5.4)%	2.5e12(2.2e10)
BEP 1×10^{-4}	75.4(1.7)%	4.3e12(3.7e10)	70.5(3.2)%	3.3e12(3.3e10)	55.7(0.9)%	2.6e12(8.4e9)	**36.1**(**1.1**)%	2.5e12(2.0e9)
BEP 1×10^{-7}	**76.0**(**0.1**)%	4.5e12(1.4e11)	**70.6**(**0.2**)%	3.4e12(6.6e10)	**56.2**(**0.4**)%	2.7e12(1.4e10)	30.4(5.1)%	2.5e12(2.2e9)
	CIFAR-100 (Small scale CNN Model)							
	251K Inference FLOPs (89.4%)		113K Inference FLOPs (95.7%)		33K Inference FLOPs (98.8%)		10K Inference FLOPs (99.6%)	
	Val Acc	Train FLOPs	Val Acc	Train FLOPs	Val Acc	Train FLOPs	Val Acc	Train FLOPs
Random	27.8(6.7)%	1.9e12	27.1(0.7)%	8.5e12	3.7(2.7)%	2.5e11	1.0(0.0)%	8.0e10
SNIP	22.9(9.0)%	1.9e12	15.7(6.1)%	8.5e12	9.0(3.7)%	2.5e11	2.2(1.2)%	8.0e10
GraSP	28.4(7.0)%	1.9e12	22.6(5.4)%	8.5e12	13.9(3.2)%	2.5e11	1.0(0.0)%	8.0e10
PFS	37.3(0.9)%	1.9e12	26.9(4.0)%	8.5e12	19.3(2.4)%	2.5e11	8.5(0.7)%	8.0e10
EagleEye	19.8(12.0)%	1.9e12	20.2(7.1)%	8.5e12	12.6(2.6)%	2.5e11	4.7(2.2)%	8.0e10
BEP 1×10^{-2}	40.6(0.2)%	4.2e12(4.8e9)	32.2(0.6)%	3.3e12(2.2e9)	19.1(0.5)%	2.8e12(4.3e8)	7.1(1.6)%	2.7e12(1.3e9)
BEP 1×10^{-4}	**41.3**(**0.3**)%	4.6e12(3.7e10)	**32.4**(**0.3**)%	3.5e12(2.3e10)	**19.7**(**0.8**)%	2.9e12(6.5e10)	**8.5**(**0.8**)%	2.7e12(4.7e10)
BEP 1×10^{-7}	40.6(0.2)%	4.8e12(1.0e11)	33.0(0.5)%	3.5e12(5.9e10)	19.5(0.5)%	2.9e12(1.2e10)	6.6(1.5)%	2.7e12(5.2e9)

4.4.1.4 BEP on CIFAR-10/CIFAR-100 Dataset

We apply the tested algorithms to prune a portion of filters/neurons of each layer[9] and evaluate their performance with various degrees of pruning percentage. As shown in Table 4.3, our approach better preserves performance at equivalent pruning percentage. A lower penalty λ yields higher performing results but larger training FLOPs, which shows that λ in BEP serves well at balancing predictive performance vs. training cost. A clear superiority of BEP in validation accuracy can be observed when the pruning percentage is large (i.e., right column of Table 4.3). Although PruneFromScratch (Yulong Wang et al. 2020) demonstrates comparable performance to BEP in some cases, we show in more complex experiments (Section 4.4.2) a

[9]We do pruning *per layer* instead of across the whole network since the saliency measurement has been known to not work well in comparing network element efficacy across layers (see Appendix A.1 and A.2 of (Molchanov et al. 2017) and Section 3 of (Chaoqi Wang, G. Zhang, and Grosse 2020)). Developing novel saliency functions which overcome this shortcoming is outside the scope of this work.

Table 4.4: Ablation study showing validation accuracy (standard error) with varying early pruning hyperparameters: MOGP variational inducing points (Ind. pnts.), MOGP latent functions (Lat. func.), and T_{step}. Default setting for hyperparameters are 60, 1.0×, and 10 respectively. Outside of the highest sparsity setting ($6K$ Inf. FLOPs), the validation accuracy of DNN is robust to changes of all hyperparameters, with mild degradation observed in the extremal settings.

		CIFAR-10			CIFAR-100		
		$95K$ Inf.	$24K$ Inf.	$6K$ Inf.	$95K$ Inf.	$24K$ Inf.	$6K$ Inf.
Ind. pnts.	1	70.2(0.3)%	55.6(0.3)%	32.1(0.5)%	31.7(0.6)%	19.1(0.2)%	6.6(1.2)%
	2	70.4(0.2)%	55.8(0.3)%	35.4(0.2)%	33.3(0.4)%	18.8(0.2)%	8.0(0.2)%
	5	70.5(0.2)%	56.3(0.2)%	35.9(0.2)%	32.9(0.2)%	19.3(0.3)%	4.7(1.3)%
	10	70.4(0.5)%	56.1(0.4)%	30.6(4.7)%	32.6(0.4)%	19.1(0.6)%	7.4(1.5)%
	26	69.6(0.4)%	56.8(0.2)%	29.7(4.9)%	31.7(0.6)%	19.2(0.5)%	8.3(0.7)%
	40	70.9(0.1)%	55.6(0.6)%	30.5(5.1)%	32.3(0.7)%	19.6(0.3)%	6.6(1.4)%
	60	70.5(3.2)%	55.7(0.9)%	36.1(1.1)%	32.4(0.3)%	19.7(0.8)%	8.5(0.8)%
	90	70.4(0.3)%	55.1(0.7)%	35.5(1.9)%	32.6(0.4)%	18.5(0.6)%	8.7(0.3)%
Lat. func.	0.10×	70.1(0.3)%	55.3(0.7)%	30.9(4.7)%	31.8(0.4)%	20.2(0.2)%	5.9(1.8)%
	0.15×	70.6(0.2)%	55.1(0.4)%	25.9(5.8)%	32.1(0.3)%	20.2(0.2)%	6.5(1.3)%
	0.20×	69.8(0.1)%	56.0(0.3)%	20.4(5.7)%	33.3(0.2)%	19.3(0.5)%	4.7(1.3)%
	0.25×	70.4(0.4)%	55.6(0.8)%	35.8(0.2)%	32.6(0.3)%	16.3(3.8)%	7.4(1.8)%
	0.50×	70.0(0.2)%	56.9(0.4)%	34.5(0.6)%	32.1(0.5)%	18.9(0.7)%	7.0(1.5)%
	1.0×	70.5(3.2)%	55.7(0.9)%	36.1(1.1)%	32.4(0.3)%	19.7(0.8)%	8.5(0.8)%
	2.0×	69.8(0.3)%	55.7(0.7)%	34.8(0.5)%	32.0(0.4)%	20.8(0.2)%	7.7(0.4)%
T_{step}	2	69.2(0.5)%	54.7(0.6)%	29.4(5.0)%	32.1(0.2)%	20.0(0.3)%	4.3(1.5)%
	5	70.3(0.2)%	55.6(0.5)%	31.6(5.4)%	32.7(0.4)%	19.4(0.4)%	5.2(1.8)%
	10	70.5(3.2)%	55.7(0.9)%	36.1(1.1)%	32.4(0.3)%	19.7(0.4)%	8.5(0.8)%
	20	70.3(0.2)%	56.2(0.1)%	29.8(5.0)%	32.8(0.5)%	19.6(0.4)%	6.8(1.5)%
	40	70.8(0.3)%	55.5(0.6)%	34.6(0.5)%	32.8(0.2)%	18.9(0.4)%	8.3(0.5)%
	80	72.0(0.3)%	58.4(1.8)%	39.2(6.6)%	33.9(0.5)%	22.3(0.7)%	9.5(0.8)%
	100	74.3(0.4)%	62.4(0.6)%	36.7(6.4)%	37.1(0.2)%	24.6(0.4)%	9.4(2.0)%

significant performance gap emerges. Although BEP incurs larger training FLOPs than other tested algorithms, we can further reduce the training cost via BEP-LITE as will be shown in Section 4.4.2. EagleEye achieves much lower validation accuracy than other tested algorithms, which implies that an *after training* pruning method typically does not work well when applied to the initialization stage for reducing training cost.

4.4.1.5 Establishing Robustness through Ablation Study

The objective of BEP is to reduce the cost for DNN training. As such, hyperparameter tuning of BEP on a per DNN architecture basis is not feasible due to its expensive cost. Thus, we check the robustness of BEP and MOGP hyperparameters in this section, which demonstrates that tuning is not necessary on a per DNN architecture basis.

We vary the number of MOGP variational inducing points, MOGP latent functions as well as T_{step}. Under these varying conditions we test the performance of BEP 1×10^{-4} on CIFAR-10/CIFAR-100 at $95K$, $24K$, and $6K$ inference FLOPs with our small scale CNN model. As shown in Table 4.4, we observe that in general, the validation accuracy of the pruned DNN is robust to the changes of all hyperparameters. Degradation is observed in extremal hyperparameter settings. Reducing the inducing points, and latent functions has a strong effect on the effectiveness of the algorithm in the extremal setting (e.g., 6K Inf. FLOPS and minimal inducing points or latent functions). However, this can be easily avoided in practice. Pruning with a large T_{step} offers improved performance, however this correspondingly increases training cost. The hyperparameter robustness in our approach demonstrates the feasibility of applying BEP to "never-before-seen" network architectures and datasets without additional hyperparameter tuning.

4.4.2 ResNet Early Pruning

We train ResNet-50 with BEP and other tested algorithm for 100 epochs on 4× Nvidia Geforce GTX 1080Ti GPUs. More experimental details can be found in

Table 4.5: BEP and BEP-LITE vs. related work for ResNet-50 on ImageNet dataset. We vary the inference FLOPs (percentage of pruned FLOPs) of the model after training. 'Model' refers to all inclusive overhead of the pruning algorithm. PFS algorithm failed to prune to the desired sparsity in the unlisted scenarios. Unpruned baseline train and inference FLOPs are 9.3e17 and 7.3e9, respectively.

	ResNet-50											
	9.7e8 Inf. FLOPs (86.7%)			4.4e8 Inf. FLOPs (94.0%)			4.2e8 Inf. FLOPs (94.3%)			1.7e8 Inf. FLOPs (97.7%)		
	Acc	Train FLOPs	Model	Acc	Train FLOPs	Model	Acc	Train FLOPs	Model	Acc	Train FLOPs	Model
Random	69.2%	1.2e17	0.0h	51.6%	5.7e16	0.0h	35.7%	5.4e16	0.0h	34.3%	2.3e16	0.0h
SNIP	69.3%	1.2e17	0.7h	50.9%	5.7e16	0.7h	35.1%	5.4e16	0.7h	31.8%	2.3e16	0.7h
GraSP	69.3%	1.2e17	2.7h	52.2%	5.7e16	2.7h	36.7%	5.4e16	2.7h	34.1%	2.3e16	2.7h
IterSNIP	0.4%	1.2e17	1.4h	0.4%	5.7e16	1.4h	0.5%	5.4e16	1.4h	0.4%	2.3e16	1.4h
SynFlow	32.3%	1.2e17	1.2h	0.3%	5.7e16	1.2h	0.1%	5.4e16	1.2h	0.1%	2.3e16	1.2h
PFS	60.3%	1.2e17	1.6h	-	-	-	-	-	-	-	-	-
EagleEye	26.1%	1.2e17	18h	36.6%	5.7e16	18h	24.1%	5.4e16	18h	26.6%	2.3e16	18h
BEP-LITE 1×10^{-4}	69.7%	1.4e17	2.6h	**53.7**%	6.6e16	2.9h	39.5%	6.2e16	2.8h	**37.0**%	2.6e16	2.4h
BEP 1×10^{-4}	**70.0**%	2.2e17	1.9h	53.5%	1.7e17	2.4h	**40.0**%	1.6e17	2.3h	36.1%	1.3e17	1.6h
	ResNet-50 (Compare with PruneTrain)											
	1.4e9 Inf. FLOPs (80.7%)			5.4e8 Inf. FLOPs (92.6%)			1.3e8 Inf. FLOPs (98.2%)			3.0e7 Inf. FLOPs (99.6%)		
PruneTrain	69.2%	2.9e17	0.0h	60.6%	2.0e17	0.0h	40.6%	7.2e16	0.0h	8.3%	5.8e16	0.0h
BEP-LITE 1×10^{-4}	71.4%	1.4e17	2.4h	66.3%	6.6e16	2.9h	**53.8**%	6.2e16	2.6h	**20.6**%	2.6e16	1.9h
BEP 1×10^{-4}	**71.6**%	2.4e17	2.8h	**66.8**%	1.7e17	3.0h	53.6%	1.6e17	1.9h	20.6%	1.3e17	1.7h

Appendix A.7.1. We used $\lambda = 1e{-}4$ because of its strong performance in our smaller scale experiments. As can be observed in Table 4.5, the proposed methods achieve higher validation accuracy than other tested algorithms, with BEP-LITE showing only a modest 15% increase in training FLOPs over pruning at initialization. BEP-LITE achieves a 85% training cost reduction over BEP for 1.7e8 inference FLOPs while achieving superior validation accuracy. The modeling and pruning overhead of our algorithm is comparable to other tested algorithms. PruneFromScratch (Yulong Wang et al. 2020) shows severe degradation when compared to BEP in the 86.7% pruned FLOPs experiment, and fails to prune altogether in higher sparsity settings. IterSnip (Jorge et al. 2021) and SynFlow (Tanaka et al. 2020) are unable to prune effectively at high pruning ratios, with severe degradation observed in all tested scenarios. EagleEye continues showing poor performance, which demonstrates the inability of *pruning-after-training* techniques to be applied to the early pruning problem. In particular, BEP and BEP-LITE significantly outperforms competing

Table 4.6: Timing evaluation. Overhead time consists of disk I/O, and image decoding. Unpruned baseline is 47h wall-clock and 31h GPU time.

		ResNet-50 (Timing)								
		86.7%	94.0%	94.3%	97.7%			86.7%	94.0%	94.3%	97.7%
BEP-LITE	GPU	12.2h	6.9h	6.4h	3.8h	BEP	GPU	14.6h	9.2h	9.2h	7.1h
	Wall-clock	27.8h	22.6h	22.4h	19.4h		Wall-clock	30.2h	25.2h	24.9h	22.7
	Overhead	15.6h	16.0h	15.8h	15.6h		Overhead	15.6h	15.9h	15.8h	15.6h
	Model	2.6h	2.9h	2.8h	2.4h		Model	1.9h	2.4h	2.3h	1.6h

approaches at larger pruning ratios. This improvement is crucial as DNNs continue to grow in size and require considerable pruning to allow training and inference on commodity hardware. We note that PruneTrain does not provide a mechanism to constrain the trained network size; see constraint 4.2b. To compare with PruneTrain, we train ResNet-50 under varying pruning settings offered by PruneTrain. After training is completed for these networks, we train equivalent inference cost networks using BEP.

4.4.3 Training-Time Improvements and Discussion

Our approach delivers training time improvements in wall-clock time. In Table 4.6 we show the GPU, wall-clock, overhead, and model time for BEP and BEP-LITE on the ResNet-50 pruning tasks. GPU training time speedup is correlated with the size of the model after training completion. BEP-LITE delivers improved performance in wall-clock and GPU time. This improvement is delivered with no significant loss of performance after training when compared to BEP.

The measured wall-clock time is significantly higher than GPU time due to the disk read time overhead and image decoding overhead of training. The GPU time reduction is well correlated with the amount of pruning, with higher pruning yielding shorter GPU time. However, due to the constant training overhead, these

improvements do not perfectly translate to wall-clock time improvements. Despite this, BEP and BEP-LITE are able to deliver significant improvements in wall-clock training time compared to the unpruned baseline of 47h. In particular, with 86.7% pruned flops, BEP-LITE shows a 40% improvement in wall-clock time with only a 5.5% drop in accuracy.

The training overhead can be significantly reduced in many ways to deliver further wall-clock time improvements. Disk overhead can be reduced by utilizing faster disks, or disk arrays for higher throughput. Image decoding overhead can be alleviated by storing predecoded files in bitmap form. These approaches can further reduce the wall-clock time of the training process. Thus our approach delivers significant, practical improvements in GPU time reduction and wall-clock time reduction. The wall-clock time reduction can be further improved with minimal effort.

4.5 Conclusion

This paper presents a novel efficient algorithm to perform pruning of DNN elements such as neurons, or convolutional layers *during the training process*. To achieve pruning during training while preserving the performance of the DNN upon convergence, a Bayesian model (i.e., MOGP) is used to predict the future (unseen) saliency of DNN elements by leveraging the exponentially decaying behavior of the saliency and the correlations between saliency of different network elements. Then, we utilize several properties (Lemma 4.1 and Lemma 4.2) of the objective function and propose an efficient Bayesian early pruning algorithm. Empirical evaluations on

benchmark datasets show that our algorithm performs favorably to related works for pruning convolutional filters and neurons. In particular, BEP shows strong improvement when compared to related work with minimal cost overhead when a significant portion of the DNN is pruned. Moreover, the proposed BEP is robust to changes in hyperparameters (see Table 4.4), which demonstrates its applicability to "never-before-seen" network architectures and datasets without further hyperparameter tuning. Our approach also remains flexible to changes in saliency function, and appropriately *balances* the trade-off between training cost vs. test-time efficacy in DNN pruning.

We are able to scale up approaches rooted in probabilistic methods while keeping their robustness advantages. We have demonstrated the scalability of this approach by pruning larger DNNs and showing the superior performance of our approach when compared against competing approaches. We have demonstrated the robustness of our approach using an extensive ablation study. This work gives evidence for scaling up decision-making under uncertainty, and also shows how it can outperform deep learning approaches under certain scenarios.

4.5.1 Limitations

Our work has some limitations that may be addressed by future work. The chief limitation of our work is its computationally heavy cost. Although our approach comes with theoretical guarantees, this requires a pruning runtime measured in hours. In practice, usage of Gaussian Processes or theoretically principled pruning algorithms may not be necessary for strong empirical performance. In such scenarios,

using more performant deep learning models and heuristic based pruning algorithms that do not give theoretical guarantees may yield better empirical performance at a cheaper cost.

Another avenue of addressing the limitations of our work is to transform our approach into the deep learning paradigm. In the deep learning paradigm, cheaper approximations and heuristics are used in order to deliver similar or equivalent performance.

A final limitation of our work is our dependence on dynamic penalty scaling. Ideally, a more comprehensive and theoretically motivated solution is needed which accomplishes the same purpose as dynamic penalty scaling.

Chapter 5

Hessian-Aware Bayesian Optimization for Decision-Making Models

5.1 Introduction

Decision-Making Models choose sequences of actions to accomplish a goal. Multi-Agent Decision-Making Models choose actions for multiple agents working together towards a shared goal. Multi-Agent Reinforcement Learning (MARL) has emerged as a competitive approach for optimizing Decision-Making Models in the multi-agent setting. MARL optimizes a *policy* under the partially observable Markov Decision Process (POMDP) framework, where decision making happens in an *environment* determined by a set of possible states and actions, and the *reward* for an action is conditioned upon the partially observable state of the environment. A policy forms a set of decision-making rules capturing the most rewarding actions in a given state. MARL utilizes gradient-based methods requiring informative gradients to make progress. This approach benefits from dense reward, which allows reinforcement learning methods to infer a causal relationship between individual actions and their corresponding reward. This feedback may not be present in the scenario of sparse

reward (Pathak et al. 2017; Qian and Yu 2021). In addition, gradient-based methods are susceptible to falling into local maxima.

In contrast to optimization by MARL, Bayesian Optimization (BO) offers an alternative approach to policy optimization. Since BO is a gradient-free optimizer capable of searching globally, applying BO to multi-agent policy search (MAPS) both ensures global searching of the policy, and overcomes poor gradient behavior in the reward function (Qian and Yu 2021). The chief challenge in BO for MAPS is the high dimensionality of complex multi-agent interactions.

A significant degree of high-dimensional multi-agent interactions exist in MAPS. For example, considering an autonomous drone delivery system, several agents (i.e., drones) must work together to maximize the throughput of deliveries. In doing so, these agents may separate themselves into different roles, for example, long-distance or short-distance deliveries. The optimal policy for each role may be significantly different due to distances to recharging base stations (e.g., drones must conserve battery). In forming the optimal policy, the *interaction* between agents must be considered to both optimally divide the task between the drones, as well as coordinate actions between drones (e.g., collision avoidance). These interactions may change over time. For example, a drone must avoid collision with nearby drones, which changes as it moves through the environment. With many agents, these interactions become more complex.

However, we propose the usage of BO for MAPS on memory-constrained devices which necessitates very compact policies which enables the possibility of overcoming the above limitation. In the context of memory-constrained devices such as Internet

of Things (IoT) devices (Merenda, Porcaro, and Iero 2020), small policies must be used. Secondly, in environments with sparse reward feedback, training these networks with RL presents significant challenges due to unhelpful policy gradients. Finally, the possibility of *globally optimizing* a compact policy for memory-constrained systems is appealing due to its strong performance guarantees.

To achieve this, we use a higher-order model (HOM) which *generates* a policy model. This higher-order model encapsulates the abstractions, or the inductive biases into a set of algorithms. The HOM is divided into immutable instructions (i.e., algorithms) corresponding to the abstractions of the role and role interaction and mutable parameters that are used to generate (GEN) a policy model during evaluation. The generation process executes the algorithms in order to generate the model which can perform inference.

To optimize our proposed HOM, we specialize BO by exploiting task-specific structures. A promising avenue of High-dimensional Bayesian Optimization (HDBO) is through additive decomposition. Additive decomposition separates a high-dimensional optimization problem into several independent low-dimensional sub-problems (Duvenaud, Nickisch, and Rasmussen 2011; Kandasamy, Schneider, and Póczos 2015). These sub-problems are independently solved thus reducing the complexity of high dimensional optimization. However, a significant challenge in additive decomposition is *learning the independence structure* which is unknown a-priori. Learning the additive decomposition is accomplished using stochastic sampling such as Gibbs sampling (Kandasamy, Schneider, and Póczos 2015; Rolland et al. 2018; E. Han, Arora, and Scarlett 2021b) which is known to have poor performance in

high dimensions (Johnson, Saunderson, and Willsky 2013; Barbos et al. 2017).

In our work, we overcome this shortcoming by observing the GEN process of the HOM. In particular, we can measure a surrogate Hessian during the GEN process which significantly simplifies the task of learning the additive structure. This surrogate Hessian informs the dependency structure of the optimization problem due to the equivalence between a zero Hessian value, and independence between dimensions due to the linearity of addition. We term this approach Hessian-Aware GP-UCB (HA-GP-UCB) and visualize our approach in Fig. 5.2. Our proposed BO approach is also applicable to policy-search in the single-agent setting, showing its general-purpose applicability in Decision-Making Models. In this work, we make the following contributions:

- We propose a parameter-efficient HOM for MAPS which is both expressive and compact. Our approach is made feasible by using specific abstractions of *roles* and *role interactions*.
- We propose HA-GP-UCB, a variant of BO that simplifies the learning of dependency structure and provides strong regret guarantees which scale with $\mathcal{O}(\log(D))$ under reasonable assumptions, where D is the dimensionality of the optimization problem.
- We validate our approach on several multi-agent benchmarks and show our approach outperforms related works for compact models fit for memory-constrained scenarios. Our HA-GP-UCB also overcomes sparse reward behavior in the

Table 5.1: Summary of key notations.

Notation	Description
v	The objective function being optimized by Bayesian Optimization
Θ	The domain for the objective function v
θ_t	A point in the domain Θ that is picked at time t
μ_T^k	The posterior mean (inferred after observations up to time $T-1$) at time T using the kernel k
$[\sigma_T^k]^2$	The posterior variance at time T using the kernel k
$r(\theta_t)$	The difference between the maxima of the function v in domain Θ, $v(\theta^*)$, and $v(\theta_t)$
R_T	The cumulative regret, $\sum_{t=1}^{T} r(\theta_t)$
Θ^a	Dimension a of the domain D
$\mathcal{G}_d$	A graph showing the dependencies between dimensions where edges exist between two dimensions if they are dependent
V_d	In the graph indicated by $\mathcal{G}_d$ the set of dimensions corresponding to Θ
E_d	In the graph indicated by $\mathcal{G}_d$ the set of edges corresponding to the dependencies between Θ
$\Theta^{(i)}$	Collection of dimensions indicated by (i) corresponding to a maximal clique in the graph $\mathcal{G}_d$
$k^{\Theta^{(i)}}$	A Gaussian process kernel correspond to the maximal clique (i)
k	The Gaussian process kernel for inference corresponding to the sum of $k^{\Theta^{(i)}}$: $k \triangleq \sum_i k^{\Theta^{(i)}}$
$v^{(i)}$	Under the additive assumption, it is assumed that $v = \sum_i v^{(i)}$ where each $v^{(i)}$ is sampled from $k^{\Theta^{(i)}}$
$\mathcal{U}(\Theta)$	A uniform random distribution over the domain Θ
$H(\theta_{t,h})$	A query to the Hessian at $\theta_{t,h}$
$\widetilde{\mathcal{G}}_d$	The graph corresponding to the detected dependency structure by querying the Hessian
Max-Cliques($\widetilde{\mathcal{G}}_d$)	A function computing the maximal cliques in the graph $\widetilde{\mathcal{G}}_d$
$\mathbf{s}$	The set of states of the cooperative multi-agent system where $\mathbf{s} \triangleq [\mathbf{s}^i]_{i=1,\dots,n}$ and i denotes the index of the agent
$\mathbf{a}$	The set of actions taken by each agent where $\mathbf{a} \triangleq [\mathbf{a}^i]_{i=1,\dots,n}$ and i denotes the index of the agent
$\mathbf{s}^{\alpha(i)}$	The state for agent a taking on the role $\alpha(i)$
$\mathbf{a}^{\alpha(i)}$	The action taken by agent a taking on the role $\alpha(i)$
$\Lambda^{\theta_{r,i}}$	An affinity function for taking on role i where r denotes it belonging to the part of the HOM for role assignment
$\Lambda^{\theta_{g,v}}$	An affinity function determining whether an edge exists during the interaction of roles in the HOM policy
$M^{\theta_{g,\eta}}$	The message passing function parameterized by $\theta_{g,\eta}$ for the role interaction message passing neural network
$U^{\theta_{g,e}}$	The action update function parameterized by $\theta_{g,e}$ for the role interaction message passing neural network

reward function in multiple settings showing its effectiveness in Decision-Making Models both in the single-agent and multi-agent settings.

Table 5.1 provides a summary of notations that are used frequently in paper.

5.2 Design

We consider the problem of learning the joint policy of a set of n agents working cooperatively to solve a common task. During each interaction with the environment, each agent i is associated with a state $\mathbf{s}^i \in \mathcal{S}^i$ with the global state represented as $\mathbf{s} \triangleq [\mathbf{s}^i]_{i=1,\dots,n}$. Each agent i cooperatively chooses an action $\mathbf{a}^i \in \mathcal{A}^i$ with the global action represented by $\mathbf{a} \triangleq [\mathbf{a}^i]_{i=1,\dots,n}$. Each state, action pair is associated with a *reward* function: $\rho(\mathbf{s}, \mathbf{a})$. In order to achieve the common task, a policy

parameterized by θ: $\pi^\theta \triangleq \mathcal{S} \rightarrow \mathcal{A}$ governs the action taken by the agents, after observing state $\mathbf{s} \in \mathcal{S}$. The goal of RL is to learn the optimal policy parameters that maximizes the accumulation of rewards during a predefined number of interactions with the environment.[1] The accumulation of rewards, $v(\theta) \triangleq \sum \rho(\mathbf{s}, \mathbf{a})$, is termed the value function which we optimize. In contrast to RL, which receives feedback on the reward of an action with every interaction, we treat $v(\theta)$ as an opaque function measuring the *value* of a policy. We utilize BO to optimize θ using solely the accumulated reward, $v(\theta)$, as feedback from the environment.

5.2.1 Architectural Design

To achieve a compact and tractable policy space, we consider policies under the useful abstractions of *role* and *role interaction*. These abstractions have consistently shown strong performance in multi-agent tasks. Therefore we can simplify the policy space by limiting it to only policies using these abstractions.

As role and role interaction are immutable abstractions within our policy space, we express them as *static algorithms* which are not searched over during policy optimization. These algorithms take as input parameters which are mutable and searched over during policy optimization. This combination of immutable instructions, and mutable parameters reduces the size of the search space,[2] yet is still able to express policies which conform to the role and role interaction abstractions.

[1] Further RL overview can be found in Arulkumaran et al. 2017.

[2] This approach to efficiency is similar in spirit to the work of Y. Lee et al. 1986.

We term this approach a *Higher-Order Model* (HOM) which generates (GEN) the model using instructions and parameters into a policy model during evaluation. This HOM is separated into role assignment, and role interaction stages. We visualize an overview of this approach in Fig. 5.1, left. These parameters are interpreted in context of the current state by the instructions (Alg. 1, Alg. 2) of the HOM to form the policy model which dictates the resultant action.

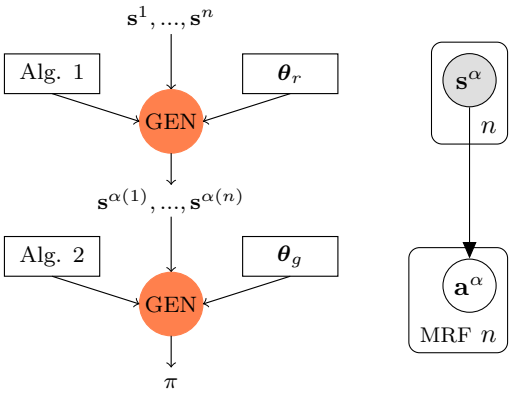

Figure 5.1: Left: HOM architecture. "Alg. 1/Alg. 2" are Algorithms 1 and 2 respectively. GEN uses θ_r and θ_g during evaluation to yield a model which represents the policy. θ_r and θ_g are optimized by BO. Right: Inferring $\mathbf{a}^\alpha$ given $\mathbf{s}^\alpha$.

5.2.2 Role Assignment

Following the popularity of role based collaboration in multi-agent systems, we assume the interaction and decision-making of each agent is governed by its assigned role. For example, in drone delivery, roles could be short-distance deliveries, and long-distance deliveries. In filling these roles, the state of each of the agents are considered. E.g., a drone with low battery may be limited to only performing short-distance deliveries. A straightforward approach to implement role based interaction is to permute agents into an equivalent number of roles.[3] This approach allows the

[3]This is a common assumption in multi-agent systems, see, e.g., Hoang Minh Le et al. 2017.

HOM to focus on optimizing role behavior, along with a function to permute agents into roles. The success of role based collaboration has been enabled by abstracting roles from agents and noting that the complexity of learned behavior is limited to the number of roles. This improves upon the complexity of learning without role based collaboration, in which case each role based behavior must be learned for each agent. We assume that an optimal policy can be decomposed as follows:

$$\pi(\mathbf{a}^1, \ldots, \mathbf{a}^n \mid \mathbf{s}^1, \ldots, \mathbf{s}^n) \triangleq \pi_r(\mathbf{a}^{\alpha(1)}, \ldots, \mathbf{a}^{\alpha(n)} \mid \mathbf{s}^{\alpha(1)}, \ldots \mathbf{s}^{\alpha(n)}) \tag{5.1}$$

where α is a permutation function dependent on the state, $\mathbf{s}^1, \ldots, \mathbf{s}^n$. The above assumption requires a permutation of agents into roles. For example, in drone delivery, roles could be short-distance deliveries, and long-distance deliveries. In filling these roles, the state of each of the agents are considered. E.g., a drone with low battery may be limited to only performing short-distance deliveries.

To capture this behavior, we define a per role affinity function: $\Lambda^{\theta_{r,i}}(\cdot)$ which is the affinity to take on role i and is parameterized by $\theta_{r,i}$. This function evaluates the affinity of agent ℓ taking on role i using the state of agent: $\mathbf{s}^\ell$. The optimal permutation maximizes the total affinity of an assignment:$\sum_{i=1}^{n} \Lambda^{\theta_{r,i}}(\mathbf{s}^{\alpha(i)})$ where α represents a permutation. This problem can be efficiently solved using the Hungarian algorithm. We integrate the Hungarian algorithm in our HOM approach during the GEN process. We formalize this in Algorithm 1 which forms the instructions in the role assignment HOM.

Algorithm 1 *RoleAssignment*

Require: $\mathbf{s}^1, \ldots, \mathbf{s}^n$

1: **return** $\arg\max_\alpha \sum_{i=1}^{n} \Lambda^{\theta_{r,i}}(\mathbf{s}^{\alpha(i)})$

Given Algorithm 1, during GEN process, the agents' state, $\mathbf{s}^1, \ldots, \mathbf{s}^n$ is contextually interpreted to yield a permutation model: α. First the role affinity is calculated using the $\Lambda^{\theta_{r,i}}$ function, following which, the total role affinity is optimized using the Hungarian Algorithm. Going forward, we consider the problem of determining the joint policy $\pi_r(\mathbf{a}^{\alpha(1)}, \ldots, \mathbf{a}^{\alpha(n)} \mid \mathbf{s}^{\alpha(1)}, \ldots \mathbf{s}^{\alpha(n)})$ which enables collaborative interactions.

5.2.3 Role Interaction

Capturing multiple roles working together is an important part of an effective multi-agent policy. For example in drone delivery, drones must both divide the available task among themselves, as well as use collision avoidance while executing deliveries. Modeling role interactions must accomplish two goals. Firstly, agent interactions may change over time. For example collision avoidance strategies involve the closest drones which change as the drone moves within the environment. Secondly, efficient parameterization is needed as the number of interactions can scale exponentially due to considering interactions between many agents.

To overcome these challenges, we propose a HOM which generates (GEN) a graphical model. The usage of a graphical model decomposes the exponentially scaling interaction problem into a pairwise interaction model, along with a message passing approach to facilitate complex interactions between many agents. The GEN process is conditioned on the agents' state, thus capturing dynamic role interactions; in addition the GEN process allows for a more compact policy space with far fewer parameters. The resultant generated graphical model captures the state-dependent

interaction between roles and yields the resultant actions for each role. After GEN, the interaction between roles are captured by the resultant conditional random field. This is presented in Fig. 5.1, right. The MRF (Markov Random Field) represents arbitrary undirected connectivity between nodes $\mathbf{a}^{\alpha(1)}, \ldots, \mathbf{a}^{\alpha(n)}$, which is denoted by $\mathcal{G}$. This connectivity allows different roles to collaborate together to determine the joint action.[4]

To generate graphical models of the above form, our HOM uses edge affinity functions, $\Lambda^{\theta_{g,v}}(\cdot)$, which enables dynamic arbitrary connectivity between roles. For all pairs of roles with state, $\mathbf{s}^{\alpha(i)}, \mathbf{s}^{\alpha(\ell)}$ an edge is generated if the affinity between these two states is sufficiently high (i.e., > 0). This dynamic edge generation approach overcomes the quadratic parameter scaling if all pairs of agents were separately modelled. The graphical model GEN process is presented in Algorithm 2 which yields a graphical model.

To cooperatively determine a set of actions for roles given the graphical model, we perform inference over the graphical model presented in Fig. 5.1 using Message Passing Neural Networks (Gilmer et al. 2017) (MPNN). We present iterative message passing rules to map from $\mathbf{s}^{\alpha}$ to a^{α}:

$$m_{t+1}^{\alpha(i)} \triangleq \sum_{\alpha(\ell) \in N^{\alpha(i)}} M^{\theta_{g,\eta}}\left(h_t^{\alpha(i)}, h_t^{\alpha(\ell)}, i, \ell\right) h_{t+1}^{\alpha(i)} \triangleq U^{\theta_{g,e}}\left(\mathbf{s}^{\alpha(\mathbf{i})}, h_t^{\alpha(i)}, m_{t+1}^{\alpha(i)}\right); \mathbf{a}^{\alpha} \triangleq \left[h_{\tau}^{\alpha(i)}\right]_{i=1,\ldots,n} \tag{5.2}$$

where M is the message function parameterized by $\theta_{g,\eta}$ which enables interaction between connected nodes, U is the action update function parameterized by $\theta_{g,e}$

[4]We refer readers to Chaohui Wang, Komodakis, and Paragios 2013 for additional overview.

which updates the node's internal hidden state conditioned on the messages received, and $N^{\alpha(i)}$ denotes the neighbors of $\alpha(i)$. The message passing procedure allows for cooperative determination of all roles' actions using pairwise message passing. Roles which are not immediate neighbors of each other influence each other's behavior through intermediary connecting nodes. The message passing procedure concludes after τ iterations of message passing with the policy actions indicated by the hidden states, $\left[h_{\tau}^{\alpha(i)}\right]_{i=1,\ldots,n}$.

To generate graphical models of the above form, our HOM uses edge affinity functions. This approach overcomes the quadratic scaling in modeling all pairs of interaction. Edge affinity functions $\Lambda^{\theta_{g,v}}(\cdot)$ determine whether an edge exists between node $\mathbf{a}^{\alpha(i)}$, and $\mathbf{a}^{\alpha(\ell)}$. The graphical model GEN process is presented in Algorithm 2.

Algorithm 2 *RoleInteraction*

Require: $\mathbf{s}^{\alpha(1)}, \ldots, \mathbf{s}^{\alpha(n)}$
for $i \leftarrow 1, \ldots, n$ **do**
  for $\ell \leftarrow 1, \ldots, n$ **do** ▷ Edge affinities.
    if $\Lambda^{\theta_{g,v}}(\mathbf{s}^{\alpha(i)}, \mathbf{s}^{\alpha(\ell)}) > 0$ **then**
      $N^{\alpha(i)}.append(\alpha(\ell))$
return $N^{\alpha(1)}, \ldots, N^{\alpha(n)}$

In Algorithm 2, the affinities between different roles is computed using the $\Lambda^{\theta_{g,v}}$ function, following which, edges are created between roles which have sufficiently high affinity.

Finally, Algorithm 3 drives the GEN process. The GEN process consists of permuting agents into roles, creating the graphical model to enable interactions between agents taking on their respective roles, and finally performing inference

over the graphical model using a MPNN.

Algorithm 3 GEN-*Policy*

Require: $\mathbf{s}^1, \ldots, \mathbf{s}^n$
1: $\alpha \leftarrow \mathit{RoleAssignment}(\mathbf{s}^1, \ldots, \mathbf{s}^n)$
2: $N \leftarrow \mathit{RoleInteraction}(\mathbf{s}^{\alpha(1)}, \ldots, \mathbf{s}^{\alpha(n)})$
3: $\mathbf{a} \leftarrow \text{MPNN}(\mathbf{s}^\alpha, N)$ ▷ See Eq. 5.2
4: **return** $[a^{\alpha^{-1}(i)}]_{i=1,\ldots,n}$

5.2.4 Additive Decomposition

Although our HOM policy representation is compact, it is still of significant dimensionality which makes optimization with BO difficult. HDBO is challenging due to the curse of dimensionality with common kernels such as Matern or RBF.[5] This curse of dimensionality stems directly from the difficulty of finding the global optima of a high-dimensional function (e.g., a value function $v(\theta)$ determining the value of a policy in some unknown environment). A common technique to overcome this is through assuming additive structural decomposition on v: $v(\theta) \triangleq \sum_{i=1}^{M} v^{(i)}(\theta^{(i)})$ where $v^{(i)}$ are independent functions, and $\theta^{(i)} \in \Theta^{(i)}$ (Duvenaud, Nickisch, and Rasmussen 2011). The additive decomposition simplifies a high-dimensional optimization problem since the optima of a function constructed through addition of subfunctions can be found by independently optimizing each subfunction as visualized in Fig. 5.2. In the context of BO, additive decomposition significantly simplifies the optimization problem due to the properties of Multivariate Gaussian variables.

[5] A parallel area in HDBO is of computational efficiency of acquisition which is outside the scope of this work. We refer readers to the works of Mutny and Krause 2018, J. T. Wilson et al. 2020, and Ament and Gomes 2022b.

In additive decomposition we denote the domain of the optimization problem, $\Theta \triangleq \Theta^1 \times \ldots \times \Theta^D$ for some dimensionality D, that is the domain is constructed through the Cartesian product of each of its dimensions. Each subfunction to optimize, $v^{(i)}$, corresponds to a subdomain restricted to some subset of these dimensions, $\Theta^{(i)} \subseteq \{\Theta^1, \ldots, \Theta^D\}$. Typically, it is assumed that each $\Theta^{(i)}$ is of low dimensionality (i.e., $v^{(i)}$ is defined on only a few dimensions for each i). This structural assumption is combined with the assumption that each $v^{(i)}$ is sampled from a GP. Due to the properties of Multivariate Gaussians, if $v^{(i)} \sim \mathrm{GP}\Big(0, k^{\Theta^{(i)}}(\theta^{(i)}, \theta^{(i)'})\Big)$ then $v \sim \mathrm{GP}\Big(0, \sum_i k^{\Theta^{(i)}}(\theta^{(i)}, \theta^{(i)'})\Big)$ (Rasmussen and C. K. I. Williams 2006), which follows from the addition of two Gaussian random variables is also a Gaussian random variable. This assumption decomposes a high dimensional GP surrogate model of v into a set of many low dimensional GPs, which is easier to jointly learn and optimize.

To contextualize an additive decomposition, we represent the decomposition by a dependency graph between the dimensions: $\mathcal{G}_d \triangleq (V_d, E_d)$ where $V_d \triangleq \{\Theta^1, \ldots, \Theta^D\}$ and $E_d \triangleq \{(\Theta^a, \Theta^b) \mid a, b \in \Theta^{(i)} \text{ for some } i\}$. A simple decomposition of an additive function and its associated dependency graph is visualized in Fig. 5.2. *We **highlight** that this graph is between the dimensions of the policy parameters, Θ, and is unrelated to the graphical model of role interactions presented in earlier sections.* It is possible to accurately model v by a kernel $k \triangleq \sum_i k^{\Theta^{(i)}}$ where each $\Theta^{(i)}$ corresponds to a *maximal clique* of the dependency graph (Rolland et al. 2018). Knowing the dependency graph greatly simplifies the complexity of optimizing v.

However, learning the dependency graph in additive decomposition remains challenging as there are $O(D^2)$ possible edges each of which may be present or

absent yielding $2^{O(D^2)}$ possible dependency structures. This difficult problem is often approached using inefficient stochastic sampling methods such as Gibbs sampling.

5.2.5 Hessian-Aware Bayesian Optimization

The assumption of additive decomposability in the above proposition is a common assumption within the Bayesian Optimization community (Rolland et al. 2018). We propose learning the dependency structure during the GEN process. Our approach is based on the following observation which is illustrated in Fig. 5.2:

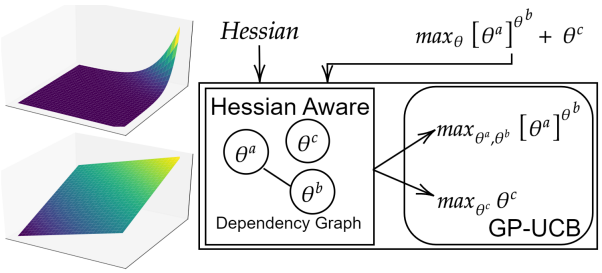

Figure 5.2: Left, above, plot of $f(x, y) = x^y$; below, plot of $f(x, y) = x + y$. The curvature of *additively constructed functions* is zero; *non-zero curvature* indicates dependency among input variables. Right, examining the Hessian learns the dependency structure which decomposes complex problems into simpler problems solved by GP-UCB.

Proposition 5.1. *Let* $\mathcal{G}_d = (V_d, E_d)$ *represent an additive dependency structure with respect to* $v(\theta)$*, then the following holds true:* $\forall a, b \; \frac{\partial^2 v}{\partial \theta^a \partial \theta^b} \neq 0 \implies (\Theta^a, \Theta^b) \in E_d$ *which is a consequence of* v *formed through addition of independent sub-functions* $v^{(i)}$*, at least one of which must contain* θ^a, θ^b *as parameters for* $\frac{\partial^2 v}{\partial \theta^a \partial \theta^b} \neq 0$ *which implies their connectivity within* E_d*.*

In practice, observing the Hessian of the value function, $\mathbf{H}_v$, is not possible due to v being an opaque function. However, during the GEN process we can observe

the Hessian of the policy, $\mathbf{H}_\pi$. This surrogate Hessian is closely related to the $\mathbf{H}_v$ as $v(\theta)$ is determined through interaction of the policy with an unknown environment. Because the *value* of a policy is a function of the policy; it follows by the chain rule that $\mathbf{H}_\pi$ is an important sub-component of $\mathbf{H}_v$. We utilize the surrogate Hessian in our work and demonstrate its strong empirical performance in validation. Following this reasoning, we consider algorithms with noisy query access to the Hessian, $\mathbf{H}_v$. Note that we assume that the surrogate Hessian, $\mathbf{H}_\pi$, can well serve as a noisy surrogate for the true Hessian, $\mathbf{H}_v$.[6]

Assumption 5.1. *Let $\mathcal{G}_d = (V_d, E_d)$ be sampled from an Erdős-Rényi model with probability $p_g < 1$: $\mathcal{G}_d \sim G(D, p_g)$. That is, each edge (Θ^a, Θ^b) is i.i.d. sampled from a binomial distribution with probability, p_g. With $[\Theta^{(i)}]_{i=1,\dots,M}$ representing the maximal cliques of $\mathcal{G}_d$, we assume that $v \sim GP\left(0, \sum_i k^{\Theta^{(i)}}(\theta^{(i)}, \theta^{(i)'})\right)$ for some kernel k taking an arbitrary number of arguments (e.g., RBF). Noisy queries can be made to the Hessian of v, $\mathbf{H}_v$. We define $H(\theta) \triangleq [\frac{\partial^2 v}{\partial \theta^a \partial \theta^b} + \epsilon_h^{(a,b)}]_{a,b=1,\dots,D}$ where $\epsilon_h^{(a,b)} \sim \mathcal{N}(0, \sigma_n^2)$ i.i.d. Each query to H has corresponding regret of $r(\theta)$.*

In the above assumption, a new setting is assuming that $\mathcal{G}_d = (V_d, E_d)$ be sampled from an Erdős-Rényi model. This assumption derives from viewing the dependency graph as a *social network* yielding a community structure. In this interpretation, the nodes of the network represent dimensions, and edges represent dependencies between dimensions. Interpreting graphical structures as social networks is a common practice in graph theory (Bollobás and Erdös 1976; Seshadhri, Kolda, and Pinar

[6]We revisit the validity of this assumption in Appendix B.7.

2012; Hamlili 2017; Zhao, H. Wu, and X. Zhang 2021). More recent works in the study of social networks utilize more realistic assumptions on the nature of the community structures (S. Fortunato and Castellano 2009). These more realistic assumptions are valuable for future work. Nonetheless, we confirm that the assumed sparse connectivity structure is empirically supported in the policy optimization setting in Fig. 5.3.

Under this assumption, we show that it's possible to learn the underlying dependency structure of $\mathcal{G}_d = (V_d, E_d)$ with a polynomial number of queries to the noisy Hessian. We present HA-GP-UCB in Algorithm 4 and prove theoretical results regarding its performance. In the first stage of HA-GP-UCB, we perform C_1 queries to the Hessian if $t \leq T_0$. These Hessian queries are then averaged and compared to a cutoff constant c_h to determine the dependency structure $\widetilde{E}_d$. We show that after $C_1 T_0$ queries to the Hessian, with high probability we have $\widetilde{E}_d = E_d$, where E_d is the unknown ground truth dependency structure for v. This argument is formalized in the following theorem.

Algorithm 4 HA-GP-UCB

Require: v, H, k

for $t \leftarrow 1, \ldots, T_0$ **do** ▷ Sample Hessian $T_0 \times C_1$ times for dependencies.
  $\theta_{t,h} \sim \mathcal{U}(\Theta)$ ▷ Randomly sample over the domain.
  for $\ell \leftarrow 1, \ldots, C_1$ **do** $h_{t,\ell} \leftarrow H(\theta_{t,h})$
$\widetilde{E}_d \leftarrow |\sum h| > c_h$ ▷ Discriminate dependencies
$\widetilde{\mathcal{G}}_d \leftarrow (\{\Theta^1, \ldots, \Theta^D\}, \widetilde{E}_d)$
$[\Theta^{(i)}]_{i=1,\ldots,M} \leftarrow$ Max-Cliques$(\widetilde{\mathcal{G}}_d)$ ▷ Compute Max-Cliques
$k \leftarrow \sum_{i=1}^{M} k^{\Theta^{(i)}}$
for $t \leftarrow T_0, \ldots, T$ **do** ▷ Run GP-UCB with dependency structure
  $\theta_t \leftarrow \arg\max_\theta \mu_{t-1}^k(\theta) + \sqrt{\beta_t}\sigma_{t-1}^k(\theta)$ ▷ Max-Cliques additive kernel
  Query θ_t to observe $y_t = v(\theta_t) + \mathcal{N}(0, \epsilon^2)$
  Update posterior, μ, σ, with θ_t, y_t

Theorem 5.1. *Suppose*[7] *there exists* σ_h^2, p_h *s.t.* $\forall i,j\ \mathbb{P}_{\theta\sim\mathcal{U}(\Theta)}\left[k^{\partial i\partial j}(\theta,\theta)\geq\sigma_h^2\right]\geq p_h$ *and* $\forall i,j,\theta,\theta'\ k^{\partial i\partial j}(\theta,\theta')\geq 0$. *Then for any* $\delta_1,\delta_2\in(0,1)$ *after* $t\geq T_0$ *steps of HA-GP-UCB we have:* $\bigcap_{i,j}P(\widetilde{E}_d^{i,j}=E_d^{i,j})\geq 1-\delta_1-\delta_2$ *when* $T_0=C_1>\frac{16D^2}{p_h\delta_1^2}\log\frac{2D^2}{\delta_1}\frac{\sigma_n^2}{\sigma_h^2}+\frac{D^2}{2\delta_2}$, $c_h\triangleq T_0\sigma_n\sqrt{2\log\frac{2D^2}{\delta_1}}$.

Our Theorem 5.1 relies on repeatedly sampling the Hessian to determine whether an edge exists between Θ^a, and Θ^b in the sampled additive decomposition. The key challenge is determining this connectivity under a very noisy setting, and for extremely low values of $\sigma_h^2\ll\sigma_n^2$ where the Hessian is zero with high probability. We are able to overcome this challenge using a Bienaymé's identity, a key tool in our analysis. We defer all proofs to the Appendix.

In the second stage of HA-GP-UCB, we extract the maximal cliques depending on $\widetilde{E}_d$ and construct the GP kernel, $k=\sum_i k^{\Theta^{(i)}}$, the sum of the aforementioned kernels and inference and acquisition proceeds same as GP-UCB.

To bound the cumulative regret, $R_t\triangleq\sum_{t=1}^{T_0}C_1 r(\theta_{t,h})+\sum_{t=T_0}^{T}r(\theta_t)$, we follow the following process. First, we bound the number and size of cliques of graphs sampled from the Erdős-Rényi model with high probability. Second, we bound the *mutual information* of an additive decomposition given the mutual information of its constituent kernels using Weyl's inequality. Third, we use similar analysis as Srinivas et al. 2010 to complete the regret bound.

Theorem 5.2. *Let* k *be the kernel as in Assumption 5.1, and Theorem 5.1. Let* $\gamma_T^k(d):\mathbb{N}\to\mathbb{R}$ *be a monotonically increasing upper bound function on the* mutual

[7]RBF kernel satisfies these assumptions when $\Theta=[0,1]^D$.

information *of kernel k taking d arguments. The cumulative regret of HA-GP-UCB is bounded with high probability as follows:*

$$R_T = \widetilde{\mathcal{O}}\left(\sqrt{T\beta_T D^{\log D+5}\gamma_T^k(4\log D + c_\gamma)}\right) \tag{5.3}$$

where c_γ is an appropriately picked constant and the base of the logarithm is $\frac{1}{p_g}$.

Whereas for typical kernels such as Matern and RBF, known regret bounds of GP-UCB (with an unknown decomposition) scale exponentially with D, our regret bounds scale with exponent $\mathcal{O}(\log D)$. In the above, $\mathcal{O}(\log D)$ represents the maximal clique size, which is logarithmic with respect to the total number of vertices (dimensions). This improved regret bound shows our approach is a theoretically grounded approach to HDBO.

5.3 Validation

We compare our work against recent algorithms in MARL on several multi-agent coordination tasks and RL algorithms for policy search in novel settings. We also perform ablation and investigation of our proposed HOM at learning roles and multi-agent interactions. We defer experimental details to Appendix B.1.

All presented figures are average of 5 *runs with shading representing* $\pm$ *Standard Error, the y-axis represents cumulative reward, the x-axis displayed above represents interactions with the environment in RL, x-axis displayed below represents iterations of BO. Commensurate with our focus on memory-constrained devices, all policy models consist of* < 500 *parameters.*

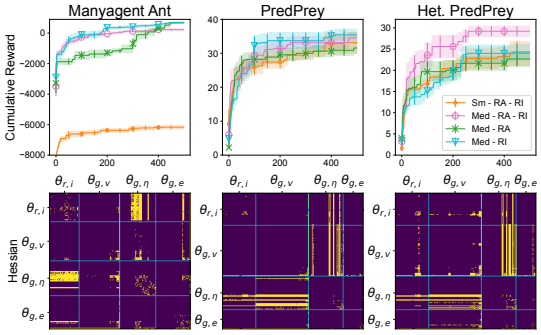

Figure 5.3: Ablation study. Training curves of our HOM and its ablated variants on different multi-agent environments.

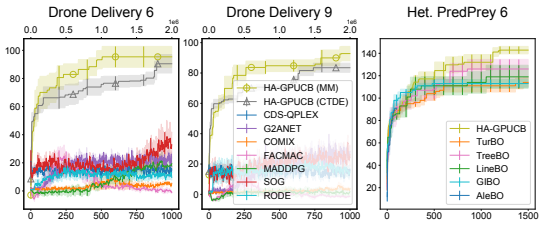

Figure 5.4: Left: Sparse reward drone delivery task. Right: Comparison with HDBO approaches.

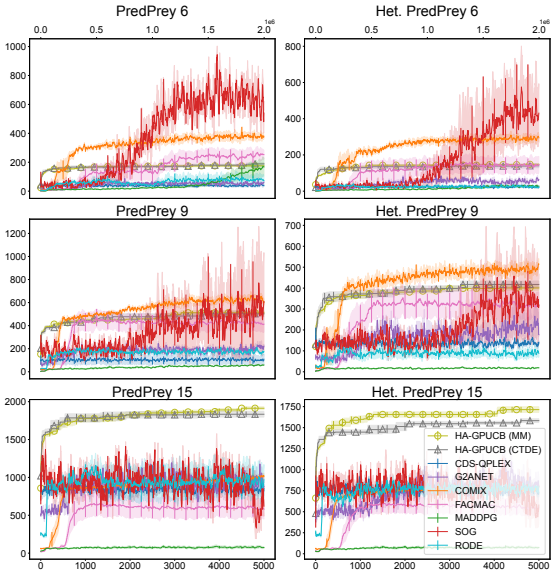

Figure 5.5: Scaling analysis. Training curves of HA-GP-UCB and competitors with increasing number of agents. The left column shows PredPrey with 6, 9, and 15 agents. The right column shows Het, PredPrey with 6, 9, and 15 agents.

5.3.1 Ablation of the Higher-Order Model

We investigate the impact of Role Assignment (RA) and Role Interaction (RI) as well as model capacity on training progress. We conduct ablation experiments on Multiagent Ant with 6 agents, PredPrey with 3 agents, and Heterogenous PredPrey with 3 agents. Multiagent Ant is a MuJoCo locomotion task where each agent controls an individual appendage. PredPrey is a task where predators must work together to catch faster, more agile prey. Het. PredPrey is similar, except the predators have different capabilities of speed and acceleration. In ablation experiments, our default configuration is *Med - RA - RI* which employs components of RA and RI parameterized by neural networks with three layers and four neurons on each layer.

We present our ablation in Fig. 5.3.

For a simpler coordination task such as Multiagent Ant, we observe limited improvement through RA or RI. In contrast, RI shows strong improvement in PredPrey and Het. PredPrey. It is because, in PredPrey, predators must work together to catch the faster prey. Since the agents in PredPrey are *homogeneous*, *ablating RA* makes the optimization simpler and more compact without losing expressiveness. Thus, ablating RA leads to a performance increase. In Het. PredPrey, the predator agents have heterogeneous capabilities in speed and acceleration. Thus, RA plays a critical role in delivering strong performance. We also show that overly shrinking the model size (*Sm - RA - RI*) can hurt performance as the policy model is no longer sufficiently expressive. This is evidenced in the Multiagent Ant task. We observed that using neural networks of three layers with four neurons each to be sufficiently balanced across a wide variety of tasks.

In Fig. 5.3, we present the detected Hessian structure by HA-GP-UCB in the respective tasks. The detected Hessian structures generally show strong block-diagonal associativity in the HOM parameters, i.e., $[\theta_{r,i}, \theta_{g,v}, \theta_{g,\eta}, \theta_{g,e}]$. This shows that our approach can detect the interdependence *within* the sub-parameters, but relative independence between the sub-parameters. We observe more off-diagonal connectivity in the complex coordination tasks of PredPrey and Het. PredPrey. The visualization of Hessian structure on PredPrey shows that our approach can detect the importance of *jointly optimizing* role assignment and interaction to deliver a strong policy in this complex coordination task. We investigate the learning behavior of the HOM further in Appendix B.2.

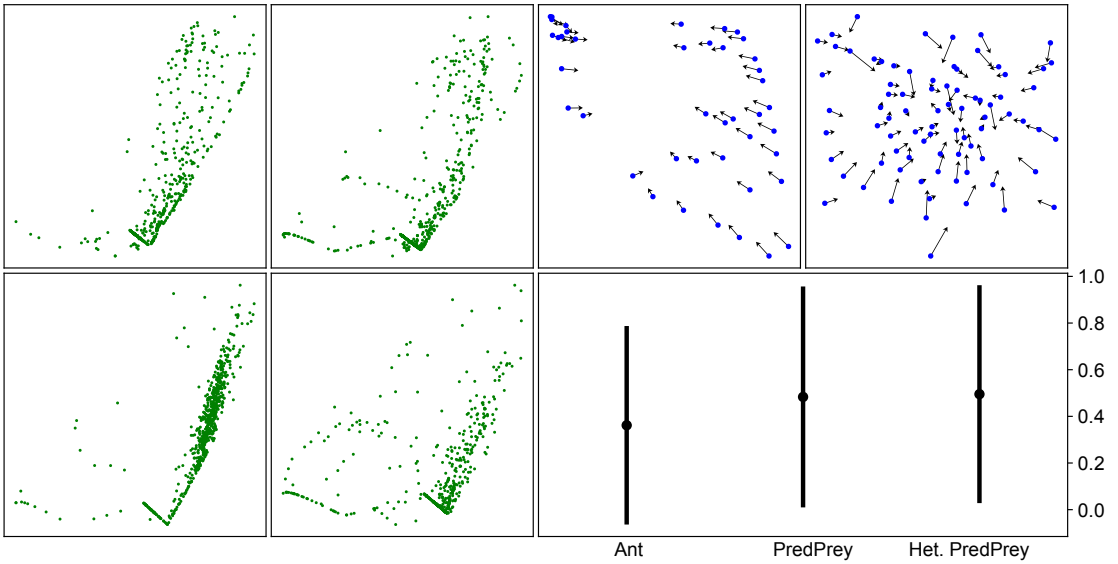

Figure 5.6: Left: Action distributions of different roles showing diversity in the Multiagent Ant environment with 6 agents. Right above: Policy modulation with role interaction in PredPrey and Het. PredPrey environment with 3 agents. Arrows represent change after message passing. Right below: Mean connectivity and standard deviation in role interaction in Multiagent Ant with 6 agents, PredPrey with 3 agents, and Het. PredPrey with 3 agents.

5.3.2 Comparison with MARL

We compare our method with competing MARL algorithms on several multi-agent tasks where the number of agents is increased. We validate both the HOM with HA-GP-UCB (HA-GP-UCB (MM)) and neural network policies trained in the CTDE paradigm (HA-GP-UCB (CTDE)). We observe that on complex coordination tasks such as PredPrey and Het. PredPrey our approach delivers more performant policies when coordination is required between *a large number of agents*. This is presented in Fig. 5.5. Although SOG (Shao et al. 2022), a Comm-MARL approach shows compelling performance with a small number of agents, with 15 agents, both HA-GP-UCB (CTDE) and HA-GP-UCB (MM) outperform this strategy. We highlight that HA-GP-UCB (CTDE) outperforms Comm-MARL approaches

Table 5.2: HA-GP-UCB typically outperforms RL with higher sparsity (e.g., Sparse-100, or Sparse-200).

	Ant-v3					Hopper-v3					Swimmer-v3					Walker2d-v3				
	DDPG	PPO	SAC	TD3	Intrinsic	DDPG	PPO	SAC	TD3	Intrinsic	DDPG	PPO	SAC	TD3	Intrinsic	DDPG	PPO	SAC	TD3	Intrinsic
Baseline	−90.77	1105.69	2045.24	2606.17	2144.00	604.20	1760.65	2775.66	1895.76	1734.00	44.45	121.38	58.73	48.78	1950.00	2203.80	892.81	4297.03	1664.46	2210.00
Sparse 2	−32.88	1007.80	2563.97	1407.40	1964.00	877.93	1567.14	3380.60	1570.84	2074.00	35.59	99.50	46.75	47.23	1758.80	1470.62	1471.33	1673.46	2297.43	1952.00
Sparse 5	−2687.97	961.31	711.56	762.61	1916.00	814.59	1616.79	3239.20	2290.67	1972.00	26.66	68.69	43.84	40.12	1856.00	961.30	697.93	1697.25	2932.27	1924.00
Sparse 20	−2809.89	624.07	694.30	379.12	1838.00	783.95	1629.28	2535.17	1436.33	1537.20	19.12	54.63	37.78	37.03	2108.00	663.04	365.39	1010.63	276.56	1810.00
Sparse 50	−3067.37	−67.43	663.28	253.66	1091.20	816.25	1010.73	1238.03	551.43	642.00	23.73	51.52	38.78	30.01	812.00	572.12	428.29	349.47	298.28	834.75
Sparse 100	−3323.43	−4021.56	679.30	−115.43	450.40	988.36	324.51	260.52	342.48	406.80	9.64	21.09	27.98	30.10	376.60	523.89	205.93	200.16	147.22	480.60
Sparse 200	−3098.37	−8167.98	−107.14	−147.86	258.60	765.05	222.76	300.36	281.68	350.80	−9.97	21.69	33.35	30.48	342.80	182.84	193.43	187.16	148.06	353.20
HA-GP-UCB	1147.21					1009.3					175.73					1008.90				

without communication during execution. We also note that HA-GP-UCB (MM) outperforms HA-GP-UCB (CTDE) showing the value of our HOM approach in complex coordination tasks. We defer further experimental results in this setting to Appendix B.2.

5.3.3 Higher-Order Model Investigation

We examined policy for Multiagent Ant with 6 agents for the role based policy specialization. The policy modulation plots were generated by examining the PredPrey and Het. PredPrey environments respectively.

In Fig. 5.6 we investigate the learned HOM policies. Our investigation shows that *role* is used to specialize agent policies while maintaining a common theme. *Role interaction* modulates the policy through graphical model inferences. Finally, role interactions are sparse, however noticeably higher for complex coordination tasks such as PredPrey.

5.3.4 Policy Optimization under Malformed Reward

We compare against several competing RL and MARL algorithms under malformed reward scenarios. We train neural network policies with HA-GP-UCB and competing algorithms. We consider a sparse reward scenario where reward feedback

is given every S environment interactions for varying S. Table 5.2 shows that the performance of competing algorithms is severely degraded with sparse reward and HA-GP-UCB outperforms competing approaches on most tasks with moderate or higher sparsity. Although intrinsic motivation (S. Singh, Barto, and Chentanez 2004; Zheng, Oh, and S. Singh 2018) has shown evidence in overcoming this limitation, we find that our approach outperforms competing approaches supported by intrinsic motivations at higher sparsity. This improvement is important as sparse and malformed reward structure scenarios can occur in real-world tasks (Aubret, Matignon, and Hassas 2019). We repeat this validation in Appendix B.2 with MARL algorithms in multi-agent settings and consider a delayed feedback setting with similar results.

5.3.5 Comparison with HDBO Algorithms

We compare with several related work in HDBO. This is presented in Fig. 5.4. We compare against these algorithms at optimizing our HOM policy. For more complex tasks that require role based interaction and coordination, our approach outperforms related work. TreeBO (E. Han, Arora, and Scarlett 2021a) is also an additive decomposition approach to HDBO, but uses Gibbs sampling to learn the dependency structure. However, our approach of learning the structure through *Hessian-Awareness* outperforms this approach. Additional experimental results are deferred to Appendix B.2.

5.3.6 Drone Delivery Task

We design a drone delivery task that is well aligned with our motivation of considering policy search in *memory-constrained devices* on tasks with *unhelpful or noisy gradient*

information. In this task, drones must maximize the throughput of deliveries while avoiding collisions and conserving fuel. This task is challenging as a positive reward through completing deliveries is rarely encountered (i.e., sparse rewards). However, agents often receive negative rewards due to collisions or running out of fuel. Thus, gradient-based approaches can easily fall into local minima and fail to find policies that complete deliveries.[8] We compare HA-GP-UCB against competing approaches in Fig. 5.4. We observe that MARL based approaches fail to find a meaningfully rewarding policy in this setting, whereas our approach shows strong and compelling performance. Furthermore, HA-GP-UCB (MM) outperforms HA-GP-UCB (CTDE) through leveraging roles and role interactions.

5.3.7 On the Robustness of HA-GP-UCB

HA-GP-UCB contains few hyperparameters. These hyperparameters were found using simple grid search (i.e., searching over the valid hyperparameter values in a simple manner). These hyperparameters were held constant throughout our extensive validation. The depth and the diversity of our validation shows that HA-GP-UCB is robust to new "never-before-seen" environments. We detail the hyperparameters of HA-GP-UCB.

- Size of the largest maximal clique: Note that in the theoretical analysis, the regret scales with the size of the largest clique. We observed that having larger maximal cliques would demonstrate slow improvement in the objective function due to an overly complex additive model. Having smaller maximal

[8]Further details on this task can be found in Appendix B.8.

cliques would cause the objective function to asymptotically converge to a subpar value due to the limited expressiveness of the additive model.

- Total number of additive kernels: Due to computational reasons, we limited the total number of additive kernels generated. We encountered out-of-memory and computational issues on commodity GPUs if this value was set too high. Further details can be found in Appendix B.1.

5.4 Conclusion

We have proposed a HOM policy along with an effective optimization algorithm, HA-GP-UCB. Our HOM and HA-GP-UCB are designed to offer strong performance in high coordination multi-agent tasks under sparse or malformed reward on memory-constrained devices. HA-GP-UCB is a theoretically grounded approach to BO offering good regret bounds under reasonable assumptions. Our validation shows HA-GP-UCB outperforms RL and MARL at optimizing neural network policies in malformed reward scenarios. Our HOM optimized with HA-GP-UCB outperforms MARL approaches in high coordination multi-agent scenarios by leveraging the concepts of *role* and *role interaction*. Furthermore, we show through our drone delivery task, our approach outperforms MARL approaches in multi-agent coordination tasks with sparse reward. We make significant progress on high coordination multi-agent policy search by overcoming challenges posed by malformed reward and memory-constrained settings.

We are able to scale up approaches rooted in probabilistic methods while showing

their robustness advantages. We have demonstrated the scalability of these methods by applying them to complex high-dimensional tasks in the single agent and multi-agent setting. We show the superior performance of our approach when compared against competing approaches in reinforcement learning and HDBO. We have demonstrated the robustness of our approach by detailing the few hyperparameters of HA-GP-UCB and asserting that these hyperparameters were held constant throughout our extensive validation. This work gives evidence for scaling up decision-making under uncertainty, and also shows how it can outperform reinforcement learning approaches under certain scenarios.

5.5 Limitations

There are a few limitations of our work. Foremost among them concerns our usage of the Hessian in both our theoretical results and empirical validation. In practice, the Hessian is usually not available in Bayesian Optimization. One avenue of overcoming this limitation is using finite differences (Cheng, G. Wu, and J. Zhu 2021) on either zero'th order, or first order queries to determine the Hessian. Another limitation relies on our usage of the surrogate Hessian, which may only be available in the policy optimization setting due to the presence of a policy model. There are several avenues of improvement in this direction. One would be a formalization of the validity of the surrogate Hessian. Another would be a removal of dependence on the surrogate Hessian by using zero'th order, or first order queries. These challenging questions remain valuable for future work.

Chapter 6

Adversarially Designed Games

6.1 Introduction

Reinforcement learning, similar to deep learning, has experienced tremendous success in recent years (Yuxi Li 2017). This success has been demonstrated among a wide variety of complex environments (Mnih, Kavukcuoglu, Silver, Graves, et al. 2013; Silver, Huang, et al. 2016; Berner et al. 2019). It has also been demonstrated that reinforcement learning encounters difficulty in environments with sparse reward (Salimans and R. Chen 2018; Aubret, Matignon, and Hassas 2019).

The suboptimal behavior of reinforcement learning under sparse reward has been noted by several works (S. Singh, Barto, and Chentanez 2004; Zheng, Oh, and S. Singh 2018; Aubret, Matignon, and Hassas 2019; Simsek and Barto 2006; Linke et al. 2020). This shortcoming motivates an inquiry into whether reward structure can significantly affect the performance of reinforcement learning algorithms.

We perform an investigation into whether sparse or malformed reward detrimentally affects performance of reinforcement learning. We demonstrate a link between the behavior of the reward in an environment and the performance of reinforcement

learning approaches. In particular, we show that an adversarially designed reward structure detrimentally affects the performance of reinforcement learning approaches. In many environments, a useful reward structure enables the strong performance of reinforcement learning. However, it is possible to design scenarios with malformed reward such that strong performance becomes difficult to achieve. We term this family of environments, Adversarially Designed Games.

In addition to showing the subpar performance of reinforcement learning, we also show that a strong policy in these environments can be found by Bayesian Optimization (BO). BO has recently emerged as a competitor to reinforcement learning for finding strong policies in challenging environments (Letham et al. 2020; Müller, Rohr, and Trimpe 2021). Following this research direction, we provide theoretical guarantees that BO can find strong policies in such environments. Thus, it is not the case that Adversarially Designed Games are difficult or challenging overall. Instead, there is an empirically observed *suboptimality gap* between gradient based or local approaches such as reinforcement learning, and global optimization approaches such as BO. The presence of this suboptimality gap demonstrates the shortcomings of reinforcement learning at solving this family of games and makes them a valuable benchmark for further research in reinforcement learning.

To demonstrate these properties, our family of games is designed to be both simple, and exhibit behavior which poses challenges for local gradient based optimization techniques such as reinforcement learning. To well isolate this problem scenario, we focus on some of the simplest class of games known as one-person Solitaire games (Backhouse, W. Chen, and Ferreira 2010). Within this class of games, we

design a family that is a one-move game, with an optimal deterministic policy. This simplified design allows us to reason about the behavior of reinforcement learning and BO approaches when optimizing a policy.

For our theoretical results in reinforcement learning, we focus on policy gradient approaches (Silver, Lever, et al. 2014). Policy gradient has emerged as a dominant and performant paradigm in reinforcement learning in recent years (Arulkumaran et al. 2017). In policy gradient, a policy forming a set of decision-making rules in the environment is repeatedly improved using reinforcement learning. Typically, this policy is represented as a neural network. For our family of games, we show that policy gradient approaches have a tendency to converge to suboptimal policies. This difficulty arises from the reward structure of our designed family of games.

On the other hand, global search approaches such as BO do not suffer from this difficulty. It is well understood that the performance of BO approaches such as GP-UCB is related to the Fourier transform of the function being optimized (Kanagawa et al. 2018). In our family of games, the function being optimized (the policy) has a well-behaved and bounded Fourier transform. This property allows us to give theoretical guarantees on the convergence rate of BO to the optimal policy.

To show the above claims, we first design the game and the policy space. We then derive policy gradient rules in order to reason about the performance of policy gradient approaches. Following this derivation, we prove theoretical results showing the difficulty finding the optimal policy using policy gradient approaches. We also show that optimizing the policy is something which can be accomplished using BO approaches with strong performance guarantees.

In addition to the above theoretical contributions, we also conduct in depth empirical validation of reinforcement learning approaches vs. BO to empirically demonstrate the suboptimality gap. Our empirical validation shows that indeed the theoretically reasoned about suboptimality gap does emerge using modern policy gradient approaches. In this work, we make the following contributions:

- We introduce the concept of Adversarially Designed Games, a family of environments specifically crafted to expose the weaknesses of reinforcement learning in the presence of adversarial reward structures. These games serve as a valuable benchmark for evaluating the robustness of reinforcement learning algorithms.
- We establish links between the reward structure of an environment and the potential difficulty of optimizing policies for these environments using reinforcement learning.
- We provide theoretical guarantees for using Bayesian Optimization (BO) to find strong policies in Adversarially Designed Games. Our results show that BO can overcome the adversarially designed reward structure, offering a competitive alternative for policy optimization.
- We conduct comprehensive empirical experiments to demonstrate the suboptimality gap between reinforcement learning and BO. Our results highlight Adversarially Designed Games as a scenario where reinforcement learning struggles and where BO excels, providing insights into the conditions under which each approach is most effective. By highlighting the shortcomings of

current approaches, we aim to spur the development of more robust and effective reinforcement learning methods.

- We offer a detailed analysis of policy gradient methods, showing their tendency to converge to suboptimal policies in the presence of adversarial reward structures. This analysis underscores the importance of considering reward design in the development and evaluation of reinforcement learning algorithms.

6.2 Design

In this section, we demonstrate the design of our proposed Adversarially Designed Games. This simple family of games can prove difficult to solve (i.e., reach the optimal policy) for gradient based optimization. This difficulty arises from the limited insight provided by the gradient for solving the game. On the other hand, this family of games can be shown to be solved by BO.

We show that our proposed study of Adversarially Designed Games is challenging to solve for Policy Gradient methods. Policy Gradient approaches have become increasingly popular in reinforcement learning and have demonstrated state-of-the-art performance in many use cases and benchmarks.

Our family of Adversarially Designed Games is a continuous-valued solitaire game. The optimal policy for this family of games is deterministic. The reward for our solitaire game is defined as follows:

$$R(a) = \frac{c_1 \cos(a - \phi)}{e^{(a-\phi)^2}}. \tag{6.1}$$

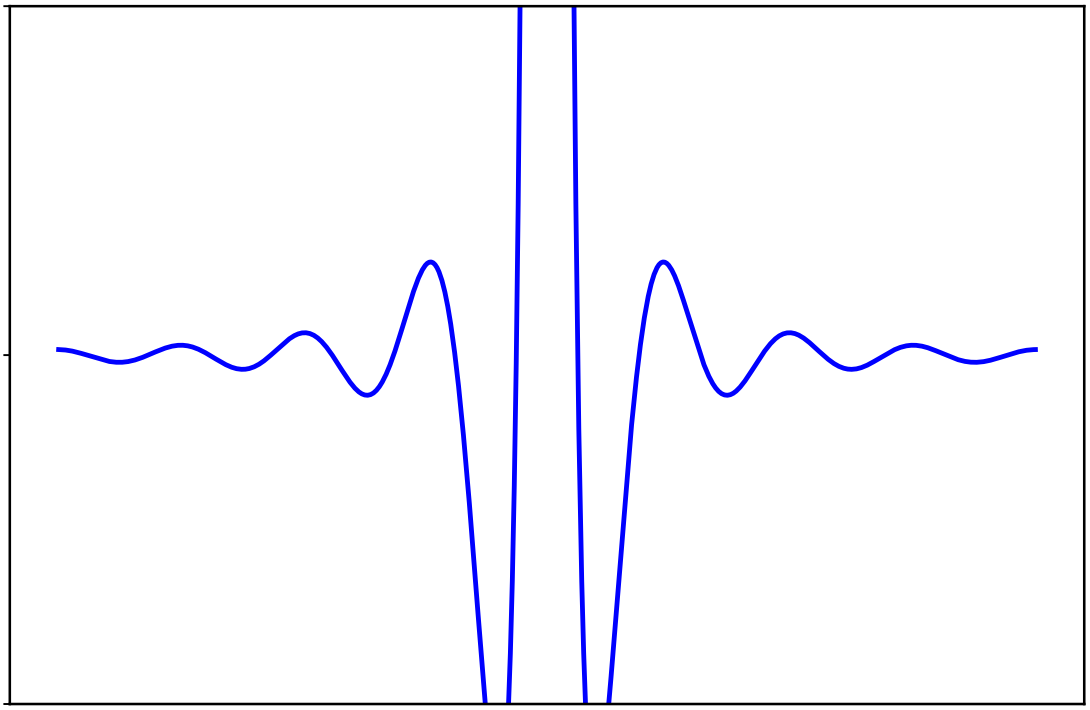

Figure 6.1: A visualization of the function $\cos(a)/a^2$, which exhibits similar periodic behavior to our solitaire game. The several local optimal in the function may cause gradient based optimizers to achieve suboptimal values during optimization.

Without loss of generality, we assume that the phase equals zero, i.e., $\phi = 0$. An agent playing this game takes a singular action which is the policy with $a \in \mathbb{R}$. This family of games has an optimal action at $a = 0$. We visualize a function with similar periodic behavior in Fig. 6.1. As can be the seen, the several local optima may cause difficulty for gradient based optimizers at reaching the global optima.

We specify the deterministic policy with respect to the above game:

$$\pi_\theta(a) \triangleq \delta_\theta(a) \tag{6.2}$$

where δ_θ is the Dirac-delta function, which is zero unless $a = \theta$. We note that this policy space is sufficiently expressive to capture the optimal policy for our family

of games, specifically with the policy $\pi_0(\cdot)$.

As is the practice in reinforcement learning during training of the policy using gradient descent, we add an exploration term.

$$\pi_\theta(a) \triangleq (1-\beta)\delta_\theta(a) + \beta U(a) \tag{6.3}$$

In addition to the Dirac-delta impulse at θ, there is an exploration factor to encourage *exploration* within the policy that is modulated by β.

Given the above definition, the optimization objective is defined as follows:

$$J(\theta) \triangleq \mathbb{E}_{\pi_\theta}\left[R(a)\right] = \mathbb{E}_{a\sim\pi_\theta}\left[\log \pi_\theta(a) R(a)\right] = \int \pi_\theta(a) R(a) da = R(\theta) \tag{6.4}$$

where $R(\cdot)$ is the reward function defined earlier.

We derive the policy gradient for the above formulation as follows:

$$\nabla_\theta J(\theta) = \mathbb{E}_{a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a) R(a)\right] = \nabla_\theta \int \pi_\theta(a) R(a) da \tag{6.5}$$

Given the above and the definition of $R(a)$, we have the following specification for the policy gradient:

$$\nabla_\theta J(\theta) = (1-\beta)\nabla_\theta \int \delta_\theta(a) R(a) da \tag{6.6}$$

Given the definition of the Dirac-delta function and using R as a Schwartz function[1], using the property of the distributional derivative we have:

[1]A Schwartz function is an infinitely differentiable function where every derivative decays faster than any polynomial. For further reading we recommend the work of Zemanian 1987 for a primer on distributions and Schwartz functions.

$$\nabla_\theta J(\theta) = (1-\beta)\nabla_\theta \int \delta_\theta(a)R(a)da = (1-\beta)\int \nabla_\theta \delta(a-\theta)R(a)da$$

where the second equality follows from the definition of $\delta_\theta = \delta(a-\theta)$ and $\delta(\cdot)$ is the Dirac-delta function with an impulse fixed at 0. Continuing with the derivation we have:

$$(1-\beta)\int \nabla_\theta \delta(a-\theta)R(a)da = (1-\beta)\int -\delta'(a-\theta)R(a)da = (1-\beta)\int \delta(a-\theta)R'(a)da$$

where the first equality follows from the derivative of the δ function, and the second equality follows from definition of the derivative of a test function. We complete the derivation:

$$(1-\beta)\int \delta(a-\theta)R'(a)da = (1-\beta)R'(\theta) \tag{6.7}$$

where the equality follows from the sifting property of the Dirac-delta function.

Given the gradient update rules, it becomes possible to utilize a family of gradient descent algorithms in order to optimize the policy space. Now we prove several lemmas that show the difficulty of optimizing the objective function using gradient based approaches along with the possibility of efficiently optimizing the policy space using BO.

6.3 Theoretical Contributions

We separate our theoretical contributions into two portions, showing results for both gradient based reinforcement learning approaches along with theoretical results

regarding BO approaches. We attempt to show that our family of Adversarially Designed Games is difficult to solve for reinforcement learning, but easy for BO approaches.

6.3.1 Reinforcement Learning Theoretical Results

Our first lemma shows that the number of local maxima between a randomly initialized policy, and the globally optimal policy is linearly correlated to the distance between the randomly initialized point and the global optima. This is a detriment for reinforcement learning as reaching the global optimal requires overcoming a numerous number of local maxima which is known to be difficult for gradient based optimization. We defer the proofs for all lemmas and theorems to the appendix.

Lemma 6.1. *Let the reward function be defined on the compact domain* $[-b, +b]$. *Let the function* $M(\theta)$ *denote the number of local maxima in the interval* $(-\theta, +\theta)$. *We have* $\mathbb{E}_{\theta \sim \mathcal{U}(-b,+b)}[M(\theta)] = \Omega(b)$.

In addition, we also consider the difficulty of optimizing the objective function using gradient descent with momentum methods. Gradient descent with momentum methods give the gradient descent process a *momentum* term to escape local optima. In particular, this momentum term builds inertia while navigating the loss surface of the optimization function. This inertia is then consumed to overcome local optima by allowing the optimizer to climb through the local valley or hill.

However, for the purpose of our reward function, we show that the loss surface is designed such that momentum optimizers have difficulty reaching the global optimal, as each successive local maxima is greater than the previous as the global maxima

is approached. Thus, the momentum accumulated from reaching the previous local maxima will be insufficient for overcoming the successive local maxima. To formalize this argument, we state the following Lemma:

Lemma 6.2. *Let the reward function be defined on the compact domain* $[-b, +b]$. *Let* $\hat{\theta}_1, \ldots, \hat{\theta}_n$ *be any sequence of local maxima of* $J(\cdot)$ *that are monotonically approaching the global maxima. That is* $sign(\hat{\theta}_1) =, \ldots, = sign(\hat{\theta}_n)$ *and* $|\hat{\theta}_1| >, \ldots, > |\hat{\theta}_n|$, $\frac{\partial J}{\partial \theta}(\hat{\theta}_1) =, \ldots, = \frac{\partial J}{\partial \theta}(\hat{\theta}_n) = 0$ *and* $\frac{\partial^2 J}{\partial \theta^2}(\hat{\theta}_1) < 0, \ldots, \frac{\partial^2 J}{\partial \theta^2}(\hat{\theta}_n) < 0$. *Then* $J(\hat{\theta}_i) < J(\hat{\theta}_j)$ *when* $i < j$.

Such characteristic behavior of the reward function is detrimental to momentum based optimizers, as the momentum cannot be accumulated to overcome the local minima between successive local maxima. These two lemmas indicate the difficulty of using gradient based methods to solve our Adversarially Designed family of games.

It remains an open question whether the above results can be strengthened. In particular, the following open question may deserve further study:

Let A represent an arbitrary deterministic gradient based optimization algorithm with access to the noisy gradient: $\widetilde{\nabla_\theta} \triangleq \nabla_\theta + \epsilon$, with $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Let the generalized reward function be:

$$R(\theta) \triangleq \frac{c_1 \cos\left(s(a-\phi)\right)}{e^{s(a-\phi)^2}}.$$

Let the reward function be picked randomly with, $s \sim [-b_s, +b_s]$, $\phi \sim [-b_\phi, +b_\phi]$, $c_1 \sim [-b_c, +b_c]$ Let A_θ represent A with $\theta \sim [-b, +b]$ as the initial point for gradient based optimization on the generalized reward function. Let $\widehat{A_\theta}$ be the

optimized result after completion of A. Are there values of σ^2, b, b_ϕ, b_s, b_c such that $\min_A J(\phi) - \mathbb{E}(J(\widehat{A_\theta}))$ be lower bounded by some quantity? That is, does a suboptimality gap provably exist for the generalized reward function?

6.3.2 Bayesian Optimization Theoretical Results

The second section of our theoretical results demonstrate the ability of Bayesian Optimization to achieve the global optima. This is possible by bounding the RKHS[2] norm of the function being optimized. Following this bounding, there exist well known theorems establishing the regret of well known algorithms such as GP-UCB. To begin our analysis we state a bound on the RKHS norm (Kanagawa et al. 2018) of the policy function being optimized:

$$||J||^2_{\mathcal{H}_k} = \frac{1}{2\pi^{1/2}} \int \frac{\mathcal{F}[J](\omega)\overline{\mathcal{F}[J](\omega)}}{\mathcal{F}[\psi](\omega)} d\omega \tag{6.8}$$

where $\mathcal{F}[\psi](\omega)$ represents the reference kernel and J represents the policy function being optimized. For our purposes we can use the unnormalized RBF kernel as the reference kernel.

To proceed with the analysis, we state the Fourier transform of the reward function, which is being optimized, along with the Fourier transform of the unnormalized RBF kernel.

$$\int_{-\infty}^{+\infty} \frac{c_1 \cos(a)}{e^{a^2}} e^{i\omega a} da = \frac{c_1}{2} \left(\frac{e^{-1/4\,(\omega-1)^2} + e^{-1/4\,(\omega+1)^2}}{\sqrt{2}} \right) \tag{6.9}$$

[2]A helpful primer on the relationship between RKHS norm and Bayesian Optimization is the work of Srinivas et al. 2010.

$$\int_{-\infty}^{+\infty} e^{-\mathrm{d}^2} e^{i\omega \mathrm{d}} d\mathrm{d} = \frac{1}{\sqrt{2}e^{\omega^2/4}} \tag{6.10}$$

The above can be verified by using the identity $cos(a) = \frac{e^{ia}+e^{-ia}}{2}$ and Bochner's theorem respectively.

Given the above, we can bound the RKHS norm as follows:

$$||J||^2_{\mathcal{H}_k} = \frac{1}{2\pi^{1/2}} \int \frac{\mathcal{F}[J](\omega)\overline{\mathcal{F}[J](\omega)}}{\mathcal{F}[\psi](\omega)} d\omega = 2c_1^2(1+e)\sqrt{\frac{1}{2e}} \tag{6.11}$$

Following the above bounding of the RKHS norm with respect to the RBF kernel, we can state the key theorem of this work.

Theorem 6.1. *Let the reward function, and the policy space be defined as earlier, then the cumulative regret of the BO Algorithm GP-UCB suffers when searching for the optimal policy is* $\widetilde{O}(\sqrt{T}(c_1^2 \log T + \log^2 T))$.

Where $\widetilde{O}(\cdot)$ is the same as $O(\cdot)$ with the log factors suppressed. Thus, by bounding the RKHS norm of the reward function, we are able to bound the regret suffered by GP-UCB when optimizing for the best policy. The regret is a key measure of the performance of the BO optimization algorithm. The above regret bound is tight up to poly-log factors, showing that not only GP-UCB shows strong performance, but this performance can only be slightly improved in the best case (Scarlett 2018). Thus, our analysis itself is also rigorous.

Our theoretical analysis has established an upper bound on the performance of BO at optimizing for the policy. In addition, we have also proved lemmas that show the difficulty of finding the optimal policy using gradient based approaches. Thus,

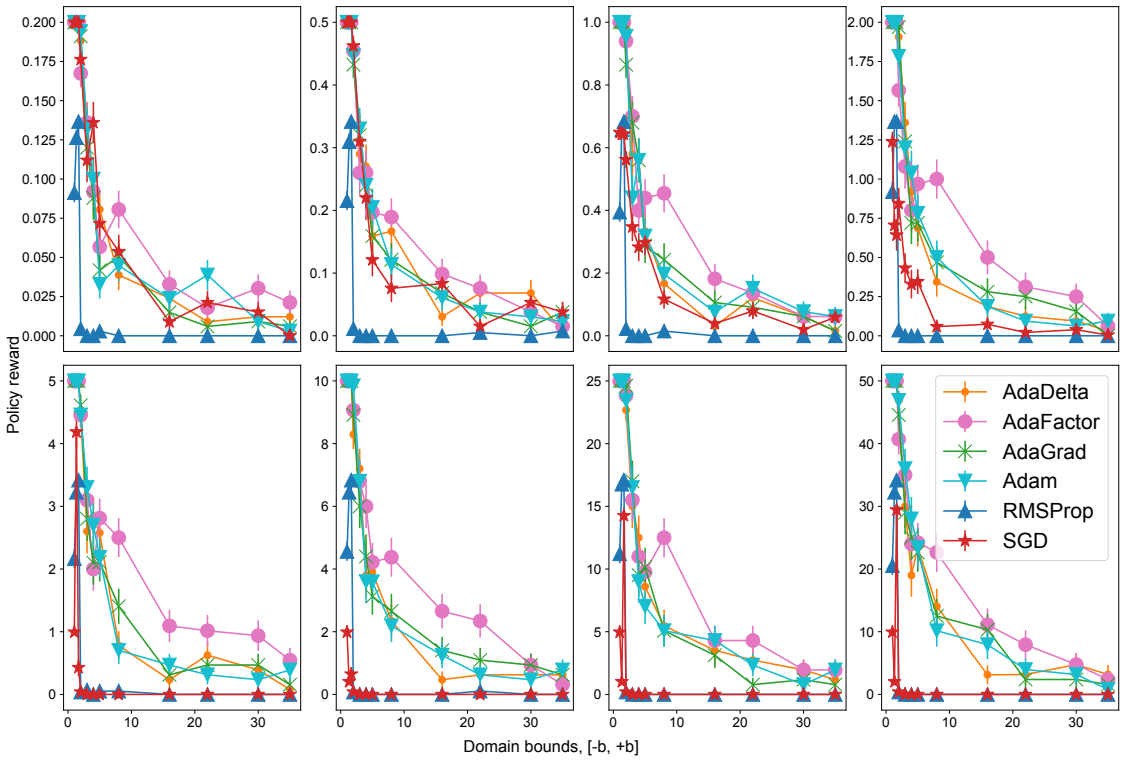

Figure 6.2: Policy gradient with various optimizers with c_1 varying from 0.2 to 50.0 and domain bounds varying from $[-1, +1]$ to $[-35, +35]$. As the domain bounds are increased, policy gradient suffers a significant drop in performance.

our theoretical analysis shows our family of games is easy or straightforward to solve using BO, but difficult to solve by RL.

Following this analysis, we perform empirical validation to support that in practice our family of Adversarially Designed Games can be optimized by a global optimization approach such as BO, and policy gradient approaches show consistently poor performance.

6.4 Validation

We validate our theoretical contributions in three different scenarios. First, we verify the poor performance of policy gradient in our family of Adversarially Designed

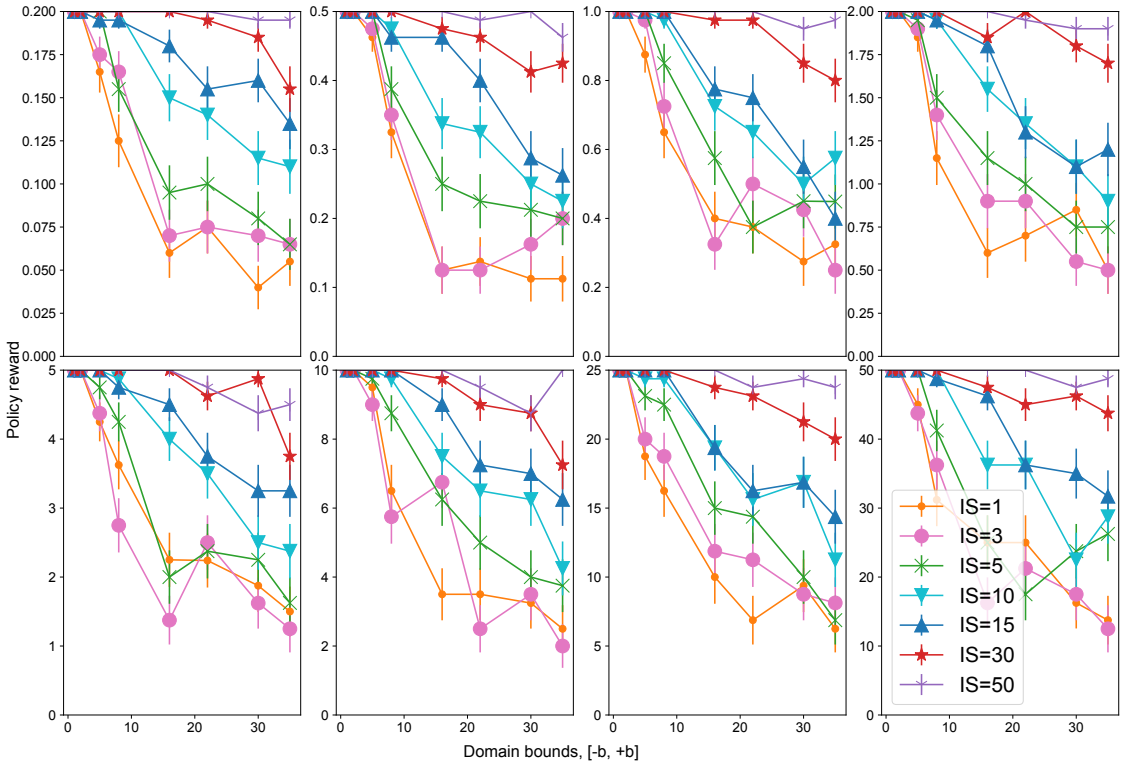

Figure 6.3: Bayesian Optimization with initial samples varying from $IS = 1$ to $IS = 50$. The constant c_1 is varied from 0.2 to 50.0 and domain bounds are varied from $[-1, +1]$ to $[-35, +35]$. As the domain bounds are increased, BO suffers performance loss. For the higher setting of initial samples, little to no performance loss is observed.

Games. Second, we validate the strong performance of BO under the same setting. Lastly, we compare the performance of policy gradient and BO to empirically establish the suboptimality gap.

To establish a diverse set of games within this family, we vary the scaling constant c_1 from 0.5 to 50.0. We also vary the domain bounds, $[-b, +b]$ from $b = 1$ to $b = 35$. This allows us to study how these family hyperparameters affect the performance of policy gradient or BO approaches. For all validation, we repeat our experiments 50 times.

For policy gradient methods, we verify using several common optimizers such

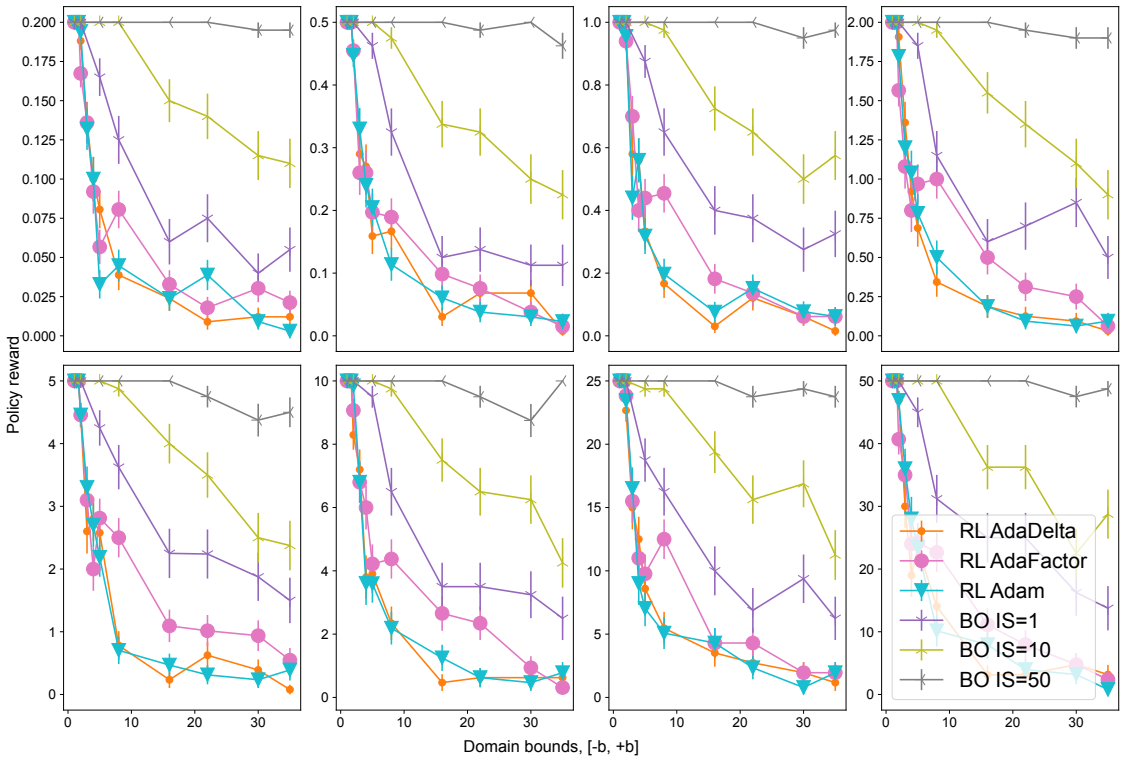

Figure 6.4: Reinforcement learning and Bayesian Optimization compared under Adversarially Designed Games. The constant c_1 is varied from 0.2 to 50.0 and domain bounds are varied from $[-1, +1]$ to $[-35, +35]$. As the domain bounds are increased, BO suffers far less performance loss than reinforcement learning methods. Even with 1 initial sample, BO outperforms all tested reinforcement learning methods.

as AdaDelta (Zeiler 2012), AdaFactor (Shazeer and Stern 2018), AdaGrad (Duchi, Hazan, and Singer 2011), Adam (Kingma and Ba 2015), RMSProp (Ruder 2016), and SGD (Ruder 2016). We present this validation in Fig. 6.2 where we vary both the scaling constant c_1 and the domain bounds, $[-b, +b]$.

As can be seen in Fig. 6.2, the performance of policy gradient with all optimizers leaves much to be desired. Performance significantly degrades as domain bounds are even slightly increased from the lowest possible setting of $[-1, +1]$. This is due to policy gradient methods becoming stuck at local optima as indicated by Lemma 6.1 and Lemma 6.2. Although some optimizers such as AdaDelta, AdaFactor, and Adam

show better performance than other optimizers, the performance is still consistently poor across all tested optimizers.

We contrast the above result with BO. We repeat the same set of experiments, and utilize GP-UCB to optimize for the policy. We vary the initial samples for BO from 1 to 50. We present our results in Fig. 6.3.

In comparison to the performance of policy gradient methods, BO offers far superior performance, even with an exceptionally low ($IS = 1$) number of initial samples. For a more realistic number of initial samples, such as $IS = 15$ or higher, the performance of BO far supercedes that of policy gradient methods. Finally, with many samples (i.e., $IS = 30$ or $IS = 50$), there is little to no degradation in BO performance even as the domain bounds are increased to high values such as $[-35, +35]$. This agrees with our theoretical claim about the performance of BO as presented in Theorem 6.1.

For ease of comparison, we present a combined figure comparing policy gradient methods and BO in Fig. 6.4. We highlight the generally superior performance of BO. In addition, we also highlight that even with an exceptionally low ($IS = 1$) number of initial samples, a suboptimality gap exists between policy gradient and BO thus verifying our theoretical contributions.

6.5 Conclusion

We have introduced the concept of Adversarially Designed Games, a family of environments specifically created to highlight the weaknesses of reinforcement learning under adversarial reward structures. Our comprehensive study links the reward struc-

ture of an environment to the difficulty of optimizing policies using reinforcement learning methods.

Our theoretical and empirical analyses reveal that policy gradient approaches tend to converge to suboptimal policies in adversarially designed scenarios due to the numerous local maxima that impede the optimization process. In contrast, we provide theoretical guarantees that Bayesian Optimization (BO) can effectively find strong policies in these challenging environments, overcoming the adversarially designed reward structures.

Our empirical validation demonstrates the suboptimality gap between reinforcement learning and BO, highlighting that BO significantly outperforms reinforcement learning in Adversarially Designed Games. This performance gap underscores the need for more robust and effective reinforcement learning methods.

By establishing a new benchmark through Adversarially Designed Games, we pave the way for future research to address the shortcomings of current reinforcement learning algorithms and to develop more resilient approaches capable of handling adversarial reward environments. This work demonstrates the importance of reward design in reinforcement learning and provides a foundation for advancing optimization techniques in complex, adversarial settings.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

We summarize the contributions of this thesis in showing how approaches rooted in probabilistic methods can be scaled up to be competitive with deep learning or reinforcement learning approaches. We investigated this problem in the setting of early pruning of deep neural networks, as well as in the setting of policy optimization in the single agent and multi-agent setting. Our contributions are detailed below.

Pruning During Training by Network Efficacy Modeling. We have demonstrated a scalable algorithm to perform early pruning. Our algorithm is able to perform early pruning in linear time with bounds given on its suboptimality gap. Development of this algorithm required theoretical principles rooted in submodular optimization and assuming a multivariate Gaussian distribution of network element efficacy. Using these approaches, we are able to solve the early pruning problem in linear time. We performed extensive validation on our approach to show its superior performance at preserving network efficacy against competing approaches from the deep learning literature. We demonstrate this in several parts. First, we

show in a smaller-scale network, our proposed MOGP model is able to well capture network saliency evolution over time. We evaluate and show the validity of important subcomponents of our approach which improve empirical performance, such as the BEP-LITE heuristic and dynamic penalty scaling. Following these experiments, we then validate the proposed BEP algorithm on the smaller scale network and show its competitive performance at preserving network efficacy at test-time. We then evaluate both BEP and BEP-LITE on the ResNet-50, architecture and demonstrate that this competitive performance translates to larger scale deep neural networks. We also show our approach is robust to changes in hyperparameters. Our work demonstrates that algorithms rooted in probabilistic methods are able to scale up in the deep learning setting of pruning.

Hessian-Aware Bayesian Optimization for Decision-Making Models. We have demonstrated a scalable approach to optimize Decision-Making Models suitable for memory constrained devices. Our proposed approach relies on a Higher-Order Model (HOM) as well as a scalable approach to High-Dimensional Bayesian Optimization (HDBO). For the HOM, we utilize the abstractions of role and role interaction to design an expressive policy which is memory efficient. For our approach to HDBO, HA-GP-UCB, we propose a new approach of learning the dependency structure between the dimensions to simplify the learning problem. This approach decomposes a large optimization problem into many smaller optimization problems which are easier to solve. We additionally provide a regret bound for HA-GP-UCB. We extensively validate our claims. We first validate our HOM and

show that the role and role interaction offer measurable advantages in empirical performance. We further investigate the internal behavior of the HOM and show that role and role interaction work as expected in abstracting agent behavior through role, and modulating agent behavior through role interaction. We compare against reinforcement learning approaches in the single agent and multi-agent setting. We show that in these settings, our approach shows superior performance in the malformed or sparse reward setting. We also show that our approach is competitive with other HDBO approaches. Finally, we demonstrate the robustness of our approach by detailing the two hyperparameters used in HA-GP-UCB and mentioning that they were held constant throughout all portions of the validation. Our work demonstrates that algorithms and approaches rooted in probabilistic methods are able to scale up in the setting of policy optimization to be competitive to reinforcement learning approaches.

Adversarially Designed Games. We introduced Adversarially Designed Games, a novel family of environments tailored to expose the weaknesses of reinforcement learning under adversarial reward structures. These environments serve as a valuable benchmark for evaluating the robustness of reinforcement learning algorithms. Our work establishes a critical link between the reward structure of an environment and the difficulty of optimizing policies using reinforcement learning. We provide theoretical and empirical evidence that gradient-based methods, particularly policy gradient approaches, struggle to achieve optimal performance in these adversarial settings due to numerous local maxima. On the other hand, we present theoretical

guarantees that BO, specifically GP-UCB, can efficiently find strong policies in Adversarially Designed Games. By bounding the RKHS norm of the reward function, we show that BO can overcome adversarially designed reward structures, offering a competitive alternative for policy optimization. We empirically validated these theoretical result to demonstrate a significant suboptimality gap between reinforcement learning and BO. We show that BO consistently outperforms reinforcement learning in Adversarially Designed Games, even with a low number of initial samples. This highlights the robustness and efficiency of BO in optimizing policies under challenging reward structures.

This thesis shows under which settings and scenarios, approaches rooted in probabilistic methods outperforms approaches from the deep learning and reinforcement learning literature. We show the possibility of superior performance in the early pruning setting and compare against deep learning approaches. We show the possibility of superior performance in the sparse reward policy optimization setting and compare against reinforcement learning approaches. Finally, we unify our works by proposing Adversarially Designed Games which serves as both a benchmark showing consistently stronger performance for Bayesian Optimization methods over reinforcement learning methods under adversarial reward structures.

Our family of Adversarially Designed Games also serves as a launching platform for future work. This benchmark serves as an excellent testbed for *robust* reinforcement learning methods, which fare well under adversarial reward structures. In addition, although Bayesian Optimization methods offer theoretically strong performance in the asymptotic case, there is still room for improvement for optimizing

adversarially designed functions. Thus, how to develop robust and practical Bayesian Optimization methods that show consistently strong performance for Adversarially Designed Games is also an open question for future work.

Bibliography

Akhtar, Naveed, Ajmal Mian, Navid Kardan, and Mubarak Shah. "Advances in adversarial attacks and defenses in computer vision: A survey". In: *IEEE Access* 9 (2021), pp. 155161–155196.

Akrour, Riad, Dmitry Sorokin, Jan Peters, and Gerhard Neumann. "Local Bayesian optimization of motor skills". In: *Proc. ICML*. 2017.

Allen-Zhu, Zeyuan, Yuanzhi Li, and Yingyu Liang. "Learning and generalization in overparameterized neural networks, going beyond two layers". In: *Proc. NeurIPS*. 2019, pp. 6155–6166.

Álvarez, Mauricio A. and Neil D. Lawrence. "Computationally efficient convolved multiple output Gaussian processes". In: *JMLR* 12.1 (2011), pp. 1459–1500.

Ament, Sebastian E. and Carla P. Gomes. "Scalable first-order Bayesian optimization via structured automatic differentiation". In: *Proc. ICML*. 2022, pp. 500–516.

Ament, Sebastian E. and Carla P. Gomes. "Scalable first-order Bayesian optimization via structured automatic differentiation". In: *Proc. ICML*. 2022, pp. 500–516.

Apicella, Andrea, Francesco Donnarumma, Francesco Isgrò, and Roberto Prevete. "A survey on modern trainable activation functions". In: *Neural Networks* 138 (2021), pp. 14–32.

Aronszajn, Nachman. “Theory of reproducing kernels”. In: *Transactions of the American mathematical society* 68.3 (1950), pp. 337–404.

Arulkumaran, Kai, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. “Deep reinforcement learning: A brief survey”. In: *IEEE Signal Process. Mag.* 34.6 (2017), pp. 26–38.

Åström, Karl Johan, Tore Hägglund, Chang C Hang, and Weng K Ho. “Automatic tuning and adaptation for PID controllers - A survey”. In: *Control Engineering Practice* 1.4 (1993), pp. 699–714.

Aubret, Arthur, Laëtitia Matignon, and Salima Hassas. “A survey on intrinsic motivation in reinforcement learning”. In: *CoRR* abs/1908.06976 (2019).

Backhouse, Roland Carl, Wei Chen, and João F. Ferreira. “The algorithmics of Solitaire-like games”. In: *Proc. MPC*. Vol. 6120. 2010, pp. 1–18.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. “Neural machine translation by jointly learning to align and translate”. In: *Proc. ICLR*. 2015.

Barbos, Andrei-Cristian, Francois Caron, Jean-François Giovannelli, and Arnaud Doucet. “Clone MCMC: parallel high-dimensional Gaussian Gibbs sampling”. In: *Proc. NeurIPS*. 2017.

Barca, Jan Carlo and Y Ahmet Sekercioglu. “Swarm robotics reviewed”. In: *Robotica* 31.3 (2013), pp. 345–359.

Bellec, Guillaume, David Kappel, Wolfgang Maass, and Robert A. Legenstein. “Deep rewiring: Training very sparse deep networks”. In: *Proc. ICLR*. 2018.

Bellman, Richard E. “Adaptive control processes: A guided tour”. Princeton, New Jersey: Princeton University Press, 2015.

Berkeley, Joel, Henry B. Moss, Artem Artemev, Sergio Pascual-Diaz, Uri Granta, Hrvoje Stojic, Ivo Couckuyt, Jixiang Qing, Nasrulloh Loka, Andrei Paleyes, Sebastian W. Ober, and Victor Picheny. "Trieste". Version 0.12.0. July 2022. URL: https://github.com/secondmind-labs/trieste.

Berner, Christopher, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. "Dota 2 with large scale deep reinforcement learning". In: *arXiv preprint arXiv:1912.06680* (2019).

Bertsekas, Dimitri P. "Approximate policy iteration: A survey and some new methods". In: *Journal of Control Theory and Applications* 9 (2011), pp. 310–335.

Bollobás, Béla and Paul Erdös. "Cliques in random graphs". In: *Proc. Cambridge Philosophical Society*. 1976.

Box, George EP and George C Tiao. "Bayesian inference in statistical analysis". John Wiley & Sons, 2011.

Brockman, Greg, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. "OpenAI Gym". 2016. eprint: arXiv:1606.01540.

Buluç, Aydin and John R. Gilbert. "Challenges and advances in parallel sparse matrix-matrix multiplication". In: *Proc. ICCP*. 2008, pp. 503–510.

Çalisir, Sinan and Meltem Kurt PehlIvanoõlu. "Model-free reinforcement learning algorithms: A survey". In: *Proc. SIU*. 2019, pp. 1–4.

Chang, Yupeng, Xu Wang, Jindong Wang, Yuan Wu, Kaijie Zhu, Hao Chen, Linyi Yang, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. "A survey on evaluation of large language models". In: *arXiv preprint arXiv:2307.03109* (2023).

Cheng, Shuyu, Guoqiang Wu, and Jun Zhu. "On the Convergence of Prior-Guided Zeroth-Order Optimization Algorithms". In: *Proc. NeurIPS*. 2021, pp. 14620–14631.

Chowdhury, Sayak Ray and Aditya Gopalan. "On kernelized multi-armed bandits". In: *Proc. ICML*. Ed. by Doina Precup and Yee Whye Teh. 2017, pp. 844–853.

Chu, John T. "On bounds for the normal integral". In: *Biometrika* 42 (1955), pp. 263–265.

Claus, Caroline and Craig Boutilier. "The dynamics of reinforcement learning in cooperative multiagent systems". In: *Proc. IAAI*. 1998.

Clevert, Djork-Arné, Thomas Unterthiner, and Sepp Hochreiter. "Fast and accurate deep network learning by exponential linear units (ELUs)". In: *Proc. ICLR*. 2016.

Condon, Anne. "The complexity of stochastic games". In: *Information and Computation* 96.2 (1992), pp. 203–224.

Courbariaux, Matthieu, Yoshua Bengio, and Jean-Pierre David. "BinaryConnect: Training deep neural networks with binary weights during propagations". arXiv:1511.00363. 2015.

D'Eramo, Carlo, Davide Tateo, Andrea Bonarini, Marcello Restelli, and Jan Peters. "Mushroomrl: Simplifying reinforcement learning research". In: (2021).

Dai, Xiaoliang, Hongxu Yin, and Niraj K. Jha. "NeST: A neural network synthesis tool based on a grow-and-prune paradigm". In: *IEEE Trans. Computers* 68.10 (2019), pp. 1487–1497.

Das, Abhishek, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. "TarMAC: Targeted multi-agent communication". In: *Proc. ICML*. 2019, pp. 1538–1546.

Denton, Emily L., Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. "Exploiting linear structure within convolutional networks for efficient evaluation". In: *Proc. NeurIPS*. 2014, pp. 1269–1277.

Dettmers, Tim and Luke Zettlemoyer. "Sparse networks from scratch: Faster training without losing performance". arXiv:1907.04840. 2019.

Dong, Xin, Shangyu Chen, and Sinno Jialin Pan. "Learning to prune deep neural networks via layer-wise optimal brain surgeon". In: *Proc. NeurIPS*. 2017, pp. 4857–4867.

Dorling, Kevin, Jordan Heinrichs, Geoffrey G. Messier, and Sebastian Magierowski. "Vehicle routing problems for drone delivery". In: *IEEE Trans. Syst. Man Cybern. Syst.* 47.1 (2017), pp. 70–85.

Dorri, Ali, Salil S. Kanhere, and Raja Jurdak. "Multi-agent systems: A survey". In: *IEEE Access* 6 (2018), pp. 28573–28593.

Dubey, Shiv Ram, Satish Kumar Singh, and Bidyut Baran Chaudhuri. "Activation functions in deep learning: A comprehensive survey and benchmark". In: *Neurocomputing* 503 (2022), pp. 92–108.

Duchi, John C., Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization". In: *JMLR* 12 (2011), pp. 2121–2159.

Dugas, Charles, Yoshua Bengio, François Bélisle, Claude Nadeau, and René Garcia. "Incorporating second-order functional knowledge for better option pricing". In: *Proc. NeurIPS*. 2000, pp. 472–478.

Duvenaud, David, Hannes Nickisch, and Carl Edward Rasmussen. "Additive Gaussian Processes". In: *Proc. NeurIPS*. 2011, pp. 226–234.

Elsken, Thomas, Jan Hendrik Metzen, and Frank Hutter. "Neural architecture search: A Survey". In: *JMLR* 20 (2019), 55:1–55:21.

Eriksson, David, Michael Pearce, Jacob R. Gardner, Ryan Turner, and Matthias Poloczek. "Scalable global optimization via local Bayesian optimization". In: *Proc. NeurIPS*. 2019.

Eriksson, David, Michael Pearce, Jacob R. Gardner, Ryan Turner, and Matthias Poloczek. "Scalable global optimization via local Bayesian optimization". In: *Proc. NeurIPS*. 2019, pp. 5497–5508.

Ermolova, Natalia Y. and Sven-Gustav Häggman. "Simplified bounds for the complementary error function". In: *Proc. Eurasip*. 2004, pp. 1087–1090.

Ernst, Damien, Mevludin Glavic, Pierre Geurts, and Louis Wehenkel. "Approximate value iteration in the reinforcement learning context. Application to electrical power system control." In: *International Journal of Emerging Electric Power Systems* 3.1 (2005).

Feller, William. "An introduction to probability theory and its applications". Vol. 81. 1991.

Foerster, Jakob, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. "Counterfactual multi-agent policy gradients". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

Fortunato, Santo and Claudio Castellano. "Community structure in graphs". In: *Encyclopedia of Complexity and Systems Science*. 2009, pp. 1141–1163.

Frankle, Jonathan and Michael Carbin. "The lottery ticket hypothesis: Finding sparse, trainable neural networks". In: *Proc. ICLR*. 2019.

Fröhlich, Lukas P., Melanie N. Zeilinger, and Edgar D. Klenske. "Cautious Bayesian optimization for efficient and scalable policy search". In: *Proc. L4DC*. 2021.

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

Gale, Trevor, Erich Elsen, and Sara Hooker. "The state of sparsity in deep neural networks". arXiv:1902.09574. 2019.

Gilmer, Justin, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. "Neural message passing for quantum chemistry". In: *Proc. ICML*. 2017, pp. 1263–1272.

Gleave, Adam, Michael Dennis, Cody Wild, Neel Kant, Sergey Levine, and Stuart Russell. "Adversarial policies: Attacking deep reinforcement learning". In: *Proc. ICLR*. 2020.

Gronauer, Sven and Klaus Diepold. "Multi-agent deep reinforcement learning: a survey". In: *Artif. Intell. Rev.* 55.2 (2022), pp. 895–943.

Guo, Chuan, Alexandre Sablayrolles, Hervé Jégou, and Douwe Kiela. "Gradient-based adversarial attacks against text transformers". In: *Proc. EMNLP*. 2021, pp. 5747–5757.

Guo, Yiwen, Anbang Yao, and Yurong Chen. "Dynamic network surgery for efficient DNNs". In: *Proc. NeurIPS*. 2016, pp. 1379–1387.

Haarnoja, Tuomas, 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*. PMLR. 2018, pp. 1861–1870.

Hamlili, Ali. "Intelligibility of Erdös-Rényi random graphs and time varying social network modeling". In: *Proc. ICSDE*. 2017, pp. 201–206.

Han, Eric, Ishank Arora, and Jonathan Scarlett. "High-dimensional Bayesian optimization via tree-Structured additive models". In: *Proc. AAAI*. 2021, pp. 7630–7638.

Han, Eric, Ishank Arora, and Jonathan Scarlett. "High-dimensional Bayesian optimization via tree-structured additive models". In: *Proc. AAAI*. 2021.

Han, Kai, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, Zhaohui Yang, Yiman Zhang, and Dacheng Tao. "A Survey on Vision Transformer". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 45.1 (2023), pp. 87–110.

Han, Song, Jeff Pool, John Tran, and William Dally. "Learning both weights and connections for efficient neural networks". In: *Proc. NeurIPS*. 2015, pp. 1135–1143.

Hassibi, Babak and David G. Stork. “Second order derivatives for network pruning: Optimal brain surgeon”. In: *Proc. NeurIPS*. 1992, pp. 164–171.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep residual learning for image recognition”. In: *Proc. CVPR*. 2016, pp. 770–778.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Identity mappings in deep residual networks”. In: *Proc. ECCV*. 2016, pp. 4432–4440.

He, Yihui, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. “AMC: AutoML for model compression and acceleration on mobile devices”. In: *Proc. ECCV*. 2018, pp. 784–800.

Heidrich-Meisner, Verena and Christian Igel. “Evolution strategies for direct policy search”. In: *Proc. PPSN*. 2008, pp. 428–437.

Hennig, Philipp and Christian J. Schuler. “Entropy search for information-efficient global optimization”. In: *JMLR* 13 (2012), pp. 1809–1837.

Hensman, James, Alexander Matthews, and Zoubin Ghahramani. “Scalable variational Gaussian process classification”. In: *Proc. AISTATS*. 2015, pp. 351–360.

Hinton, Geoffrey E., Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. “Improving neural networks by preventing co-adaptation of feature detectors”. arXiv:1207.0580. 2012.

Hinton, Geoffrey E., Oriol Vinyals, and Jeffrey Dean. “Distilling the Knowledge in a neural network”. arXiv:1503.02531. 2015.

Hochreiter, Sepp and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

Hospedales, Timothy M., Antreas Antoniou, Paul Micaelli, and Amos J. Storkey. "Meta-learning in neural networks: A survey". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 44.9 (2022), pp. 5149–5169.

Hu, Junling and Michael P Wellman. "Nash Q-learning for general-sum stochastic games". In: *JMLR* 4.Nov (2003), pp. 1039–1069.

Hubara, Itay, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. "Quantized neural networks: Training neural networks with low precision weights and activations". In: *JMLR* 18.1 (2017), pp. 6869–6898.

Idelbayev, Yerlan and Miguel Á. Carreira-Perpiñán. "LC: A flexible, extensible open-source toolkit for model compression". In: *Proc. CIKM*. 2021, pp. 4504–4514.

Idelbayev, Yerlan and Miguel Á. Carreira-Perpiñán. "More general and effective model compression via an additive combination of compressions". In: *Proc. ECML PKDD*. 2021, pp. 233–248.

Jabbar, Abdul, Xi Li, and Bourahla Omar. "A Survey on generative adversarial networks: Variants, applications, and training". In: *ACM Comput. Surv.* 54.8 (2022), 157:1–157:49.

Jaderberg, Max, Andrea Vedaldi, and Andrew Zisserman. "Speeding up convolutional neural networks with low rank expansions". In: *Proc. BMVC*. 2014.

Johnson, Matthew J., James Saunderson, and Alan S. Willsky. "Analyzing hogwild parallel Gaussian Gibbs sampling". In: *Proc. NeurIPS*. 2013.

Jones, Donald R., Matthias Schonlau, and William J. Welch. “Efficient global optimization of expensive black-box functions”. In: *J. Glob. Optim.* 13.4 (1998), pp. 455–492.

Jorge, Pau de, Amartya Sanyal, Harkirat S. Behl, Philip H. S. Torr, Grégory Rogez, and Puneet K. Dokania. “Progressive skeletonization: Trimming more fat from a network at initialization”. In: *Proc. ICLR*. 2021.

Kaelbling, Leslie Pack, Michael L. Littman, and Anthony R. Cassandra. “Planning and acting in partially observable stochastic domains”. In: *Artif. Intell.* 101.1-2 (1998), pp. 99–134.

Kaelbling, Leslie Pack, Michael L. Littman, and Andrew W. Moore. “Reinforcement learning: A survey”. In: *J. Artif. Intell. Res.* 4 (1996), pp. 237–285.

Kanagawa, Motonobu, Philipp Hennig, Dino Sejdinovic, and Bharath K Sriperumbudur. “Gaussian processes and kernel methods: A review on connections and equivalences”. In: *arXiv preprint arXiv:1807.02582* (2018).

Kandasamy, Kirthevasan, Jeff G. Schneider, and Barnabás Póczos. “High dimensional Bayesian optimisation and bandits via additive models”. In: *Proc. ICML*. 2015, pp. 295–304.

Karaboga, Dervis and Bahriye Akay. “A survey: Algorithms simulating bee swarm intelligence”. In: *Artificial intelligence review* 31 (2009), pp. 61–85.

Karniadakis, George Em, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. “Physics-informed machine learning”. In: *Nature Reviews Physics* 3.6 (2021), pp. 422–440.

Karnin, Ehud D. “A simple procedure for pruning back-propagation trained neural networks”. In: *IEEE Trans. Neural Networks* 1.2 (1990), pp. 239–242.

Kingma, Diederik P. and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *Proc. ICLR*. 2015.

Kirschner, Johannes, Mojmir Mutny, Nicole Hiller, Rasmus Ischebeck, and Andreas Krause. “Adaptive and safe Bayesian optimization in high dimensions via one-dimensional subspaces”. In: *Proc. ICML*. 2019, pp. 3429–3438.

Konda, Vijay R. and John N. Tsitsiklis. “Actor-critic algorithms”. In: *Proc. NeurIPS*. 1999, pp. 1008–1014.

Krause, Andreas and Daniel Golovin. “Submodular function maximization.” In: *Tractability* 3.71-104 (2014), p. 3.

Krishnamurthy, Vikram. “Partially observed Markov decision processes”. Cambridge university press, 2016.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Proc. NeurIPS*. 2012.

Le, Hoang M, Yisong Yue, Peter Carr, and Patrick Lucey. “Coordinated multi-agent imitation learning”. In: *Proc. ICML*. 2017, pp. 1995–2003.

Le, Hoang Minh, Yisong Yue, Peter Carr, and Patrick Lucey. “Coordinated multi-agent imitation learning”. In: *Proc. ICML*. 2017, pp. 1995–2003.

LeCun, Yann, John S. Denker, and Sara A. Solla. “Optimal brain damage”. In: *Proc. NeurIPS*. 1989, pp. 598–605.

Lee, Namhoon, Thalaiyasingam Ajanthan, and Philip H. S. Torr. “SNIP: Single-shot network pruning based on connection sensitivity”. In: *Proc. ICLR*. 2019.

Lee, YC, Gary Doolen, HH Chen, GZ Sun, Tom Maxwell, and HY Lee. "Machine learning using a higher order correlation network". Tech. rep. Los Alamos National Lab (LANL), Los Alamos, NM (United States); Univ. of Maryland, College Park, MD (United States), 1986.

Letham, Benjamin, Roberto Calandra, Akshara Rai, and Eytan Bakshy. "Re-examining linear embeddings for high-dimensional Bayesian optimization". In: *Proc. NeurIPS*. 2020.

Lhaksmana, Kemas M, Yohei Murakami, and Toru Ishida. "Role-based modeling for designing agent behavior in self-organizing multi-agent systems". In: *International Journal of Software Engineering and Knowledge Engineering* 28.01 (2018), pp. 79–96.

Li, Bailin, Bowen Wu, Jiang Su, and Guangrun Wang. "EagleEye: Fast sub-net evaluation for efficient neural network pruning". In: *Proc. ECCV*. 2020, pp. 639–654.

Li, Chenghao, Tonghan Wang, Chengjie Wu, Qianchuan Zhao, Jun Yang, and Chongjie Zhang. "Celebrating diversity in shared multi-agent reinforcement learning". In: *Proc. NeurIPS*. 2021, pp. 3991–4002.

Li, Hao, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. "Pruning filters for efficient ConvNets". In: *Proc. ICLR*. 2017.

Li, Linyang, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. "BERT-ATTACK: Adversarial attack against BERT using BERT". In: *Proc. EMNLP*. 2020, pp. 6193–6202.

Li, Yao, Shengzhu Shi, Zhichang Guo, and Boying Wu. "Adversarial Training for Physics-Informed Neural Networks". In: *arXiv preprint arXiv:2310.11789* (2023).

Li, Yuxi. "Deep reinforcement learning: An overview". In: *arXiv preprint arXiv:1701.07274* (2017).

Li, Zewen, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. "A survey of convolutional neural networks: Analysis, applications, and prospects". In: *IEEE Trans. Neural Networks Learn. Syst.* 33.12 (2022), pp. 6999–7019.

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

Lin, Shaohui, Rongrong Ji, Chenqian Yan, Baochang Zhang, Liujuan Cao, Qixiang Ye, Feiyue Huang, and David Doermann. "Towards optimal structured CNN pruning via generative adversarial learning". In: *Proc. CVPR*. 2019, pp. 2790–2799.

Lin, Yiheng, Guannan Qu, Longbo Huang, and Adam Wierman. "Multi-agent reinforcement learning in stochastic networked systems". In: *Proc. ICLR*. 2017.

Linke, Cam, Nadia M. Ady, Martha White, Thomas Degris, and Adam White. "Adapting Behavior via Intrinsic Reward: A Survey and Empirical Study". In: *J. Artif. Intell. Res.* 69 (2020), pp. 1287–1332.

Liu, Junjie, Zhe Xu, Runbin Shi, Ray C. C. Cheung, and Hayden Kwok-Hay So. "Dynamic sparse training: Find efficient sparse network from scratch with trainable masked layers". In: *Proc. ICLR*. 2020.

Liu, Yong, Weixun Wang, Yujing Hu, Jianye Hao, Xingguo Chen, and Yang Gao. "Multi-agent game abstraction via graph attention neural network". In: *Proc. AAAI*. 2020, pp. 7211–7218.

Lizotte, Daniel J., Tao Wang, Michael H. Bowling, and Dale Schuurmans. "Automatic gait optimization with Gaussian process regression". In: *Proc. IJCAI*. 2007.

Louizos, Christos, Max Welling, and Diederik P. Kingma. "Learning sparse neural networks through L_0 regularization". In: *Proc. ICLR*. 2018.

Lowe, Ryan, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. "Multi-agent actor-critic for mixed cooperative-competitive environments". In: *Advances in neural information processing systems* 30 (2017).

Lu, Liang, Michelle Guo, and Steve Renals. "Knowledge distillation for small-footprint highway networks". In: *Proc. ICASSP*. 2017, pp. 4820–4824.

Luengo, David, Luca Martino, Mónica F. Bugallo, Víctor Elvira, and Simo Särkkä. "A survey of Monte Carlo methods for parameter estimation". In: *EURASIP J. Adv. Signal Process.* 2020.1 (2020), p. 25.

Lym, Sangkug, Esha Choukse, Siavash Zangeneh, Wei Wen, Sujay Sanghavi, and Mattan Erez. "PruneTrain: Fast neural network training by dynamic sparse model reconfiguration". In: *Proc. SC*. 2019, pp. 1–13.

Machado, Gabriel Resende, Eugênio Silva, and Ronaldo Ribeiro Goldschmidt. "Adversarial machine learning in image classification: A survey toward the defender's perspective". In: *ACM Comput. Surv.* 55.2 (2023), 8:1–8:38.

Magalhães, Eduardo. "On the Properties of the Hessian tensor for vector functions". In: *viXra preprint viXra:2005.0044* (2020).

Mahmood, Kaleel, Rigel Mahmood, and Marten van Dijk. "On the robustness of vision transformers to adversarial examples". In: *Proc. ICCV*. IEEE, 2021, pp. 7818–7827.

Marco, Alonso, Philipp Hennig, Jeannette Bohg, Stefan Schaal, and Sebastian Trimpe. "Automatic LQR tuning based on Gaussian process global optimization". In: *Proc. ICRA*. 2016.

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". Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/`.

Martinez-Cantin, Ruben. "Bayesian optimization with adaptive kernels for robot control". In: *Proc. ICRA*. 2017.

Matignon, Laëtitia, Guillaume J. Laurent, and Nadine Le Fort-Piat. "Independent reinforcement learners in cooperative Markov games: a survey regarding coordination problems". In: *Knowl. Eng. Rev.* 27.1 (2012).

Matthews, Alexander, Mark van der Wilk, Tom Nickson, Keisuke Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman.

"GPflow: A Gaussian process library using TensorFlow". In: *JMLR* 18.1 (2017), pp. 1–6.

Matthews, Alexander G. de G., Mark van der Wilk, Tom Nickson, Keisuke. Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman. "GPflow: A Gaussian process library using TensorFlow". In: *Journal of Machine Learning Research* 18.40 (Apr. 2017), pp. 1–6. URL: http://jmlr.org/papers/v18/16-537.html.

Matula, David W. "The largest clique size in a random graph". Department of Computer Science, Southern Methodist University Dallas, Texas, 1976.

McLeod, Mark, Stephen J. Roberts, and Michael A. Osborne. "Optimization, fast and slow: optimally switching between local and Bayesian optimization". In: *Proc. ICML*. 2018.

Merenda, Massimo, Carlo Porcaro, and Demetrio Iero. "Edge machine learning for AI-Enabled IoT devices: A Review". In: *Sensors* 20.9 (2020), p. 2533.

Micikevicius, Paulius, Sharan Narang, Jonah Alben, Gregory F. Diamos, Erich Elsen, David García, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. "Mixed precision training". In: *Proc. ICLR*. 2018.

Mittal, Sparsh and Shraiysh Vaishay. "A survey of techniques for optimizing deep learning on GPUs". In: *J. Syst. Archit.* 99 (2019).

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

Mnih, Volodymyr, 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". In: *nature* 518.7540 (2015), pp. 529–533.

Mocanu, Decebal Constantin, Elena Mocanu, Peter Stone, Phuong H. Nguyen, Madeleine Gibescu, and Antonio Liotta. "Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science". In: *Nature* 9.1 (2018), pp. 1–12.

Moerland, Thomas M., Joost Broekens, Aske Plaat, and Catholijn M. Jonker. "Model-based reinforcement learning: A survey". In: *Found. Trends Mach. Learn.* 16.1 (2023), pp. 1–118.

Molchanov, Pavlo, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. "Pruning convolutional neural networks for resource efficient inference". In: *Proc. ICLR*. 2017.

Mostafa, Hesham and Xin Wang. "Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization". In: *Proc. ICML*. 2019, pp. 4646–4655.

Mozer, Michael and Paul Smolensky. "Skeletonization: A technique for trimming the fat from a network via relevance assessment". In: *Proc. NeurIPS*. 1988, pp. 107–115.

Müller, Sarah, Alexander von Rohr, and Sebastian Trimpe. "Local policy search with Bayesian optimization". In: *Proc. NeurIPS*. 2021, pp. 20708–20720.

Mutny, Mojmir and Andreas Krause. “Efficient high dimensional bayesian optimization with additivity and quadrature fourier features”. In: *Proc. NeurIPS*. 2018, pp. 9019–9030.

Nadarajah, Saralees and Samuel Kotz. “Exact distribution of the max/min of two Gaussian random variables”. In: *Trans. VLSI* 16.2 (2008), pp. 210–212.

Nair, Vinod and Geoffrey E. Hinton. “Rectified linear units improve restricted Boltzmann machines”. In: *Proc. ICML*. 2010, pp. 807–814.

Narang, Sharan, Greg Diamos, Shubho Sengupta, and Erich Elsen. “Exploring sparsity in recurrent neural networks”. In: *Proc. ICLR*. 2017.

Nowlan, Steven J. and Geoffrey E. Hinton. “Simplifying Neural Networks by Soft Weight-Sharing”. In: *Neural Computation* 4.4 (1992), pp. 473–493.

Papavasileiou, Evgenia, Jan Cornelis, and Bart Jansen. “A systematic literature review of the successors of “neuroevolution of augmenting topologies””. In: *Evolutionary Computation* 29.1 (2021), pp. 1–73.

Papernot, Nicolas, Patrick D. McDaniel, Ananthram Swami, and Richard E. Harang. “Crafting adversarial input sequences for recurrent neural networks”. In: *Proc. MILCOM*. 2016, pp. 49–54.

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

Peng, Bei, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. “Facmac: Factored multi-

agent centralised policy gradients". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 12208–12221.

Peng, Peng, Ying Wen, Yaodong Yang, Quan Yuan, Zhenkun Tang, Haitao Long, and Jun Wang. "Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games". In: *arXiv preprint arXiv:1703.10069* (2017).

Picheny, Victor, Henry B. Moss, Léeonard Torossian, and Nicolas Durrande. "Bayesian quantile and expectile optimisation". In: *Proc. UAI*. 2022, pp. 1623–1633.

Polyak, Adam and Lior Wolf. "Channel-level acceleration of deep face representations". In: *IEEE Access* 3 (2015), pp. 2163–2175.

Prakash, Utkarsh, Aryan Chollera, Kushagra Khatwani, Prabuchandran K. J., and Tejas Bodas. "Practical first-order Bayesian optimization algorithms". In: *Proc. COMAD*. 2024, pp. 173–181.

Qian, Hong and Yang Yu. "Derivative-free reinforcement learning: a review". In: *Frontiers of Computer Science* 15.6 (2021), pp. 1–19.

Qu, Guannan, Yiheng Lin, Adam Wierman, and Na Li. "Scalable multi-agent reinforcement learning for networked systems with average reward". In: *Proc. NeurIPS*. 2020.

Qu, Guannan, Adam Wierman, and Na Li. "Scalable reinforcement learning of localized policies for multi-agent networked systems". In: *Proc. L4DC*. 2020.

Ramachandran, Prajit, Barret Zoph, and Quoc V. Le. "Searching for activation functions". In: *Proc. ICLR, Workshop*. 2018.

Rashid, Tabish, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 4295–4304.

Rasmussen, Carl Edward and Christopher K. I. Williams. "Gaussian processes for machine learning." MIT Press, 2006.

Rizk, Yara, Mariette Awad, and Edward W Tunstel. "Decision making in multiagent systems: A survey". In: *IEEE Transactions on Cognitive and Developmental Systems* 10.3 (2018), pp. 514–529.

Rohr, Alexander von, Sebastian Trimpe, Alonso Marco, Peer Fischer, and Stefano Palagi. "Gait Learning for soft microrobots controlled by light fields". In: *Proc. IROS*. 2018.

Roijers, Diederik M, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. "A survey of multi-objective sequential decision-making". In: *Journal of Artificial Intelligence Research* 48 (2013), pp. 67–113.

Rolland, Paul, Jonathan Scarlett, Ilija Bogunovic, and Volkan Cevher. "High-dimensional Bayesian optimization via additive models with overlapping groups". In: *Proc. AISTATS*. 2018, pp. 298–307.

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional networks for biomedical image segmentation". In: *Proc. MICCAI*, pp. 234–241.

Rosenblatt, Frank. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* (1958).

Ruder, Sebastian. “An overview of gradient descent optimization algorithms”. In: *arXiv preprint arXiv:1609.04747* (2016).

Rumelhart, David E, Geoffrey E Hinton, Ronald J Williams, et al. “Learning internal representations by error propagation”. 1985.

Salimans, Tim and Richard Chen. “Learning montezuma’s revenge from a single demonstration”. In: *arXiv preprint arXiv:1812.03381* (2018).

Scarlett, Jonathan. “Tight regret bounds for Bayesian optimization in one dimension”. In: *Proc. ICML*. 2018, pp. 4507–4515.

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

Seshadhri, Comandur, Tamara G Kolda, and Ali Pinar. “Community structure and scale-free collections of Erdős-Rényi graphs”. In: *Physical Review E* 85.5 (2012), p. 056109.

Shani, Guy, Joelle Pineau, and Robert Kaplow. “A survey of point-based POMDP solvers”. In: *Autonomous Agents and Multi-Agent Systems* 27 (2013), pp. 1–51.

Shao, Jianzhun, Zhiqiang Lou, Hongchang Zhang, Yuhang Jiang, Shuncheng He, and Xiangyang Ji. “Self-organized group for cooperative multi-agent reinforcement learning”. In: *Proc. NeurIPS* (2022), pp. 5711–5723.

Shazeer, Noam and Mitchell Stern. “Adafactor: Adaptive learning rates with sublinear memory cost”. In: *Proc. ICML*. 2018, pp. 4603–4611.

Shekarpaz, Simin, Mohammad Azizmalayeri, and Mohammad Hossein Rohban. “Piat: Physics informed adversarial training for solving partial differential equations”. In: *arXiv preprint arXiv:2207.06647* (2022).

Shekhar, Shubhanshu and Tara Javidi. “Significance of gradient information in Bayesian optimization”. In: *Proc. AISTATS*. 2021.

Shekhar, Shubhanshu and Tara Javidi. “Significance of gradient information in Bayesian optimization”. In: *Proc. AISTATS*. 2021, pp. 2836–2844.

Silver, David, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. “Mastering the game of Go with deep neural networks and tree search”. In: *Nature* 529.7587 (2016), pp. 484–489.

Silver, David, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin A. Riedmiller. “Deterministic policy gradient algorithms”. In: *Proc. ICML*. 2014, pp. 387–395.

Simonyan, Karen and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *Proc. ICLR*. 2015.

Simonyan, Karen and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *Proc. ICLR*. 2015.

Simsek, Özgür and Andrew G. Barto. “An intrinsic reward mechanism for efficient exploration”. In: *Proc. ICML*. 2006, pp. 833–840.

Singh, Amanpreet, Tushar Jain, and Sainbayar Sukhbaatar. "Learning when to communicate at scale in multiagent cooperative and competitive tasks". In: *Proc. ICLR*. 2019.

Singh, Satinder, Andrew G. Barto, and Nuttapong Chentanez. "Intrinsically motivated reinforcement learning". In: *Proc. NeurIPS*. 2004, pp. 1281–1288.

Skorski, Maciej. "Chain rules for hessian and higher derivatives made easy by tensor calculus". In: *arXiv preprint arXiv:1911.13292* (2019).

Snoek, Jasper, Hugo Larochelle, and Ryan P. Adams. "Practical Bayesian Optimization of Machine Learning Algorithms". In: *Proc. NeurIPS*. 2012.

Son, Kyunghwan, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. "Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 5887–5896.

Srinivas, Niranjan, Andreas Krause, Sham M. Kakade, and Matthias W. Seeger. "Gaussian process optimization in the bandit setting: No regret and experimental design". In: *Proc. ICML*. 2010.

Srivastava, Nitish, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: A simple way to prevent neural networks from overfitting". In: *JMLR* 15.1 (2014), pp. 1929–1958.

Sunehag, Peter, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. "Value-decomposition networks for cooperative multi-agent learning". In: *arXiv preprint arXiv:1706.05296* (2017).

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. “Sequence to sequence learning with neural networks”. In: *Proc. NeurIPS*. 2014, pp. 3104–3112.

Swersky, Kevin, Jasper Snoek, and Ryan Prescott Adams. “Freeze-thaw Bayesian optimization”. arXiv:1406.3896. 2014.

Tanaka, Hidenori, Daniel Kunin, Daniel L. Yamins, and Surya Ganguli. “Pruning neural networks without any data by iteratively conserving synaptic flow”. In: *Proc. NeurIPS*. 2020.

Tung, Frederick and Greg Mori. “Similarity-preserving knowledge distillation”. In: *Proc. ICCV*. 2019, pp. 1365–1374.

Ullrich, Karen, Edward Meeds, and Max Welling. “Soft weight-sharing for neural network compression”. In: *Proc. ICLR*. 2017.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. “Attention is all you need”. In: *Proc. NeurIPS*. 2017, pp. 5998–6008.

Verbraeken, Joost, Matthijs Wolting, Jonathan Katzy, Jeroen Kloppenburg, Tim Verbelen, and Jan S. Rellermeyer. “A survey on distributed machine learning”. In: *ACM Comput. Surv.* 53.2 (2021), 30:1–30:33.

Vinyals, Oriol, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander Sasha Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom Le Paine, Çaglar Gülçehre, Ziyu Wang, Tobias

Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy P. Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. "Grandmaster level in StarCraft II using multi-agent reinforcement learning". In: *Nature* 575.7782 (2019), pp. 350–354.

Vinyals, Oriol, Meire Fortunato, and Navdeep Jaitly. "Pointer networks". In: *Proc. NeurIPS*. 2015, pp. 2692–2700. URL: https://proceedings.neurips.cc/paper/2015/hash/29921001f2f04bd3baee84a12e98098f-Abstract.html.

Wang, Chaohui, Nikos Komodakis, and Nikos Paragios. "Markov Random Field modeling, inference & learning in computer vision & image understanding: A survey". In: *Comput. Vis. Image Underst.* 117.11 (2013), pp. 1610–1627.

Wang, Chaoqi, Guodong Zhang, and Roger B. Grosse. "Picking winning tickets before training by preserving gradient flow". In: *Proc. ICLR*. 2020.

Wang, Jianhao, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. "QPLEX: Duplex dueling multi-agent Q-Learning". In: *Proc. ICLR*. 2021.

Wang, Ke Alexander, Geoff Pleiss, Jacob R. Gardner, Stephen Tyree, Kilian Q. Weinberger, and Andrew Gordon Wilson. "Exact Gaussian processes on a million data points". In: *Proc. NeurIPS*. 2019, pp. 14622–14632.

Wang, Linnan, Rodrigo Fonseca, and Yuandong Tian. "Learning search space partition for black-box optimization using monte carlo tree search". In: *Proc. NeurIPS*. 2020.

Wang, Qi, Yue Ma, Kun Zhao, and Yingjie Tian. "A comprehensive survey of loss functions in machine learning". In: *Annals of Data Science* (2020), pp. 1–26.

Wang, Tonghan, Heng Dong, Victor Lesser, and Chongjie Zhang. "Roma: Multi-agent reinforcement learning with emergent roles". In: *arXiv preprint arXiv:2003.08039* (2020).

Wang, Tonghan, Tarun Gupta, Anuj Mahajan, Bei Peng, Shimon Whiteson, and Chongjie Zhang. "RODE: Learning roles to decompose multi-agent tasks". In: *Proc. ICLR*. 2021.

Wang, Yulong, Xiaolu Zhang, Lingxi Xie, Jun Zhou, Hang Su, Bo Zhang, and Xiaolin Hu. "Pruning from scratch". In: *Proc. AAAI*. 2020, pp. 12273–12280.

Watkins, Christopher JCH and Peter Dayan. "Q-learning". In: *Machine learning* 8 (1992), pp. 279–292.

Wen, Wei, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. "Learning structured sparsity in deep neural networks". In: *Proc. NeurIPS*. 2016, pp. 2074–2082.

White, Colin, Willie Neiswanger, and Yash Savani. "BANANAS: Bayesian optimization with neural architectures for neural architecture search". In: *Proc. AAAI*. 2021, pp. 10293–10301.

Wierstra, Daan, Tom Schaul, Jan Peters, and Jürgen Schmidhuber. "Fitness expectation maximization". In: *Proc. PPSN*. 2008, pp. 337–346.

Wilson, Aaron, Alan Fern, and Prasad Tadepalli. "Using trajectory data to improve Bayesian optimization for reinforcement learning". In: *JMLR* 15.1 (2014).

Wilson, James T., Viacheslav Borovitskiy, Alexander Terenin, Peter Mostowsky, and Marc Peter Deisenroth. "Efficiently sampling functions from Gaussian process posteriors". In: *Proc. ICML*. 2020, pp. 10292–10302.

Witt, Christian Schroeder de, Bei Peng, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. "Deep multi-agent reinforcement learning for decentralized continuous cooperative control". In: *arXiv preprint arXiv:2003.06709* (2020).

Wu, Jian, Matthias Poloczek, Andrew Gordon Wilson, and Peter I. Frazier. "Bayesian optimization with gradients". In: *Proc. NeurIPS*. 2017.

Wu, Zonghan, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. "A comprehensive survey on graph neural networks". In: *IEEE Trans. Neural Networks Learn. Syst.* 32.1 (2021), pp. 4–24.

Yang, Carl, Aydin Buluç, and John D. Owens. "Design principles for sparse matrix multiplication on the GPU". In: *Proc. Euro-Par*. 2018, pp. 672–687.

Yang, Li and Abdallah Shami. "On hyperparameter optimization of machine learning algorithms: Theory and practice". In: *Neurocomputing* 415 (2020), pp. 295–316.

Yang, Ling, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. "Diffusion models: A comprehensive survey of methods and applications". In: *ACM Computing Surveys* 56.4 (2023), pp. 1–39.

Yang, Yaodong, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. "Mean field multi-agent reinforcement learning". In: *Proc. ICML*. 2018.

Yim, Junho, Donggyu Joo, Ji-Hoon Bae, and Junmo Kim. "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning". In: *Proc. CVPR*. 2017, pp. 7130–7138.

Zaman, Muhammad Aneeq uz, Kaiqing Zhang, Erik Miehling, and Tamer Basar. "Reinforcement learning in non-stationary discrete-time linear-quadratic mean-field games". In: *Proc. CDC*. 2020.

Zeiler, Matthew D. "Adadelta: an adaptive learning rate method". In: *arXiv preprint arXiv:1212.5701* (2012).

Zemanian, Armen H. "Distribution theory and transform analysis: an introduction to generalized functions, with applications". 1987.

Zhang, Chen, Yu Xie, Hang Bai, Bin Yu, Weihong Li, and Yuan Gao. "A survey on federated learning". In: *Knowl. Based Syst.* 216 (2021), p. 106775.

Zhao, Xiangyu, Hanzhou Wu, and Xinpeng Zhang. "Watermarking graph neural networks by random graphs". In: *Proc. ISDFS*. 2021, pp. 1–6.

Zheng, Zeyu, Junhyuk Oh, and Satinder Singh. "On learning intrinsic rewards for policy gradient methods". In: *Proc. NeurIPS*. 2018, pp. 4649–4659.

Zhu, Changxi, Mehdi Dastani, and Shihan Wang. "A survey of multi-agent reinforcement learning with communication". In: *arXiv preprint arXiv:2203.08975* (2022).

Zoph, Barret and Quoc V. Le. "Neural architecture search with reinforcement learning". In: *Proc. ICLR*. 2017.

Zou, Andy, Zifan Wang, J Zico Kolter, and Matt Fredrikson. "Universal and transferable adversarial attacks on aligned language models". In: *arXiv preprint arXiv:2307.15043* (2023).

Zügner, Daniel, Oliver Borchert, Amir Akbarnejad, and Stephan Günnemann. "Adversarial attacks on graph neural networks: Perturbations and their patterns". In: *ACM Trans. Knowl. Discov. Data* 14.5 (2020), 57:1–57:31.

Appendix A

Appendix for Chapter 4

A.1 Saliency Function

In this work, we use a first-order Taylor series saliency function proposed by Molchanov et al. 2017. Our design (Section 4.3) remains flexible to allow usage of arbitrary saliency functions in a plug-n-play basis. We partition a DNN of L layers, where each layer ℓ contains C_ℓ convolutional filters, into a sequence of convolutional filters $[z_{\ell,c}]_{\ell=1,\ldots,L}^{c=1,\ldots,C_\ell}$. Each filter $z_{\ell,c} : \mathbb{R}^{C_{\ell-1}\times W_{\ell-1}\times H_{\ell-1}} \rightarrow \mathbb{R}^{W_\ell \times H_\ell}$ can be considered as one *network element* in $\boldsymbol{v}_T$ and $z_{\ell,c}(\mathbf{P}_{\ell-1}) \triangleq \mathcal{R}(\mathbf{W}_{\ell,c} * \mathbf{P}_{\ell-1} + b_{\ell,c})$ where $\mathbf{W}_{\ell,c} \in \mathbb{R}^{C_\ell \times O_\ell \times O'_\ell}$, $b_{\ell,c}$ are kernel weights and bias. With receptive field $O_\ell \times O'_\ell$, '$*$' represents the convolution operation, $\mathcal{R}$ is the activation function, $\mathbf{P}_{\ell-1}$ represents the output of $\boldsymbol{z}_{\ell-1} \triangleq [z_{\ell-1,c'}]_{c'=1,\ldots,C_{\ell-1}}$ with $\mathbf{P}_0$ corresponding to an input $\mathbf{x}_d \in \mathcal{X}$, and W_ℓ, H_ℓ are width and height dimensions of layer ℓ for $\ell = 1, \ldots, L$. Let $\mathcal{N}_{\boldsymbol{z}_\ell:\boldsymbol{z}_{\ell'}} \triangleq \boldsymbol{z}_{\ell'} \circ, \ldots, \circ \boldsymbol{z}_\ell$ denote a *partial* neural network of layers $[\ell, \ldots, \ell']_{1\leq\ell\leq\ell'\leq L}$. The Taylor series saliency function on the convolutional filter $z_{\ell,c}$ denoted as $s([\ell, c])$ is defined[1]:

$$s([\ell, c]) \triangleq \frac{1}{D} \sum_{d=1}^{D} \left| \frac{1}{W_\ell \times H_\ell} \sum_{j=1}^{W_\ell \times H_\ell} \frac{\partial \mathcal{L}(\mathbf{P}_\ell^{(\mathbf{x}_d)}, y_d; \mathcal{N}_{\boldsymbol{z}_{\ell+1}:\boldsymbol{z}_L})}{\partial P_{\ell,c,j}^{(\mathbf{x}_d)}} P_{\ell,c,j}^{(\mathbf{x}_d)} \right|. \tag{A.1}$$

[1]For brevity, we omit parameters $\mathcal{X}$, $\mathcal{Y}$, $\mathcal{N}_{\boldsymbol{z}_1:\boldsymbol{z}_L}$, $\mathcal{L}$.

where $\mathbf{P}_{\ell}^{(\mathbf{x}_d)}$ is the output of the partial neural network $\mathcal{N}_{\boldsymbol{z}_1:\boldsymbol{z}_\ell}$ with $\mathbf{x}_d$ as the input and $[P_{\ell,c,j}^{\mathbf{x}_d}]_{j=1,\ldots,W_\ell \times H_\ell}$ interprets the output of the cth filter in vectorized form. This function uses the first-order Taylor series approximation of $\mathcal{L}$ to approximate the change in loss if $z_{\ell,c}$ was changed to a constant 0 function. Using the above saliency definition, pruning filter $z_{\ell,c}$ corresponds to collectively zeroing $\mathbf{W}_{\ell,c}$, $b_{\ell,c}$ as well as weight parameters[2] $[\mathbf{W}_{\ell+1,c',\{:,:,c\}}]_{c'=1,\ldots,C_{\ell+1}}$ of $\boldsymbol{z}_{\ell+1}$ which utilize the output of $z_{l,c}$. This definition can be extended to elements (e.g., neurons) which output scalars by setting $W_\ell = H_\ell = 1$.

A.2 Proof of Pruning Lower Bound

We state Lemma 4 asserting the lower bound in Equation 4.4.

Lemma A.1. *Let $\boldsymbol{m}_t \in \{0,1\}^M$ then the following holds true:*

$$\begin{aligned} &\max_{\boldsymbol{m}_t} \mathbb{E}_{p(\boldsymbol{s}_{t+1}|\tilde{\mathbf{s}}_{1:t})}\Big[\rho_{t+1}(\boldsymbol{m}_t, B_{t,c} - \|\boldsymbol{m}_t\|_0, B_s)\Big] \\ &\geq \max_{\boldsymbol{m}_t} \mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\mathbf{s}}_{1:t})}[\rho_T(\boldsymbol{m}_t, B_{t,c} - (T-t)\|\boldsymbol{m}_t\|_0, B_s)]. \end{aligned} \tag{A.2}$$

Proof. To prove the above, we show a solution to the latter that can be transformed into an equivalent feasible solution to the former. Let

$$\boldsymbol{m}_t^* \triangleq \max_{\boldsymbol{m}_t} \mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\mathbf{s}}_{1:t})}[\rho_T(\boldsymbol{m}_t, B_{t,c} - (T-t)\|\boldsymbol{m}_t\|_0, B_s)].$$

Accordingly, we define a feasible solution for the former optimization problem:

$$\boldsymbol{m}_{t+1}^* = \boldsymbol{m}_{t+2}^* = \ldots = \boldsymbol{m}_T^* = \boldsymbol{m}_t^*.$$

[2]Here we use {} to distinguish indexing into a tensor from indexing into the sequence of tensors $[\mathbf{W}_{\ell+1,c'}]$.

Let the above serve as solutions to $\rho_{t+1}, \rho_{t+2}, \ldots, \rho_T$ satisfies the constraint of ρ_t in the former optimization problem:

$$\begin{aligned}
&\rho_t(\boldsymbol{m}_t^*, B_{t,c} - \|\boldsymbol{m}_t^*\|_0, B_s) \\
&= \rho_{t+1}(\boldsymbol{m}_t^*, B_{t,c} - 2\|\boldsymbol{m}_t^*\|_0, B_s) \\
&\vdots \\
&= \rho_T(\boldsymbol{m}_t^*, B_{t,c} - (T-t)\|\boldsymbol{m}_t^*\|_0, B_s)
\end{aligned}$$

which completes the proof as the maximization of the former optimization can only be greater or equal to a feasible solution. □

A.3 Proof of Lemma 4.1

We restate Lemma 4.1 for clarity.

Lemma 4.1. *Let* $\boldsymbol{m}'$, $\boldsymbol{m}'' \in \{0,1\}^M$, *and* $e^{(a)}$ *be an arbitrary* M*-dimensional one hot vector with* $1 \le a \le M$ *with* $P(\boldsymbol{m}) \triangleq \mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\boldsymbol{s}}_{1:t})}[\hat{\rho}_T(\boldsymbol{m}, B_s)]$. *We have* $P(\boldsymbol{m}' \vee e^{(a)}) - P(\boldsymbol{m}') \ge P(\boldsymbol{m}'' \vee e^{(a)}) - P(\boldsymbol{m}'')$ *for any* $\boldsymbol{m}' \dot{\le} \boldsymbol{m}''$ *when* $\boldsymbol{m}' \wedge e^{(a)} = 0^M$, *and* $\boldsymbol{m}'' \wedge e^{(a)} = 0^M$.

Proof. According to Equation 4.2,

$$\mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\boldsymbol{s}}_{1:t})}[\hat{\rho}_T(\boldsymbol{m}, B_s)] = \mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\boldsymbol{s}}_{1:t})}\left[\max_{\boldsymbol{m}_T}\left[\boldsymbol{m}_T \cdot \tilde{\boldsymbol{s}_T} \;\text{ s.t. }\; \|\boldsymbol{m}_T\|_0 \le B_s, \boldsymbol{m}_T \dot{\le} \boldsymbol{m}\right]\right]$$

Let $\alpha(\boldsymbol{m}) \triangleq \arg\max_{\boldsymbol{m}_T}\left[\boldsymbol{m}_T \cdot \tilde{\boldsymbol{s}_T} \;\text{ s.t. }\; \|\boldsymbol{m}_T\|_0 \le B_s, \boldsymbol{m}_T \dot{\le} \boldsymbol{m}\right]$ return the optimized mask $\boldsymbol{m}_T$ given any $\boldsymbol{m}$, $\Lambda_{\boldsymbol{m}} \triangleq \min(\alpha(\boldsymbol{m}) \odot \boldsymbol{s}_T)$ be the minimal saliency of the

network elements selected at iteration T for $P(\boldsymbol{m})$. Then, we have

$$\begin{aligned} P(\boldsymbol{m} \vee e^{(a)}) &= \mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\boldsymbol{s}}_{1:t})}\left[\hat{\rho}_T(\boldsymbol{m} \vee e^{(a)}, B_s)\right] \\ &= \mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\boldsymbol{s}}_{1:t})}\left[\hat{\rho}_T(\boldsymbol{m}, B_s) - \Lambda_{\boldsymbol{m}} + \max(s_T^a, \Lambda_{\boldsymbol{m}})\right] \end{aligned}$$

The second equality is due to the fact that the network element v_T^a would only replace the lowest included element in $\boldsymbol{m}_T$ in order to maximize the objective. Then,

$$\begin{aligned} & P(\boldsymbol{m} \vee e^{(a)}) - P(\boldsymbol{m}) \\ &= \mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\boldsymbol{s}}_{1:t})}\left[\hat{\rho}_T(\boldsymbol{m}, B_s) - \Lambda_{\boldsymbol{m}} + \max(s_T^a, \Lambda_{\boldsymbol{m}})\right] - \mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\boldsymbol{s}}_{1:t})}\left[\hat{\rho}_T(\boldsymbol{m}, B_s)\right] \\ &= \mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\boldsymbol{s}}_{1:t})}\left[-\Lambda_{\boldsymbol{m}} + \max(s_T^a, \Lambda_{\boldsymbol{m}})\right] \\ &= \mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\boldsymbol{s}}_{1:t})}\left[\max(s_T^a - \Lambda_{\boldsymbol{m}}, 0)\right] \end{aligned} \tag{A.3}$$

Given $\boldsymbol{m}' \dot{\leq} \boldsymbol{m}''$, we have $\Lambda_{m'} \leq \Lambda_{m''}$ since $\boldsymbol{m}_T \dot{\leq} \boldsymbol{m}$ in $\alpha(\boldsymbol{m}')$ is a tighter constraint than that in $\alpha(\boldsymbol{m}'')$. Consequently, we can get $s_t^a - \Lambda_{m'} \geq s_t^a - \Lambda_{m''}$, and thus

$$\left[P(\boldsymbol{m}' \vee e^{(a)}) - P(\boldsymbol{m}')\right] \geq \left[P(\boldsymbol{m}'' \vee e^{(a)}) - P(\boldsymbol{m}'')\right] .$$

□

A.4 Proof of Lemma 4.2

We restate Lemma 4.2 for clarity.

Lemma 4.2. *Let $\boldsymbol{e}^{(i)}$ be a M-dimensional one-hot vectors with the i^{th} element being 1. $\forall\ 1 \leq a, b \leq M, \boldsymbol{m} \in \{0,1\}^M$ s.t. $\boldsymbol{m} \wedge (\boldsymbol{e}^{(a)} \vee \boldsymbol{e}^{(b)}) = 0^M$. Given a matrix $\tilde{\mathbf{s}}_{1:t}$ of observed saliency measurements, if $\mu_{T|1:t}^a \geq \mu_{T|1:t}^b$ and $\mu_{T|1:t}^a \geq 0$, then*

$$\mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\mathbf{s}}_{1:t})}[\hat{\rho}_T(\boldsymbol{m} \vee \boldsymbol{e}^{(b)})] - \mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\mathbf{s}}_{1:t})}[\hat{\rho}_T(\boldsymbol{m} \vee \boldsymbol{e}^{(a)})] \leq \mu_{T|1:t}^b\ \Phi(\nu/\theta) + \theta\ \phi(\nu/\theta)$$

where $\theta \triangleq \sqrt{\sigma^{aa}_{T|1:t} + \sigma^{bb}_{T|1:t} - 2\sigma^{ab}_{T|1:t}}$, $\nu \triangleq \mu^{b}_{T|1:t} - \mu^{a}_{T|1:t}$, *and* Φ *and* ϕ *are standard normal CDF and PDF, respectively.*

To prove this Lemma, we prove the following first:

Lemma A.1. $\mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\mathbf{s}}_{1:t})}\left[\hat{\rho}_T(\boldsymbol{m} \vee \boldsymbol{e}^{(b)})\right] - \mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\mathbf{s}}_{1:t})}\left[\hat{\rho}_T(\boldsymbol{m} \vee \boldsymbol{e}^{(a)})\right] \leq \mathbb{E}[\max(s^b_T - s^a_T, 0)]$.

Proof. Due to Equation A.3, we have

$$
\begin{aligned}
&\mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\mathbf{s}}_{1:t})}\left[\hat{\rho}_T(\boldsymbol{m} \vee \boldsymbol{e}^{(b)})\right] - \mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\mathbf{s}}_{1:t})}\left[\hat{\rho}_T(\boldsymbol{m} \vee \boldsymbol{e}^{(a)})\right] \\
&= P(\boldsymbol{m} \vee e^{(b)}) - P(\boldsymbol{m}) - (P(\boldsymbol{m} \vee e^{(a)}) - P(\boldsymbol{m})) \\
&= \mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\boldsymbol{s}}_{1:t})}\left[\max(s^b_T - \Lambda_{\boldsymbol{m}}, 0)\right] - \mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\boldsymbol{s}}_{1:t})}\left[\max(s^a_T - \Lambda_{\boldsymbol{m}}, 0)\right] \\
&= \mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\mathbf{s}}_{1:t})}\left[\max(s^b_T - \Lambda_{\boldsymbol{m}}, 0) - \max(s^a_T - \Lambda_{\boldsymbol{m}}, 0)\right] \qquad \text{(A.4)} \\
&= \mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\mathbf{s}}_{1:t})}\left[\max(s^b_T - s^a_T, \Lambda_{\boldsymbol{m}} - s^a_T) - \max(0, \Lambda_{\boldsymbol{m}} - s^a_T)\right] \qquad \text{(A.5)} \\
&\leq \mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\mathbf{s}}_{1:t})}\left[\max(s^b_T - s^a_T, 0)\right] \qquad \text{(A.6)}
\end{aligned}
$$

The equality Equation A.5 is achieved by adding $\Lambda_{\boldsymbol{m}} - s^a_T$ in each term of the two max functions in Equation A.4. The inequality Equation A.6 can be proved by considering the following two cases:

If $\Lambda_{\boldsymbol{m}} - s^a_T \geq 0$, then

$$
\begin{aligned}
&\max(s^b_T - s^a_T, \Lambda_{\boldsymbol{m}} - s^a_T) - \max(0, \Lambda_{\boldsymbol{m}} - s^a_T) \\
&= \max(s^b_T - s^a_T, \Lambda_{\boldsymbol{m}} - s^a_T) - (\Lambda_{\boldsymbol{m}} - s^a_T) \\
&= \max(s^b_T - s^a_T - (\Lambda_{\boldsymbol{m}} - s^a_T), 0) \\
&\leq \max(s^b_T - s^a_T, 0) \ .
\end{aligned}
$$

If $\Lambda_{\boldsymbol{m}} - s_T^a < 0$, then

$$\begin{aligned}
&\max(s_T^b - s_T^a, \Lambda_{\boldsymbol{m}} - s_T^a) - \max(0, \Lambda_{\boldsymbol{m}} - s_T^a) \\
&= \max(s_T^b - s_T^a, \Lambda_{\boldsymbol{m}} - s_T^a) \\
&\leq \max(s_T^b - s_T^a, 0) \ .
\end{aligned}$$

□

Next we utilize a well known bound regarding the maximum of two Gaussian random variables (Nadarajah and Kotz 2008), which we restate:

Lemma A.2. *Let s^a, s^b be Gaussian random variables with means μ^a, μ^b and standard deviations σ^a, σ^b, then $\mathbb{E}[\max(s^a, s^b)] \leq \mu^a \Phi\Big(\frac{\mu^b - \mu^a}{\theta}\Big) + \mu^b \Phi\Big(\frac{\mu^b - \mu^a}{\theta}\Big) + \theta\phi\Big(\frac{\mu^b - \mu^a}{\theta}\Big)$ where $\theta \triangleq \sqrt{[\sigma^b]^2 + [\sigma^a]^2 - 2cov(s^b, s^a)}$ and Φ, ϕ are standard normal CDF and PDF respectively.*

Then,

$$\begin{aligned}
&\mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\mathbf{s}}_{1:t})}[\max(s_T^b - s_T^a, 0)] \\
&= \mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\mathbf{s}}_{1:t})}[\max(s_T^b, s_T^a)] - \mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\mathbf{s}}_{1:t})}[s_T^a] \\
&\leq (\mu_{T|1:t}^b + \mu_{T|1:t}^a)\Phi\Big(\frac{\mu_{T|1:t}^b - \mu_{T|1:t}^a}{\theta}\Big) + \theta\phi\Big(\frac{\mu_{T|1:t}^b \mu_{T|1:t}^a}{\theta}\Big) - \mu_{T|1:t}^a \\
&= \mu_{T|1:t}^b \Phi\Big(\frac{\mu_{T|1:t}^b - \mu_{T|1:t}^a}{\theta}\Big) + \theta\phi\Big(\frac{\mu_{T|1:t}^b \mu_{T|1:t}^a}{\theta}\Big) + \mu_{T|1:t}^a \left(\Phi\left(\frac{\mu_{T|1:t}^b - \mu_{T|1:t}^a}{\theta}\right) - 1\right) \\
&\leq \mu_{T|1:t}^b \Phi\Big(\frac{\mu_{T|1:t}^b - \mu_{T|1:t}^a}{\theta}\Big) + \theta\phi\Big(\frac{\mu_{T|1:t}^b - \mu_{T|1:t}^a}{\theta}\Big)
\end{aligned}$$

The first inequality follows from Lemma A.2. The second inequality is due to $\Phi\Big(\frac{\mu_{T|1:t}^b - \mu_{T|1:t}^a}{\theta}\Big) \leq 1$ and $\mu_{T|1:t}^a \geq 0$.

A.5 Proof of Lemma 4.3

We restate Lemma 4.3 for clarity.

Lemma 4.3. *Let $\boldsymbol{e}^{(*)}$ represent a pruned element at time t with the highest predictive mean $\mu^*_{T|1:t} \geq 0$. Given an arbitrary pruned element $\boldsymbol{e}^{(a)}$ at time t, then for all $\delta \in (0,1)$, the following holds:*

$$p\left(\hat{\rho}_T(\boldsymbol{e}^{(a)} \vee \boldsymbol{m}_t, B_s) - \hat{\rho}_T(\boldsymbol{m}_t, B_s) < \frac{\lambda_t}{\delta}(T-t+\epsilon)\right) > 1-\delta$$

where $\epsilon \triangleq \lambda_t^{-1}\left[\mu^a_{T|1:t}\Phi(\nu/\theta) + \theta\ \phi(\nu/\theta)\right]$ with $\theta \triangleq \left(\sigma^{**}_{T|1:t} + \sigma^{aa}_{T|1:t} - 2\sigma^{*a}_{T|1:t}\right)^{1/2}$, and $\nu \triangleq \mu^a_{T|1:t} - \mu^*_{T|1:t}$.

Proof. The proof follows as a consequence of Lemma 4.2 and Markov inequality. By definition of $e^{(*)}$ being a *pruned* element with the highest $\mu^*_{T|1:t}$ according to Algorithm 1 Line 15:

$$\Delta(*, \boldsymbol{m}_t, \tilde{\mathbf{s}}_{1:t}, B_s) \leq \lambda_t(T-t).$$

By substituting the definition of Δ:

$$\mathbb{E}_{p(\boldsymbol{s}_T|\tilde{\mathbf{s}}_{1:t})}[\hat{\rho}_T(\boldsymbol{m}_t \vee \boldsymbol{e}^{(*)}) - \hat{\rho}_T(\boldsymbol{m}_t)] \leq \lambda_t(T-t). \tag{A.7}$$

Consequently, as $\mu^*_{T|1:t} \geq \mu^a_{T|1:t}$, we can apply Lemma 4.2 and achieve:

$$\begin{aligned}
&\mathbb{E}[\hat{\rho}_T(\boldsymbol{m}_t \vee \boldsymbol{e}^{(a)}) - \hat{\rho}_T(\boldsymbol{m}_t)] \\
=&\mathbb{E}[\hat{\rho}_T(\boldsymbol{m}_t \vee \boldsymbol{e}^{(*)})] - \mathbb{E}[\hat{\rho}_T(\boldsymbol{m}_t)] + \mathbb{E}[\hat{\rho}_T(\boldsymbol{m}_t \vee \boldsymbol{e}^{(a)})] - \mathbb{E}[\hat{\rho}_T(\boldsymbol{m}_t \vee \boldsymbol{e}^{(*)})] \\
=&\mathbb{E}[\hat{\rho}_T(\boldsymbol{m}_t \vee \boldsymbol{e}^{(*)}) - \hat{\rho}_T(\boldsymbol{m}_t)] + \mathbb{E}[\hat{\rho}_T(\boldsymbol{m}_t \vee \boldsymbol{e}^{(a)})] - \mathbb{E}[\hat{\rho}_T(\boldsymbol{m}_t \vee \boldsymbol{e}^{(*)})] \\
\leq&\lambda_t(T-t) + \lambda_t\epsilon \\
=&\lambda_t(T-t+\epsilon)
\end{aligned}$$

Table A.1: Performance (standard error) of the tested algorithms with varying inference FLOPs (percentage of pruned FLOPs) for CIFAR-10 and CIFAR-100 on the VGG-16 model. Unpruned baseline train and inference FLOPs are 8.2e15 and 6.6e8, respectively.

	CIFAR-10 (VGG-16)							
	26.5M Inference FLOPs (96.0%)		6.68M Inference FLOPs (98.9%)		1.72M Inference FLOPs (99.7%)		414K Inference FLOPs (99.9%)	
	Val Acc	Train FLOPs	Val Acc	Train FLOPs	Val Acc	Train FLOPs	Val Acc	Train FLOPs
Random	82.3(0.7)%	3.3e14	71.8(0.5)%	8.2e13	49.2(3.5)%	2.1e13	26.5(1.8)%	5.1e12
SNIP	82.8(0.8)%	3.3e14	70.2(0.7)%	8.2e13	49.9(1.2)%	2.1e13	26.4(1.5)%	5.1e12
GraSP	82.7(0.4)%	3.3e14	71.6(0.1)%	8.2e13	46.2(2.1)%	2.1e13	19.4(3.9)%	5.1e12
BEP 1×10^{-2}	83.1(0.3)%	1.1e15(1.9e11)	69.9(1.0)%	9.3e14(4.6e10)	51.9(1.1)%	8.7e14(1.2e10)	29.5(0.5)%	8.6e14(1.2e10)
BEP 1×10^{-4}	**84.1**(**0.2**)%	1.2e15(3.2e11)	**72.2**(**0.4**)%	9.4e14(4.1e10)	50.1(0.9)%	8.8e14(2.0e10)	**31.8**(**0.8**)%	8.6e14(9.6e9)
BEP 1×10^{-7}	83.7(0.2)%	1.3e15(3.4e11)	70.7(0.7)%	9.5e14(1.7e12)	**53.1**(**1.4**)%	8.8e14(2.8e11)	27.9(2.2)%	8.6e14(8.8e10)
	CIFAR-100 (VGG-16)							
	60.2M Inference FLOPs (90.9%)		26.6M Inference FLOPs (95.9%)		6.76M Inference FLOPs (98.9%)		1.76M Inference FLOPs (99.7%)	
	Val Acc	Train FLOPs	Val Acc	Train FLOPs	Val Acc	Train FLOPs	Val Acc	Train FLOPs
Random	55.9(0.2)%	7.4e14	47.3(0.7)%	3.4e14	25.7(1.4)%	8.3e13	7.8(0.5)%	2.2e13
SNIP	54.9(0.4)%	7.4e14	45.7(1.2)%	3.4e14	22.1(0.6)%	8.3e13	6.2(0.4)%	2.2e13
GraSP	54.1(0.6)%	7.4e14	46.7(0.01)%	3.4e14	23.3(0.6)%	8.3e13	6.9(0.7)%	2.2e13
BEP 1×10^{-2}	55.2(0.5)%	1.5e15(1.3e11)	46.2(0.1)%	1.1e15(2.6e11)	26.3(1.1)%	9.3e14(7.0e10)	8.6(0.4)%	8.7e14(1.2e10)
BEP 1×10^{-4}	55.2(0.6)%	1.7e15(1.4e13)	46.4(0.6)%	1.2e15(3.7e12)	28.0(0.2)%	9.4e14(4.6e11)	11.4(0.6)%	8.8e14(2.1e11)
BEP 1×10^{-7}	**56.0**(**0.0**)%	1.8e15(1.8e12)	**47.3**(**0.1**)%	1.3e15(2.4e12)	**28.4**(**0.5**)%	9.6e14(6.9e11)	**11.6**(**0.2**)%	8.8e14(2.9e11)

where the inequality is due to Equation A.7 and Lemma 4.2. The proof is complete by applying Markov's inequality:

$$
\begin{aligned}
& p\left(\hat{\rho}_T(\boldsymbol{m}_t \vee \boldsymbol{e}^{(a)}) - \hat{\rho}_T(\boldsymbol{m}_t) \geq \frac{\lambda_t}{\delta}(T-t+\epsilon)\right) \\
\leq & \frac{\mathbb{E}[\hat{\rho}_T(\boldsymbol{m}_t \vee \boldsymbol{e}^{(a)}) - \hat{\rho}_T(\boldsymbol{m}_t)]}{\lambda_t(T-t+\epsilon)/\delta} \leq \frac{\lambda_t(T-t+\epsilon)}{\lambda_t(T-t+\epsilon)/\delta} = \delta \ .
\end{aligned}
$$

Observing the negation of the above yields the desired result. □

A.6 Additional Experiments

To verify the robustness of our approach, we repeat our CIFAR-10 and CIFAR-100 early pruning experiments on the VGG-16 architecture comparing against the most competetive baselines (SNIP and GraSP). This is presented in Table A.1.

A.7 Experimental Details

A.7.1 Experimental Details

To train our CIFAR-10 and CIFAR-100 models we used an Adam optimizer (Kingma and Ba 2015) with an initial learning rate of 0.001. The learning rate used an exponential decay of $k = 0.985$, and a batch size of 32 was used. Training was paused three times evenly spaced per epoch. During this pause, we collected saliency measurements using 40% of the training dataset. This instrumentation subset was randomly select from the training dataset at initialization, and remained constant throughout the training procedure. We performed data preprocessing of saliency evaluations into a standardized $[0, 10]$ range.[3] We used Equation A.1 to measure saliency of neurons/convolutional filters. For the convolutional layers we used 12 latent MOGP functions. For the dense layer we used 4 latent MOGP functions.

For our ResNet-50 model we used an SGD with Momentum optimizer with an initial learning rate of 0.1. The learning rate was divided by ten at $t = [30, 60, 80]$ epochs. We collected saliency data every 5 SGD iterations, and averaged them into buckets corresponding to 625 SGD iterations to form our dataset. We used a minimum of 10 latent functions per MOGP, however this was dynamically increased if the model couldn't fit the data up to a maximum of 15. We used these hyperparameter settings for the VGG-16 architecture for CIFAR-10 and CIFAR-100 experiments. In our VGG-16 experiments, we also used BatchNormalization to reduce overfitting.

[3] Generally, saliency evaluations are relatively small (≤ 0.01), which leads to poor fitting models or positive log-likelihood. Precise details of our data preprocessing is in Appendix A.7.3.

We sampled 10K points from our MOGP model to estimate $\Delta(\cdot)$ for CIFAR-10/CIFAR-100. For ResNet we sampled 15K points. We repeated experiments 5 times for reporting accuracy on CIFAR-10/CIFAR-100.

A.7.2 Pruning on ResNet

ResNet architecture is composed of a sequence of residual units: $Z_\ell \triangleq \mathcal{F}(\mathbf{P}_{\ell-1}) + \mathbf{P}_{\ell-1}$, where $\mathbf{P}_{\ell-1}$ is the output of the previous residual unit $Z_{\ell-1}$ and '+' denotes element-wise addition. Internally, $\mathcal{F}$ is typically implemented as three stacked convolutional layers: $\mathcal{F}(\mathbf{P}_{\ell-1}) \triangleq [z_{\ell_3} \circ z_{\ell_2} \circ z_{\ell_1}](\mathbf{P}_{\ell-1})$ where z_{ℓ_1}, z_{ℓ_2}, z_{ℓ_3} are convolutional layers. Within this setting we consider convolutional filter pruning. Although z_{ℓ_1}, z_{ℓ_2} may be pruned using the procedure described earlier. Pruning z_{ℓ_3} requires a different procedure. Due to the direct addition of $\mathbf{P}_{\ell-1}$ to $\mathcal{F}(\mathbf{P}_{\ell-1})$, the output dimensions of $Z_{\ell-1}$ and z_{ℓ_3} must match exactly. Thus a ResNet architecture consists of sequences of residual units of length B with matching input/output dimensions: $\zeta \triangleq [Z_\ell]_{\ell=1,\ldots,B}$, s.t. $dim(\mathbf{P}_1) = dim(\mathbf{P}_2) = \ldots = dim(\mathbf{P}_B)$. We propose *group pruning* of layers $[z_{\ell_3}]_{\ell=1,\ldots,B}$ where filters are removed from all z_{ℓ_3} in a residual unit sequence in tandem. We define $\boldsymbol{s}([\zeta, c]) \triangleq \sum_{\ell=1}^{B} s([\ell_3, c])$, where $s(\cdot)$ is defined for convolutional layers as in Equation A.1. To prune the channel c from ζ, we prune it from each layer in $[z_{\ell_3}]_{\ell=1,\ldots,B}$. Typically we pruned sequence channels less aggressively than convolutional filters as these channels feed into several convolutional layers.

We group pruned less aggressively as residual unit channels feed into a large number of residual units, thus making aggressive pruning likely to degrade performance.

A.7.3 Data Preprocessing

Our chief goal in this work is to speed up training of large-scale DNNs such as ResNet (K. He et al. 2016a; K. He et al. 2016b) on the ImageNet dataset. Pruning ResNet requires a careful definition of network element saliency to allow pruning of all layers. ResNet contains long sequences of *residual units* with matching number of input/output channels. The inputs of residual units are connected with *shortcut connections* (i.e., through addition) to the output of the residual unit. Due to shortcut connections, this structure requires that within a sequence of residual units, the number of inputs/output channels of all residual units must match exactly. This requires *group pruning* of residual unit channels for a sequence of residual units, where group pruning an output channel of a residual unit sequence requires pruning it from the inputs/outputs of all residual units within the sequence.

We followed the same data preprocessing procedure for both our small scale and ImageNet experiments. To standardize the saliency measurements for a training dataset $\tilde{\boldsymbol{s}}_{1:t}$ in our modeling experiments we clip them between 0 and an upper bound computed as follows: $ub \triangleq percentile(\tilde{\boldsymbol{s}}_{1:t}, 95) \times 1.3$. This procedure removes outliers. We used 1.3 as a multiplier, as this upper bound is used to transform test dataset as well, which may have higher saliency evaluations.

After clipping the training data, we perform a trend check for each element v^a by fitting a Linear Regression model to the data $\tilde{\boldsymbol{s}}^a_{1:t}$. For $\tilde{\boldsymbol{s}}^a_{1:t}$ with an increasing trend (i.e., the linear regression model has positive slope) we perform the transformation $\tilde{\boldsymbol{s}}^a_{1:t} = ub - \tilde{\boldsymbol{s}}^a_{1:t}$. The reasoning behind this is that the exponential kernel

strongly prefers *decaying* curves. After this preprocessing, we scale up the saliency measurements to a $[0, 10]$ range: $\tilde{\boldsymbol{s}}_{1:t} = \tilde{\boldsymbol{s}}_{1:t} \times 10$. We found that without scaling to larger values, log-likelihood of our models demonstrated extremely high positive values due to small values of unscaled saliency measurements.

We transform the test data in our modeling experiments $\tilde{\boldsymbol{s}}_{t+1:T}$ with the same procedure using the same ub and per-element v^a regression models as computed by the training data. We measure log-likelihood after this transformation for both the test dataset in our small scale experiments.

During the BEP Algorithm, the same steps are followed, however we inverse the trend check transformation ($\tilde{\boldsymbol{s}}^a_{1:t} = ub - \tilde{\boldsymbol{s}}^a_{1:t}$) on the predicted MOGP distribution of $\boldsymbol{s}_T$ prior to sampling for estimation of $\Delta(\cdot)$.

Appendix B

Appendix for Chapter 5

B.1 Experimental Details

We used Trieste (Berkeley et al. 2022), Tensorflow (Martín Abadi et al. 2015), and GPFLow (A. G. d. G. Matthews et al. 2017) to build our work and perform comparisons using MushroomRL (D'Eramo et al. 2021), MultiagentMuJoCo (Witt et al. 2020), OpenAI Gym (Brockman et al. 2016), and Multi-agent Particle environment (Lowe et al. 2017). When comparing with related work, we used neural network policies of equivalent size. All of our tested policies are < 500 parameters, however the XL models are constructed using 3 layers of 400 neurons each.

To estimate the Hessian, we used the Hessian-Vector product approximation. We relaxed the discrete portions of our HOM policy into differentiable continuous approximation for this phase using the Sinkhorn-Knopp algorithm for the Role Assignment phase. For role interaction network connectivity, we used a sigmoid to create differentiable "soft" edges between each role. We pragmatically kept all detected edges in the Hessian while maintaining computational feasibility. We observed that our approach could support up to 1500 edges in the dependency graph

prior to experiencing computational intractability. We used the Matern-$\frac{5}{2}$ as the base kernel in all our models.

B.1.1 Ablation and Investigation

In the ablation, we perform experiments on MultiagentMuJoCo with environments Multiagent Ant with 6 segments, Multiagent Swimmer with 6 segments, Predator Prey with 3 predators, and Heterogeneous Predator Prey with 3 predators. In the Predator Prey environment, multiple predators must work together to capture faster and more agile prey. In Heterogeneous Predator Prey, each Predator has differing capabilities of speed and acceleration. This modification is challenging as a policy must not only coordinate between the Predators, but roles based specialization must be considered given the heterogeneous nature of each predator's capabilities.

To generate Fig. 5.6, we examined policy for Multiagent Ant with 6 agents for the role based policy specialization. The policy modulation plots were generated by examining the PredPrey and Het. PredPrey environments respectively.

B.1.2 Comparison with MARL

For the MARL setting, we compare against MADDPG (Lowe et al. 2017), FACMAC (B. Peng et al. 2021), COMIX (B. Peng et al. 2021), RODE (T. Wang, Gupta, et al. 2021) and CDS (C. Li et al. 2021) using QPLEX (Jianhao Wang et al. 2021) as a base algorithm. We also compare against Comm-MARL approaches SOG (Shao et al. 2022), and G2ANet (Y. Liu et al. 2020). RODE and QPLEX are limited to discrete environments, thus we are unable to provide comparisons on continuous action space tasks such as Multiagent Ant or Multiagent Swimmer. All MARL

environments were trained for $2,000,000$ timesteps. The neural network policies were 3-layers each with 15 neurons per layer, and were greater than or equal to the size of the compared HOM policy. For Actor-Critic approaches, we did not reduce the size or expressivity of the critic. All used hyperparameters and Algorithmic configurations were as advised by the authors of the work.

In the MARL setting we use Multiagent Ant, Multiagent Swimmer, Predator-Prey, Heterogeneous Predator-Prey. Multiagent Ant, and Multiagent Swimmer are MuJoCo locomotion tasks where each agent controls a segment of an Ant or Swimmer. Predator-Prey (PredPrey N) environment is a cooperative environment where N of agents work together to chase and capture prey agents. In Heterogeneous Predator Prey, each Predator has differing capabilities of speed and acceleration. This modification is challenging as a policy must not only coordinate between the Predators, but roles based specialization must be considered given the heterogeneous nature of each predator's capabilities. We also validated related work on the drone delivery task under which a drone swarm of N agents (Drone Delivery-N) must complete deliveries of varying distances while avoiding collisions and conserving fuel. The code of which is available in supplementary materials and will be open sourced.

We used batching (Picheny et al. 2022) in our comparisons with MARL to allow for a large number of iterations of BO. We used a batch size of 15 in our comparison experiments. In this setting, all MuJoCo environments use the default epoch (total number of interactions with the environment for computing reward) length of 1000, for Predator-Prey environments, epoch length was 25, for Drone Delivery environment, epoch length was 150.

B.1.3 RL and MARL under Malformed Reward

For single agent RL we compared against SAC (Haarnoja et al. 2018), PPO (Schulman et al. 2017), TD3 (Fujimoto, Hoof, and Meger 2018), and DDPG (Lillicrap et al. 2015) as well as an algorithm using intrinsic motivation (Zheng, Oh, and S. Singh 2018). In single agent setting, we trained related work for $200,000$ timesteps. In the MARL setting, we trained for $2,000,000$ timesteps. In both single-agent setting and multi-agent setting all policy networks for both HA-GP-UCB and related work was 3 layers of 10 neurons each. The tested environments were standard OpenAI Gym benchmarks of Ant, Hopper, Swimmer, and Walker2D.

In the MARL setting we compared against COVDN (B. Peng et al. 2021), COMIX, FACMAC, and MADDPG. Comparisons were not possible against other approaches as these do not support continuous action environments and are restricted to discrete action spaces.

For all environments and algorithms, we used the recommended hyperparameter settings as defined by the authors.

B.1.4 Comparison with HDBO Algorithms

For this comparison, we compared with several related works in HDBO. We compared with TurBO (Eriksson et al. 2019b), Alebo (Letham et al. 2020), TreeBO (E. Han, Arora, and Scarlett 2021a), LineBO (Kirschner et al. 2019), and a recent variant of BO for policy search, GIBO (Müller, Rohr, and Trimpe 2021).

For computational efficiency, the epoch length for MuJoCo environments was reduced to 500.

B.1.5 Drone Delivery Task

The experimental details follow that of comparisons with MARL.

B.1.6 Compute

All experiments were performed on commodity CPU and GPUs. Each experimental setting took no more than 2 days to complete on a single GPU.

Table B.1: Policy model sizes. Unfilled entries mean this environment was not considered during validation.

	Ant-v3	Hopper-v3	Swimmer-v3	Walker2d-v3	Ant-v3 (MARL)	Hopper-v3 (MARL)	Swimmer-v3 (MARL)	Walker2d-v3 (MARL)
RL (Single Agent)	478	263	222	356				
MARL (CTDE)					310	267	267	353
HA-GP-UCB (Single Agent)	478	263	222	356				
HA-GP-UCB (CTDE)					310	267	267	353

Table B.2: Policy model sizes. Unfilled entries mean this environment was not considered during validation.

	Multiagent-Swimmer 4	Multiagent-Swimmer 8	Multiagent-Swimmer 12	Multiagent-Ant 8	Multiagent-Ant 12	Multiagent-Ant 16	PredPrey 6	PredPrey 9	PredPrey 15	Het. PredPrey 6	Het. PredPrey 9	Het. PredPrey 15
MARL (CTDE)	267	267	267	396	396	396	478	478	478	478	478	478
HA-GP-UCB (CTDE)	267	267	267	396	396	396	478	478	478	478	478	478
HA-GP-UCB (MM)	216	244	244	406	434	434	373	373	393	373	393	393

B.1.7 Policy Sizes

We list the policy sizes of our models in Table B.1 and B.2.

Of note is in each environment, the compared against policy of RL or MARL is greater than or equal to in size vs. the policy optimized by HA-GP-UCB.

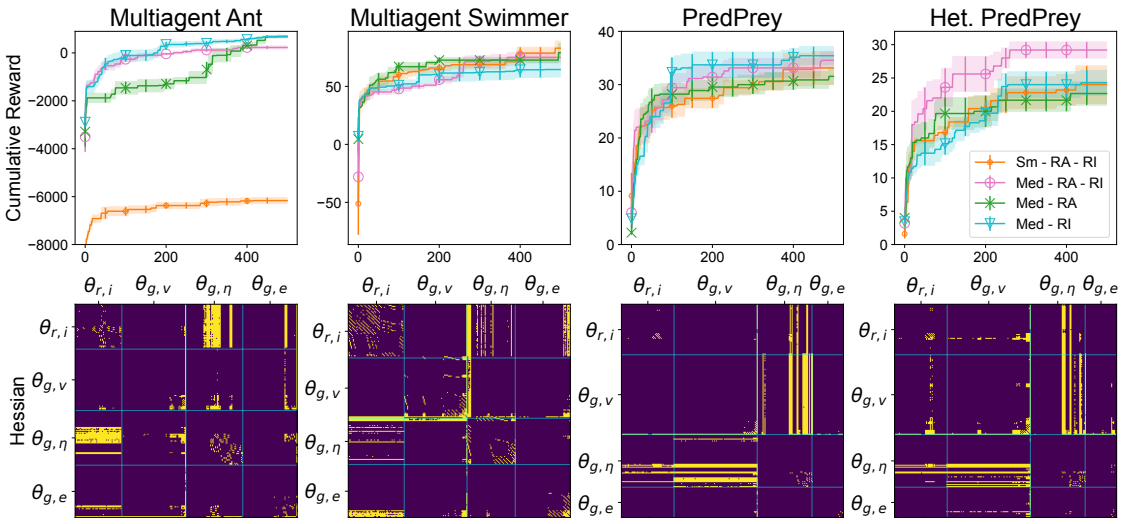

Figure B.1: Ablation study. Training curves of HA-GP-UCB and its ablated variants on different multi-agent environments.

B.2 Additional Experiments

B.2.1 Ablation

We present an expanded version of Fig. 5.3 in Fig. B.1 including the ablation for Multiagent Swimmer. Multiagent Swimmer shows similar behavior as the simpler task Multiagent Ant, with stronger block-diagonal Hessian structure.

B.2.2 Comparison with MARL

We present an expanded version of Fig. 5.5 in Fig. B.2 including the results for Multiagent-Ant and Multiagent-Swimmer. We observe that in this relatively uncomplicated task not well-suited for our approach with dense reward, our HOM approach shows comparable performance to MARL approaches and far outperforms HA-GP-UCB (CTDE). This shows the overall value of our HOM approach.

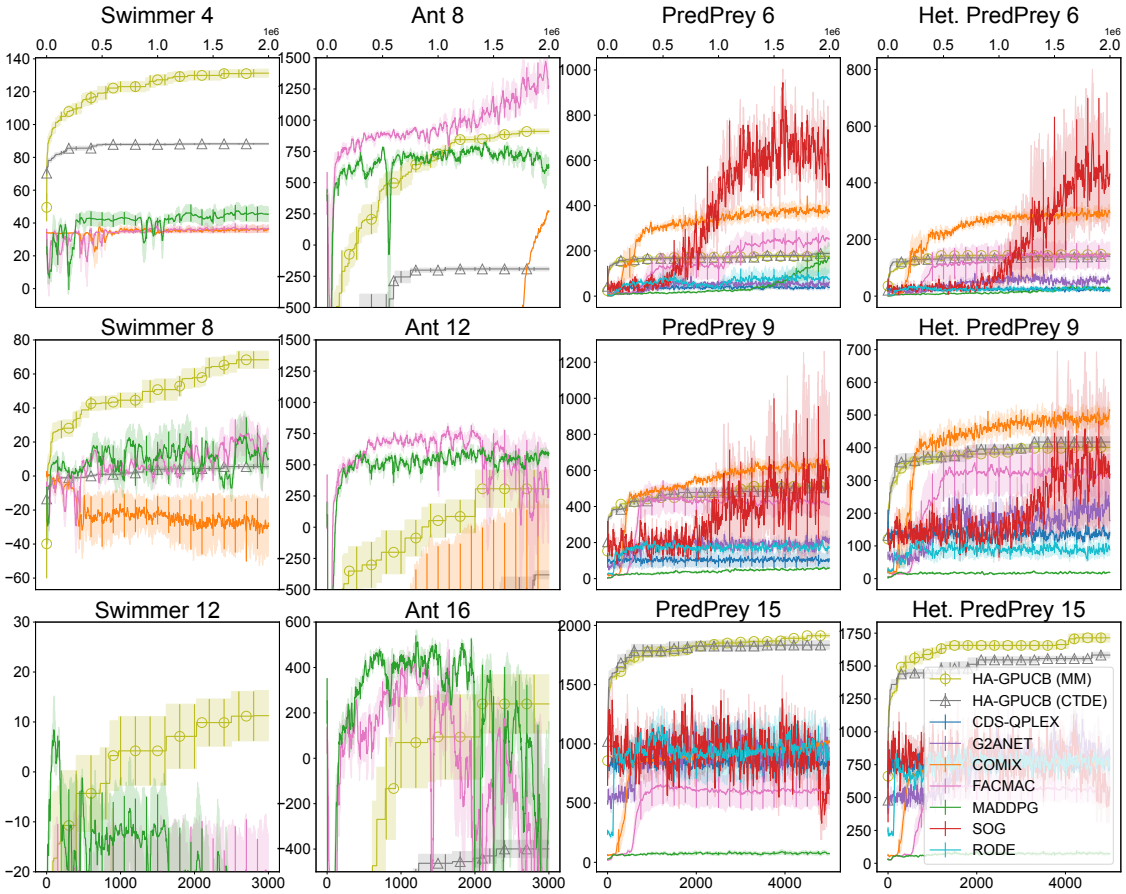

Figure B.2: Comparison with MARL approaches with varying number of agents.

B.2.3 RL and MARL under Malformed Reward

We present additional experiments under malformed reward for both RL and MARL. We formally define the Sparse reward scenario. Let $v(\theta) \triangleq \sum_{\Gamma=1}^{\hat{\Gamma}} r_\Gamma$ where the value of the policy is determined through $\hat{\Gamma}$ interactions with some unknown environment and each interaction is associated with the reward, r_Γ. Typically, RL algorithms observe the reward, r_Γ after every interaction with the environment. We consider a sparse reward scenario where reward feedback is given every S steps: $\tilde{r_\Gamma^S} \triangleq \sum_{\Gamma-S}^{\Gamma} r_\Gamma$ if $\Gamma \equiv 0 \mod S$ and 0 o.w. In addition to the sparse reward setting described earlier, we also consider the setting of delayed reward. The delayed reward scenario

is defined: $r_\Gamma^{\tilde{D}} \triangleq r_{\Gamma-D}$ if $\Gamma > D$ and 0 o.w. Thus in the delayed reward scenario, feedback on an action taken is *delayed*. This scenario is important as it arises in long term planning tasks where the value of an action is not immediately clear, but rather is ascertained after significant delays. We present the complete table comparing related works in RL with HA-GP-UCB in Table B.3. As can be seen, similar to the Sparse reward scenarios, significant degradation can be observed across all tested RL algorithms with HA-GP-UCB outperforming RL algorithms with moderate to severe amount of sparsity or delay. This degradation cannot be overcome by increasing the size of the policy, as we verify with the "XL" models which are orders of magnitude larger with 3 layers of 400 neurons.

We repeat these experimental scenarios in the MARL setting with similar results in Table B.4 where MARL approaches are compared against HA-GP-UCB in the CTDE setting. Thus our validation shows that in both RL and MARL strong performance requires dense, informative feedback which may not be present outside of simulator settings. In these settings, our approach of optimizing small compact policies using HA-GP-UCB outperforms related work in both RL and MARL.

Table B.3: RL under sparse reward. Sparse n refers to sparse reward. Delay n refers to delayed reward. Averaged over 5 runs. Parenthesis indicate standard error.

	Ant-v3					Hopper-v3					Swimmer-v3					Walker2d-v3				
	DDPG	PPO	SAC	TD3	Intrinsic	DDPG	PPO	SAC	TD3	Intrinsic	DDPG	PPO	SAC	TD3	Intrinsic	DDPG	PPO	SAC	TD3	Intrinsic
Baseline	−90.77(318.36)	1105.69(72.49)	2045.24(477.20)	2606.17(96.63)	2144.00(56.19)	604.20(153.01)	1760.65(84.29)	2775.66(390.61)	1895.76(495.13)	1734.00(152.46)	44.45(7.03)	121.38(5.25)	58.73(1.86)	48.78(0.48)	1950.00(136.79)	2203.80(177.40)	892.81(208.68)	4297.03(163.71)	1664.46(490.97)	2210.00(221.94)
Sparse 2	−32.88(548.48)	1007.80(93.20)	2563.97(440.27)	1407.40(239.70)	1964.00(97.47)	877.93(141.24)	1567.14(113.01)	3380.60(76.51)	1570.84(627.28)	2074.00(219.94)	35.59(5.65)	99.50(11.44)	46.75(0.52)	47.23(1.98)	1758.80(254.49)	1470.62(173.60)	1471.33(440.18)	1673.46(704.37)	2297.43(584.68)	1952.00(175.53)
Sparse 5	−2687.97(302.46)	961.31(100.83)	711.56(87.40)	762.61(38.90)	1916.00(96.98)	814.59(110.96)	1616.79(330.77)	3239.20(93.46)	2290.67(560.51)	1972.00(85.15)	26.66(5.52)	68.69(14.60)	43.84(0.99)	40.12(0.76)	1856.00(237.63)	961.30(231.98)	697.93(274.48)	1697.25(682.00)	2932.27(361.78)	1924.00(213.72)
Sparse 20	−2809.89(292.76)	624.07(41.20)	694.30(16.55)	379.12(145.78)	1838.00(189.72)	783.95(175.83)	1629.28(112.73)	2535.17(368.16)	1436.33(535.61)	1537.20(336.19)	19.12(12.07)	54.63(9.12)	37.78(2.06)	37.03(2.63)	2108.00(130.24)	663.04(114.66)	365.39(75.80)	1010.63(490.13)	276.56(69.79)	1810.00(253.57)
Sparse 50	−3067.37(116.57)	−67.43(33.33)	663.28(21.26)	253.66(149.78)	1091.20(102.39)	816.25(145.64)	1010.73(165.81)	1238.03(499.72)	551.43(107.79)	642.00(155.66)	23.73(4.84)	51.52(7.55)	38.78(3.69)	30.01(7.84)	812.00(124.17)	572.12(115.80)	428.29(67.93)	349.47(29.70)	298.28(52.01)	834.75(89.95)
Sparse 100	−3323.43(43.74)	−4021.56(1893.85)	679.30(23.68)	−115.43(387.63)	450.40(51.04)	988.36(15.38)	324.51(53.61)	260.52(31.38)	342.48(36.19)	406.80(51.77)	9.64(9.63)	21.09(4.97)	27.98(6.35)	30.10(1.95)	376.60(80.28)	523.89(114.07)	205.93(18.11)	200.16(36.90)	147.22(24.45)	480.60(75.14)
Sparse 200	−3098.37(73.21)	−8167.98(2801.42)	−107.14(614.25)	−147.86(339.75)	258.60(46.74)	765.05(146.14)	222.76(30.59)	300.36(23.10)	281.68(24.31)	350.80(58.42)	−9.97(6.70)	21.69(1.97)	33.35(3.40)	30.48(2.96)	342.80(10.65)	182.84(56.03)	193.43(21.18)	187.16(33.73)	148.06(23.58)	353.20(39.59)
Sparse XL 100	−2887.23(182.60)	−5101.30(1979.27)	448.67(118.92)	−730.72(720.12)	429.28(141.31)	608.79(116.39)	481.72(68.89)	338.95(10.91)	188.06(67.16)	570.60(59.04)	9.56(8.54)	31.47(1.56)	36.70(2.86)	50.99(11.73)	473.60(67.95)	867.98(65.29)	222.13(23.32)	250.27(12.96)	95.01(43.31)	421.80(31.56)
Sparse XL 200	−2681.07(338.79)	−11084.05(2144.89)	378.36(134.59)	−1331.39(593.91)	373.40(28.56)	1025.16(27.15)	361.34(80.88)	331.69(17.64)	217.04(78.00)	323.00(27.99)	9.87(3.57)	26.53(2.31)	36.61(12.84)	35.21(1.31)	414.40(20.72)	843.38(70.49)	206.66(31.57)	228.23(32.14)	270.81(40.87)	339.20(39.73)
Lag 1	−645.56(227.58)	1028.21(103.66)	1981.58(547.20)	2570.06(222.18)	2378.00(111.98)	872.14(200.36)	1907.03(177.63)	2957.36(394.10)	2817.97(434.89)	1870.00(147.92)	22.56(8.02)	111.21(11.94)	70.53(17.88)	47.56(0.70)	1980.00(106.70)	2197.41(266.42)	1180.54(362.97)	3696.70(712.13)	3806.52(125.83)	1788.00(165.73)
Lag 5	−2849.09(349.92)	848.20(43.75)	866.79(5.83)	860.53(14.72)	1761.00(265.85)	803.61(165.08)	1812.64(219.72)	3280.25(113.50)	3377.79(48.14)	1844.00(124.02)	16.78(6.09)	89.70(14.44)	55.33(4.99)	47.22(4.34)	2018.00(245.36)	1328.87(246.34)	939.46(325.01)	3786.54(650.86)	3192.36(411.20)	1856.00(136.84)
Lag 20	−2578.96(232.02)	239.10(69.51)	824.88(16.22)	54.63(619.95)	1718.00(131.74)	692.46(145.45)	1944.12(263.78)	3385.03(99.86)	2172.20(584.96)	1526.00(141.38)	21.16(9.25)	57.67(4.33)	30.10(6.27)	42.81(3.63)	1910.00(165.43)	424.51(86.68)	740.85(108.44)	2117.99(652.73)	2098.28(608.93)	1726.00(66.79)
Lag 50	−3093.43(184.86)	−176.74(92.67)	796.54(16.98)	791.36(13.30)	1155.60(240.31)	907.45(108.70)	815.34(169.38)	596.78(111.18)	632.85(47.74)	1086.80(109.28)	22.80(7.69)	35.28(1.43)	29.66(2.61)	26.24(2.97)	1092.20(154.49)	816.82(282.02)	368.41(82.30)	388.86(89.26)	463.18(69.68)	942.80(34.55)
Lag 100	−2784.59(92.56)	−5569.67(2110.05)	691.43(22.64)	640.74(150.22)	257.60(39.28)	1028.14(4.61)	216.30(36.22)	264.75(39.10)	280.71(28.86)	292.60(37.57)	11.72(7.89)	23.17(4.52)	37.77(2.77)	29.70(3.84)	377.20(54.35)	623.61(158.22)	197.16(17.30)	209.38(33.69)	164.23(46.02)	382.40(47.09)
Lag 200	−2655.00(279.69)	−9419.68(2630.52)	605.73(38.85)	163.37(233.15)	392.00(35.41)	786.96(167.20)	271.60(46.30)	278.98(25.34)	261.85(28.64)	299.60(28.97)	4.79(7.07)	19.18(2.56)	30.63(5.42)	22.95(15.78)	340.40(42.30)	488.44(128.77)	182.44(4.52)	147.78(20.99)	128.41(9.08)	304.40(29.79)
Lag XL 100	−2722.44(340.81)	−3140.81(1713.55)	503.88(61.88)	−404.24(557.86)	406.80(41.65)	280.53(151.98)	577.16(15.41)	306.99(9.95)	196.16(71.21)	431.40(34.74)	23.54(3.69)	25.59(3.22)	26.03(5.69)	24.72(9.90)	442.60(26.17)	656.25(107.48)	227.56(33.60)	275.18(34.80)	153.97(46.48)	478.00(33.40)
Lag XL 200	−2507.38(142.96)	−8436.44(461.63)	300.83(115.24)	−959.92(648.64)	377.60(26.88)	439.52(175.28)	390.18(80.50)	331.84(15.65)	191.93(69.82)	342.60(33.82)	9.84(5.67)	23.67(3.79)	17.53(1.37)	9.07(7.23)	317.60(54.21)	844.88(40.32)	219.77(23.85)	237.50(19.80)	168.55(49.49)	336.40(44.66)
HA-GP-UCB	1147.21(36.88)					1009.3(17.95)					175.73(15.54)					1008.90(11.95)				

Table B.4: MARL under sparse reward. Sparse n refers to sparse reward. Delay n refers to delayed reward. Averaged over 5 runs. Parenthesis indicate standard error.

	Ant-v3				Hopper-v3				Swimmer-v3				Walker2d-v3			
	COVDN	COMIX	MADDPG	FACMAC	COVDN	COMIX	MADDPG	FACMAC	COVDN	COMIX	MADDPG	FACMAC	COVDN	COMIX	MADDPG	FACMAC
Baseline	970.50(5.96)	959.20(2.83)	982.58(54.19)	909.37(40.08)	38.92(0.06)	38.88(0.06)	305.58(107.48)	638.80(245.03)	−0.05(7.75)	13.43(0.56)	11.21(0.52)	14.06(0.11)	249.21(9.65)	275.85(10.04)	280.30(22.16)	543.71(180.78)
Sparse 5	877.09(20.95)	906.38(7.48)	912.61(12.48)	882.94(46.35)	138.07(72.33)	39.84(0.21)	320.74(123.44)	38.76(0.13)	10.13(2.16)	7.10(4.77)	14.19(0.81)	13.05(0.36)	341.89(51.92)	260.58(29.98)	236.37(28.88)	267.70(118.68)
Sparse 50	242.69(252.04)	−212.74(75.21)	772.29(41.13)	799.71(11.76)	909.88(71.73)	38.88(0.16)	362.66(264.39)	172.37(109.04)	8.49(2.52)	10.50(2.19)	12.06(0.34)	12.10(1.01)	190.92(8.87)	121.51(67.31)	227.60(31.76)	287.36(67.07)
Sparse 100	403.99(208.18)	−756.61(201.61)	490.19(62.22)	564.73(75.80)	443.95(170.88)	27.74(9.52)	47.47(7.08)	38.78(0.06)	−3.44(3.28)	1.12(3.02)	10.11(3.29)	14.14(1.19)	106.24(66.51)	194.91(17.35)	159.24(14.53)	444.41(215.77)
Sparse 200	65.04(89.82)	−197.26(64.03)	−632.05(967.04)	584.23(28.34)	687.82(249.11)	38.78(0.02)	50.74(9.86)	37.10(1.36)	−0.15(3.39)	−1.65(1.20)	14.43(0.73)	13.84(0.13)	165.50(67.12)	277.89(4.89)	123.91(10.61)	229.11(129.53)
Sparse XL 100	766.66(208.18)	209.58(201.61)	632.15(62.22)	552.28(75.80)	813.86(170.88)	236.74(9.52)	50.62(7.08)	72.11(0.06)	9.52(3.28)	14.45(3.02)	13.69(3.29)	14.16(1.19)	135.95(66.51)	28.22(17.35)	127.66(14.53)	184.52(215.77)
Sparse XL 200	442.92(89.82)	322.54(64.03)	479.03(967.04)	553.20(28.34)	996.85(249.11)	39.47(0.02)	26.41(9.86)	71.31(1.36)	3.58(3.39)	4.67(1.20)	13.28(0.73)	11.76(0.13)	167.62(67.12)	207.81(4.89)	157.37(10.61)	88.47(129.53)
Lag 1	656.69(5.96)	426.57(2.83)	901.14(54.19)	835.77(40.08)	55.16(0.06)	38.90(0.06)	113.16(107.48)	38.94(245.03)	1.99(7.75)	9.73(0.56)	13.60(0.52)	14.34(0.11)	413.26(9.65)	318.73(10.04)	312.03(22.16)	357.73(180.78)
Lag 5	255.56(20.95)	302.46(7.48)	873.81(12.48)	910.90(46.35)	629.94(72.33)	38.85(0.21)	376.38(123.44)	852.61(0.13)	10.07(2.16)	4.16(4.77)	14.51(0.81)	13.20(0.36)	170.91(51.92)	289.44(29.98)	302.41(28.88)	640.21(118.68)
Lag 50	56.94(252.04)	−272.78(75.21)	870.20(41.13)	885.37(11.76)	383.76(71.73)	15.95(0.16)	612.84(264.39)	358.68(109.04)	0.79(2.52)	6.26(2.19)	14.55(0.34)	11.16(1.01)	97.29(8.87)	145.98(67.31)	183.41(31.76)	289.66(67.07)
Lag 100	112.82(208.18)	10.21(201.61)	844.80(62.22)	843.90(75.80)	997.51(170.88)	358.31(9.52)	275.65(7.08)	466.20(0.06)	10.41(3.28)	3.49(3.02)	13.67(3.29)	11.60(1.19)	114.41(66.51)	83.99(17.35)	149.93(14.53)	102.84(215.77)
Lag 200	366.62(89.82)	−125.65(64.03)	674.74(967.04)	723.30(28.34)	434.69(249.11)	38.95(0.02)	26.44(9.86)	38.74(1.36)	5.27(3.39)	7.93(1.20)	13.28(0.73)	13.86(0.13)	51.59(67.12)	147.23(4.89)	163.30(10.61)	277.79(129.53)
Lag XL 100	869.35(208.18)	292.96(201.61)	721.18(62.22)	814.25(75.80)	1000.04(170.88)	38.85(9.52)	532.12(7.08)	26.30(0.06)	9.23(3.28)	0.48(3.02)	10.12(3.29)	10.19(1.19)	105.65(66.51)	119.80(17.35)	191.74(14.53)	280.79(215.77)
Lag XL 200	657.07(89.82)	260.13(64.03)	611.73(967.04)	801.94(28.34)	692.92(249.11)	38.85(0.02)	38.78(9.86)	26.27(1.36)	5.62(3.39)	6.48(1.20)	13.57(0.73)	5.56(0.13)	56.72(67.12)	227.96(4.89)	122.66(10.61)	222.77(129.53)
HA-GP-UCB (CTDE)	988.44(3.39)				213.50(22.26)				31.95(1.3)				397.68(14.42)			

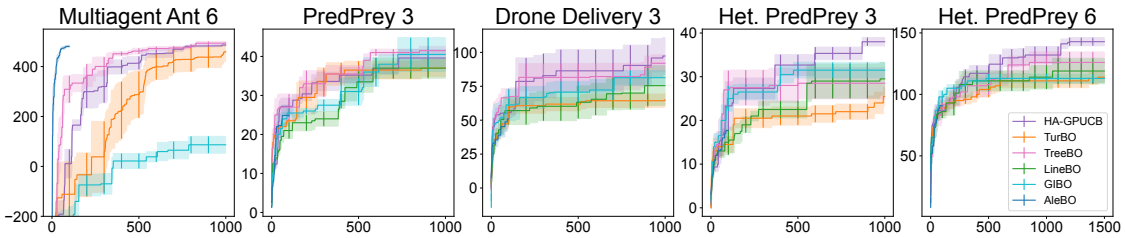

Figure B.3: Comparison with BO algorithms. HA-GP-UCB outperforms on complex multi-agent coordination tasks.

B.2.4 Comparison with HDBO Algorithms

We compare with several related work in High-dimensional BO including TurBO (Eriksson et al. 2019b), AleBO (Letham et al. 2020), LineBO (Kirschner et al. 2019), TreeBO (E. Han, Arora, and Scarlett 2021a), and GIBO (Müller, Rohr, and Trimpe 2021). This is presented in Fig. B.3. We experienced out-of-memory issues with AleBO after approximately 100 iterations, hence the AleBO plots are truncated. We compare against these algorithms at optimizing our HOM policy for solving various multi-agent policy search tasks. We validated on Multiagent Ant with 6 agents, PredPrey with 3 agents, Het. PredPrey with 3 agents, Drone Delivery with 3 agents, and also Het. PredPrey with 6 agents. We observe that these competing works offer competitive performance for simpler tasks such as Multiagent Ant and PredPrey with 3 agents. However for more complex tasks that require role based interaction and coordination, our approach outperforms related work. This is evidenced in Het. PredPrey 3, Het. PredPrey 6 as well as the Drone Delivery task with 3 agents.

Thus our validation shows that for simpler task, competing related works are able to optimize for simple policies of low underlying dimensionality. However, for more complex tasks which require sophisticated interaction using both Role and

Role Interaction, related work is less capable of optimizing for strong policies *due to the complexity of the high-dimensional BO task*. In contrast, our work offers the capability of finding stronger policies for these complex tasks and scenarios.

B.3 On the Applicability of Our Assumptions to RBF and Matern Kernel

We show that our assumption is satisfied by the RBF Kernel when $\Theta = [0,1]^D$, and evidence that by the Matern$-\frac{5}{2}$ kernel does not overly violate the assumptions. We also show that in the setting where $\Theta = [0,r]^D$ for some bounded r, our assumptions are not overly violated as although these kernels may take on small negative values, these values decay exponentially with respect to the distance. These Lemmas show that our assumptions are reasonable.

Lemma B.1. *Let* $k(\theta,\theta') \triangleq \exp(\frac{-d^2}{2})$ *be the RBF kernel with* $d \triangleq ||\theta - \theta'||$*, then*

$$k^{\partial i \partial j}(\theta,\theta') = k(\theta,\theta')\left(1-(\theta^i-\theta'^i)^2\right)\left(1-(\theta^j-\theta'^j)^2\right).$$

Proof. As shown in (Rasmussen and C. K. I. Williams 2006) Section 9.4, the derivative of a Gaussian Process is also a Gaussian Process. Let $GP(0, k(\theta,\theta'))$ be the GP from which f is sampled. This implies:

$$\frac{\partial f}{\partial \theta^a} \sim GP\left(0, \frac{\partial^2 k(\theta,\theta')}{\partial\theta^a \partial\theta'^a}\right).$$

Applying this rule once more for the Hessian, we have:

$$\frac{\partial^2 f}{\partial\theta^b\partial\theta^a} \sim GP\left(0, \frac{\partial^4 k(\theta,\theta')}{\partial\theta^b\partial\theta'^b\partial\theta^a\partial\theta'^a}\right).$$

Given the above identities, we compute the partial derivatives for the RBF kernel:

$$\frac{\partial^2 k\left(\theta,\theta'\right)}{\partial\theta^a\partial\theta'^a} = \exp\left(-\frac{||\theta-\theta'||^2}{2}\right)\left(1-(\theta^a-\theta'^a)^2\right).$$

Deriving once more we have:

$$\frac{\partial^4 k\left(\theta,\theta'\right)}{\partial\theta^b\partial\theta'^b\partial\theta^a\partial\theta'^a} = \exp\left(-\frac{||\theta-\theta'||^2}{2}\right)\left(1-(\theta^a-\theta'^a)^2\right)\left(1-(\theta^b-\theta'^b)^2\right).$$

This completes the proof noting that $k\left(\theta,\theta'\right) \triangleq \exp(\frac{-d^2}{2})$ with $d \triangleq ||\theta-\theta'||$. □

Corollary B.1. *Let* $k\left(\theta,\theta'\right) \triangleq \exp(\frac{-d^2}{2})$*, and* $\theta,\theta' \in [0,1]^D$*, then* $k^{\partial i\partial j}(\theta,\theta') \geq 0$.

Proof. The above is straightforward to see as $\exp\left(\cdot\right) \geq 0$ and with $\theta,\theta' \in [0,1]^D$ we have $(1-(\theta^a-\theta'^a)^2) \geq 0$ $\left(1-(\theta^b-\theta'^b)^2\right) \geq 0$. □

Corollary B.2. *Let* $k\left(\theta,\theta'\right) \triangleq \exp(\frac{-d^2}{2})$*, and* $\theta,\theta' \in [0,r]^d$*, then* $k^{\partial i\partial j}(\theta,\theta') \geq c\exp(-d^2)$ *for some constant* c *dependent on* r.

Proof. The above is straightforward given the above Lemma. We note that although the RBF kernel may take on negative values in the domain $\Theta = [0,r]^d$, this values experience strong tail decay showing that our assumptions are not overly violated. □

The above Lemma and Corollary shows that our assumptions are satisfied by the RBF Kernel when $\Theta = [0,1]^D$, and not overly violated when $\Theta = [0,r]^D$ after choosing a suitable p_h and σ_h^2. We show how these assumptions are not overly violated by the Matern-$\frac{5}{2}$ kernel.

Lemma B.2. *Let* $k(\theta,\theta') \triangleq (1+\sqrt{5}d+\frac{5}{3}d^2)\exp(-\sqrt{5}d)$ *be the Matern-*$\frac{5}{2}$ *kernel with* $d \triangleq ||\theta-\theta'||$*, then with* $d_i \triangleq \theta^i - \theta'^i$ *we have*

$$k^{\partial i\partial j}(\theta,\theta') = \exp(-\sqrt{5}d)\left(\frac{5\sqrt{5}}{3} - \frac{25}{3d}d_i^2 - \frac{25}{3d}d_j^2 + \frac{25\sqrt{5}}{3d^2}d_i^2 d_j^2 + \frac{25}{3d^3}d_i^3 d_j^3\right).$$

Proof. Following the proof of Lemma B.1, we state the partial derivatives of the Matern-$\frac{5}{2}$ kernel:

$$\frac{\partial^2 k(\theta,\theta')}{\partial\theta^a\partial\theta'^a} = \exp\left(-\sqrt{5}||\theta-\theta'||\right)\left(\frac{5}{3} + \frac{5\sqrt{5}}{3}||\theta-\theta'|| - \frac{25}{3}(\theta^a-\theta'^a)^2\right).$$

Differentiating one more we have

$$\begin{aligned}\frac{\partial^4 k(\theta,\theta')}{\partial\theta^b\partial\theta'^b\partial\theta^a\partial\theta'^a} &= \exp\left(-\sqrt{5}||\theta-\theta'||\right)\\ &\left(\frac{5\sqrt{5}}{3} - \frac{25}{3d}(\theta^a-\theta'^a)^2 - \frac{25}{3d}(\theta^b-\theta'^b)^2 + \frac{25\sqrt{5}}{3d^2}(\theta^a-\theta'^a)^2(\theta^b-\theta'^b)^2\right.\\ &\left. + \frac{25}{3d^3}(\theta^a-\theta'^a)^3(\theta^b-\theta'^b)^3\right).\end{aligned}$$

This completes the proof noting that $d_i \triangleq \theta^i - \theta'^i$ and $d \triangleq ||\theta-\theta'||$. □

Corollary B.3. *Let* $k(\theta,\theta') \triangleq (1+\sqrt{5}d+\frac{5}{3}d^2)\exp(-\sqrt{5}d)$ *and* $\theta,\theta' \in [0,1]^D$*. Then* $k^{\partial i\partial j}(\theta,\theta') \geq \exp(-\sqrt{5}d)\left(\frac{5\sqrt{5}}{3} - \frac{25}{3d} - \frac{25}{3d} - \frac{25}{3d^3}\right)$.

Proof. The above is an immediate consequence of Lemma B.2 and noting that $||d_i|| \leq 1$. □

Corollary B.4. *Let* $k(\theta,\theta') \triangleq (1+\sqrt{5}d+\frac{5}{3}d^2)\exp(-\sqrt{5}d)$ *and* $\theta,\theta' \in [0,r]^d$*. Then* $k^{\partial i\partial j}(\theta,\theta') \geq c\exp(-d)$ *for some* c *dependent on* r*.*

Proof. The above is an immediate consequence of Lemma B.2 and noting that $||d_i|| \leq r$. □

Although the above corollary shows that the Matern-$\frac{5}{2}$ kernel may take on negative values, we note that these values experience strong tail decay due to the presence of the $\exp\left(-\sqrt{5}d\right)$ term. Thus, the negative values are likely to be extremely small in expectation, thus not overly violating our assumptions. In our experiments, we observed no shortcoming in using the Matern-$\frac{5}{2}$ kernel in HA-GP-UCB.

B.4 Proof of Proposition 5.1

We restate Proposition 5.1 for clarity.

Proposition 5.1. *Let $\mathcal{G}_d = (V_d, E_d)$ represent an additive dependency structure with respect to $v(\theta)$, then the following holds true: $\forall a, b \, \frac{\partial^2 v}{\partial \theta^a \partial \theta^b} \neq 0 \implies (\Theta^a, \Theta^b) \in E_d$ which is a consequence of v formed through addition of independent sub-functions $v^{(i)}$, at least one of which must contain θ^a, θ^b as parameters for $\frac{\partial^2 v}{\partial \theta^a \partial \theta^b} \neq 0$ which implies their connectivity within E_d.*

Proof. The above follows from the linearity of addition, which naturally implies a lack of curvature. In the multivariate case, this corresponds to zero or non-zero entries in the Hessian.

To be precise, we prove the contrapositive:

$$(\Theta^a, \Theta^b) \notin E_d \implies \frac{\partial^2 v}{\partial \theta^a \partial \theta^b} = 0.$$

Let a, b be arbitrary dimensions with $(\Theta^a, \Theta^b) \notin E_d$. As a consequence of the definition of the dependency graph, $\nexists \Theta^{(i)}$ s.t. $\{\Theta^a, \Theta^b\} \subseteq \Theta^{(i)}$. That is, no subfunction $v^{(i)}$ takes both θ^a and θ^b as arguments.

By the linearity of the partial derivative, we see that:

$$\frac{\partial^2}{\partial \theta^a \partial \theta^b} v(\theta) = \frac{\partial^2}{\partial \theta^a \partial \theta^b} \sum_{i=1}^{M} v^{(i)}(\theta^{(i)}) = \sum_{i=1}^{M} \frac{\partial^2}{\partial \theta^a \partial \theta^b} v^{(i)}(\theta^{(i)}) = 0$$

where the last equality follows from no subfunction $v^{(i)}$ taking both θ^a and θ^b as arguments. □

B.5 Proof of Theorem 5.1

Our proof of Theorem 5.1 relies in being able to determine whether an edge does or does not exist in the dependency graph. To be able to do this, we examine the Hessian. As we have shown in Proposition 5.1, examining the Hessian answers this question. The challenge of Theorem 5.1 is detecting this dependency under noisy observations of the Hessian, as well as in domains where the variance of the second partial derivative is often zero, i.e., $k^{\partial i \partial j}(\theta, \theta') = 0$ with high probability. To overcome this challenge, we sample the Hessian multiple times to both find portions of the domain where $k^{\partial i \partial j}(\theta, \theta') \geq \sigma_h^2$, and also reduce the effect of the noise on learning the dependency structure. To proceed with the analysis, we first prove a helper lemma showing that if we can construct two Normal variables of sufficiently different variances, then it's possible to accurately determine which Normal variable has low, and high variance by taking a singular sample from each. This helper lemma will be used later to help determine edges in the dependency graph. As we shall soon show, If an edge exists, we are able to construct a Normal variable with high variance. Correspondingly, if an edge does not exist, we are able to construct a Normal variable with low variance.

Lemma B.1. *Let $X_l \sim \mathcal{N}(0, \sigma_l^2)$ and $X_h \sim \mathcal{N}(0, \sigma_h^2)$ be two random univariate gaussian variables. For any $\delta \in (0, 1)$, $\exists\, c_h$ s.t. $|X_l| \leq c_h \leq |X_h|$ with probability $1 - \delta$ when $\frac{\sigma_h^2}{\sigma_l^2} > \frac{8}{\delta^2} \log \frac{2}{\delta}$ and precisely when $\frac{\sigma_h \delta}{2} > c_h > \sigma_l \sqrt{2 \log \frac{2}{\delta}}$.*

Proof. First we note that $|X_l|$ and $|X_h|$ are Half-Normal random variables, with cumulative distribution function of $F_l(x) = \text{erf}\, \frac{x}{\sigma_l \sqrt{2}}$ and $F_h(x) = \text{erf}\, \frac{x}{\sigma_h \sqrt{2}}$ respectively.

Thus to show that $|X_l| \leq \sigma_l\sqrt{2\log\frac{2}{\delta}}$ and $|X_h| \geq \frac{\sigma_h\delta}{2}$ with high probability, we utilize well known bounds on the erf and erfc function. The proofs of the below can be found in several places, e.g., Chu 1955 and Ermolova and Häggman 2004 respectively.

$$\operatorname{erf} x \leq \sqrt{1-\exp -2x^2}\,;\ \operatorname{erfc} x \leq \exp -x^2.$$

Given the above, we show that $p(c_h \leq |X_l|) \leq \frac{\delta}{2}$ and $p(c_h \geq |X_h|) \leq \frac{\delta}{2}$ and utilizing the union bound completes the proof.

$$c_h > \sigma_l\sqrt{2\log\frac{2}{\delta}} \implies c_h^2 > 2\sigma_l^2\log\frac{2}{\delta} \implies \frac{c_h^2}{2\sigma_l^2} > -\log\frac{\delta}{2} \implies -\frac{c_h^2}{2\sigma_l^2} < \log\frac{\delta}{2}$$

$$\implies \exp -\frac{c_h^2}{2\sigma_l^2} \leq \frac{\delta}{2} \implies \operatorname{erfc}\frac{c_h}{\sqrt{2}\sigma_l} < \frac{\delta}{2} \implies 1-\operatorname{erf}\frac{c_h}{\sqrt{2}\sigma_l} \geq 1-\frac{\delta}{2} \implies F_l(c_h) \geq 1-\frac{\delta}{2}$$

$$\implies p\left(c_h \leq |X_l|\right) < \frac{\delta}{2}.$$

Following a similar line of reasoning we have:

$$c_h < \frac{\sigma_h\delta}{2} \implies \frac{c_h^2}{\sigma_h^2} < \frac{\delta^2}{4} \implies \frac{-c_h^2}{\sigma_h^2} > -\frac{\delta^2}{4} \implies \frac{-c_h^2}{\sigma_h^2} > \log 1 - \frac{\epsilon^2}{4} \implies \exp -\frac{c_h^2}{\sigma_h^2} > 1-\frac{\delta^2}{4}$$

$$\implies 1-\exp -\frac{c_h^2}{\sigma_h^2} < \frac{\delta^2}{4} \implies \sqrt{1-\exp -\frac{c_h^2}{\sigma_h^2}} < \frac{\delta}{2} \implies \operatorname{erf}\frac{c_h}{\sigma_h\sqrt{2}} < \frac{\delta}{2} \implies F_h(c_h) < \frac{\delta}{2}$$

$$\implies p(c_h \geq |X_h|) < \frac{\delta}{2}.$$

Finally, to complete the proof, we show that the interval $(\sigma_l\sqrt{2\log\frac{2}{\delta}}, \frac{\sigma_h\delta}{2})$ is not the empty set when $\frac{\sigma_h^2}{\sigma_l^2} > \frac{8}{\delta^2}\log\frac{2}{\delta}$.

$$\frac{\sigma_h^2}{\sigma_l^2} > \frac{8}{\delta^2}\log\frac{2}{\delta} \implies \frac{\sigma_h}{\sigma_l} > \frac{2\sqrt{2}}{\delta}\sqrt{\log\frac{2}{\delta}} \implies \frac{\sigma_h\delta}{2} > \sigma_l\sqrt{2\log\frac{2}{\delta}}.$$

□

We are now ready to prove Theorem 5.1.

Theorem 5.1. *Suppose*[1] *there exists* σ_h^2, p_h *s.t.* $\forall i,j\ \mathbb{P}_{\theta\sim\mathcal{U}(\Theta)}\left[k^{\partial i\partial j}(\theta,\theta) \geq \sigma_h^2\right] \geq p_h$ *and* $\forall i,j,\theta,\theta'\ k^{\partial i\partial j}(\theta,\theta') \geq 0$. *Then for any* $\delta_1,\delta_2 \in (0,1)$ *after* $t \geq T_0$ *steps of HA-GP-UCB we have:* $\bigcap_{i,j} P(\widetilde{E}_d^{i,j} = E_d^{i,j}) \geq 1-\delta_1-\delta_2$ *when* $T_0 = C_1 > \frac{16D^2}{p_h\delta_1^2}\log\frac{2D^2}{\delta_1}\frac{\sigma_n^2}{\sigma_h^2} + \frac{D^2}{2\delta_2}$, $c_h \triangleq T_0\sigma_n\sqrt{2\log\frac{2D^2}{\delta_1}}$.

Proof. We prove the above for a single pair of variables, i.e., $k^{\partial i\partial j}$ and utilize the union bound to complete the proof. The first challenge to overcome is to sufficiently sample enough points in the domain such that we are able to find enough points $\theta \in \Theta$ where $k^{\partial i\partial j}(\theta,\theta) \geq \sigma_h^2$. To achieve this we sample T_0 different θ in the domain. After sampling T_0 points if there exists an edge between Θ^a, and Θ^b, then with probability $1-\frac{\delta_2}{D^2}$ we have sampled $\frac{T_0p_h}{2} - \frac{D^2}{2\delta_2}$ points where $k^{\partial i\partial j}(\theta,\theta) \geq \sigma_h^2$. To show the above we use bounds on the cumulative distribution of the Binomial distribution. A bound is given T_0 trials, with p_h probability of success, the probability of having fewer than s successes is upper bounded as follows:

$$\frac{1}{T_0p_h - 2s}.$$

The above bound derives from the following well known tail bound (Feller 1991):

[1]RBF kernel satisfies these assumptions when $\Theta = [0,1]^D$.

$$P\left(S_n \leq s\right) \leq \frac{(T_0 - s)p_h}{(T_0 p_h - s)^2} \quad \text{if} \quad s \leq T_0 p_h$$

where S_n denotes the number of successes. The above bound can be loosened by the following process:

$$\frac{(T_0 - s)p_h}{(T_0 p_h - s)^2} \leq \frac{T_0 p_h}{(T_0 p_h - s)^2} = \frac{T_0 p_h}{T_0^2 p_h^2 + s^2 - 2T_0 p_h s} \leq \frac{T_0 p_h}{T_0^2 p_h^2 - 2T_0 p_h s} =$$

$$\frac{T_0 p_h}{T_0 p_h (T_0 p_h - 2s)} = \frac{1}{T_0 p_h - 2s}$$

which yields the bound that we utilize. We note the above bound requires $s \leq \frac{T_0 p_h}{2} - \frac{1}{2}$, however if it is the case that $s \geq \frac{T_0 p_h}{2} - \frac{1}{2}$ then the worst case tail analysis is unnecessary since $\frac{T_0 p_h}{2} - \frac{1}{2} \geq \frac{T_0 p_h}{2} - \frac{D^2}{2\delta_2}$ and our results still hold.

Given the above, we use δ_2 and derive:

$$\frac{1}{T_0 p_h - 2(\frac{T_0 p_h}{2} - \frac{D^2}{2\delta_2})} \leq \frac{\delta_2}{D^2}.$$

Given the above, with at least $\frac{T_0 p_h}{2} - \frac{D^2}{2\delta_2}$ points where $k^{\partial i \partial j}(\cdot, \cdot) \geq \sigma_h^2$, as well as our assumption $k^{\partial i \partial j}(\theta, \theta) \geq 0$, we apply Bienaymé's identity which we restate for convenience:

$$\text{Var}\left[\sum_{\ell=1}^{C_1} h_{t,\ell}\right] = \sum_{\ell=1}^{C_1} \sum_{\ell'=1}^{C_1} \text{Cov}\left(h_{t,\ell}, h_{t,\ell'}\right).$$

Noting each of the $\frac{T_0 p_h}{2} - \frac{D^2}{2\delta_2}$ successes is sampled $C_1 = T_0$ times with $\text{Cov}\left(h_{t,\ell}, h_{t,\ell'}\right) \geq \sigma_h^2$ for each of the successes and $\text{Cov}\left(h_{t,\ell}, h_{t,\ell'}\right) \geq 0$ for all samples by our assumption. Applying Bienaymé's identity and the sum of (correlated) Normal variables is

also a normal variable, we have Var $\left[\sum_{t=1}^{C_1}\sum_{\ell=1}^{C_1} h_{t,\ell}\right] \geq (\frac{T_0 p_h}{2} - \frac{D^2}{2\delta_2})T_0^2\sigma_h^2$. Compare this quantity with the variance if no edge exists between Θ^a, and Θ^b, where the variance results from i.i.d. noise: Var $\left[\sum_{t=1}^{T_0}\sum_{\ell=1}^{T_0} h_{t,\ell}\right] = T_0^2\sigma_n^2$. Comparing these two quantities, with an appropriately picked c_h determines the edge between Θ^a and Θ^b using Lemma B.1. By Lemma B.1, letting $c_h \triangleq T_0\sigma_n\sqrt{2\log\frac{2D^2}{\delta_1}}$ ensures that $p(h^{i,j} < c_h) < \frac{\delta_1}{D^2}$ if edge $E_d^{i,j}$ exists, and $p(h^{i,j} > c_h) < \frac{\delta_1}{D^2}$ if edge $E_d^{i,j}$ does not exist. Applying the union bound over D^2 pairs of variables completes the proof with $\bigcap_{i,j} P(\widetilde{E}_d^{i,j} = E_d^{i,j}) \geq 1 - \delta_1 - \delta_2$.

□

B.6 Proof of Theorem 5.2

Our proof of Theorem 5.2 is presented under the same setting and assumptions as the work of Srinivas et al. 2010.

To prove Theorem 5.2, we rely on several helper lemmas. The high-level sketch of the proof is to use the properties of Erdős-Rényi graph to bound both the *size of the maximal clique* as well as *the number of maximal cliques* with high probability. Once these two quantities are bounded, we are able to analyze the mutual information of the kernel constructed by *summing the kernels corresponding to the maximal cliques* of the sampled Erdős-Rényi graph as indicated in Assumption 5.1. Finally, once this mutual information is bounded, we use similar analysis as Srinivas et al. 2010 to complete the regret bound.

We begin by bounding the size of the maximal cliques.

Lemma B.1. *Let $\mathcal{G}_d = (V_d, E_d)$ be sampled from a Erdős-Rényi model with probability p_g: $\mathcal{G}_d \sim G(D, p_g)$, then $\forall \delta \in (0,1)$ the largest clique of $\mathcal{G}_d$ is bounded above by*

$$|\textit{Max-Clique}(\mathcal{G}_d)| \leq 2\log_{\left\lfloor \frac{1}{p_g} \right\rfloor}|V_d| + 2\sqrt{\log_{\left\lfloor \frac{1}{p_g} \right\rfloor}\frac{|V_d|}{\delta} + 1}$$

with probability at least $1 - \delta$.

Proof. The above relies on well known upper bounds on the maximal clique size on a graph sampled from an Erdős-Rényi model. As shown in (Bollobás and Erdös 1976) and (Matula 1976) the expected number of Cliques of size k, $\mathbb{E}[C_k]$ is given by:

$$\mathbb{E}\left[C_k\right] = \binom{|V_d|}{k}\left[\frac{1}{p_g}\right]^{-\binom{k}{2}} \leq |V_d|^k \left[\frac{1}{p_g}\right]^{-\frac{k(k-1)}{2}} = \left[\frac{1}{p_g}\right]^{\frac{k}{2}\left(2\log_{\left[\frac{1}{p_g}\right]}|V_d|-k+1\right)}.$$

In the sequel, we omit the base of the log: $\left[\frac{1}{p_g}\right]$ for clarity. To bound the size of the maximal clique, we find a suitable k such that $\mathbb{E}\left[C_k\right] \leq \frac{\delta}{n}$ and utilize the union bound over $[C_i]_{i=k,\ldots,n}$ where we have $|[C_i]_{i=k,\ldots,n}| \leq n$. Finally, we utilize Markov's inequality to complete the proof.

$$\text{Let } k = 2\log|V_d| + 2\sqrt{\log\frac{|V_d|}{\delta}+1}.$$

We utilize the above bound on $\mathbb{E}\left[C_k\right]$.

$$\implies \frac{k}{2}\left(2\log_{\left[\frac{1}{p_g}\right]}|V_d| - k + 1\right) =$$
$$\left(\log|V_d| + \sqrt{\log\frac{n}{\delta}}\right)\left(2\log|V_d| - 2\log|V_d| - 2\sqrt{\log\frac{n}{\delta}+1}+1\right)$$
$$\leq -\log|V_d| - \log\frac{n}{\delta} + 1 \leq \log\frac{\delta}{n}$$

$$\implies \mathbb{E}\left[C_k\right] \leq \left[\frac{1}{p_g}\right]^{\log\frac{\delta}{n}} = \frac{\delta}{n}.$$

The proof is complete by noting that by Markov inequality, $p(C_k \geq 1) \leq \mathbb{E}\left[C_k\right]$ and taking the union bound over at most n members of $[C_i]_{i=k,\ldots,n}$. □

Next, we bound the total number of maximal cliques:

Lemma B.2. *Let $\mathcal{G}_d = (V_d, E_d)$ be sampled from a Erdős-Rényi model with probability p: $\mathcal{G}_d \sim G(D, p_g)$, then $\forall \delta \in (0,1)$ the number of total maximal cliques in $\mathcal{G}_d$ is bounded above by*

$$\frac{1}{\delta}\sqrt{|V_d|^{\log_{\left\lceil \frac{1}{p_g} \right\rceil}|V_d|+5}}$$

with probability at least $1-\delta$.

Proof. We prove the above by bounding $\max_k C_k$ with high probability and noting that the number of maximal cliques is bounded by $\sum_k C_k \leq n \max_k C_k$ with high probability. To bound $\max C_k$, we first consider $\max_k \mathbb{E}[C_k]$.

$$\max_k \mathbb{E}[C_k] = \max_k \left\lceil \frac{1}{p_g} \right\rceil^{\frac{k}{2}\left(2\log_{\left\lceil \frac{1}{p_g} \right\rceil}|V_d|-k+1\right)} = \left\lceil \frac{1}{p_g} \right\rceil^{\max_k \frac{k}{2}\left(2\log_{\left\lceil \frac{1}{p_g} \right\rceil}|V_d|-k+1\right)}.$$

Taking the partial derivative of $\frac{k}{2}\left(2\log_{\left\lceil \frac{1}{p_g} \right\rceil}|V_d| - k + 1\right)$ with respect to k we determine the maximum:

$$\arg\max_k \frac{k}{2}\left(2\log_{\left\lceil \frac{1}{p_g} \right\rceil}|V_d| - k + 1\right) = \log_{\left\lceil \frac{1}{p_g} \right\rceil}|V_d| + 1.$$

Thus we are able to bound:

$$\frac{\log_{\left\lceil \frac{1}{p_g} \right\rceil}|V_d|+1}{2}\left(2\log_{\left\lceil \frac{1}{p_g} \right\rceil}|V_d| - \log_{\left\lceil \frac{1}{p_g} \right\rceil}|V_d| - 1 + 1\right) =$$

$$\frac{\log_{\left\lceil \frac{1}{p_g} \right\rceil}|V_d|+1}{2}\left(\log_{\left\lceil \frac{1}{p_g} \right\rceil}|V_d|\right) = \frac{1}{2}\log^2_{\left\lceil \frac{1}{p_g} \right\rceil}|V_d| + \frac{1}{2}\log_{\left\lceil \frac{1}{p_g} \right\rceil}|V_d|$$

Which yields the bound:

$$\mathbb{E}\left[C_k\right] \leq \left[\frac{1}{p_g}\right]^{\frac{1}{2}\log^2\left[\frac{1}{p_g}\right]|V_d|+\frac{1}{2}\log\left[\frac{1}{p_g}\right]|V_d|} = \sqrt{|V_d|^{\log\left[\frac{1}{p_g}\right]|V_d|+1}}.$$

To complete the proof, we utilize Markov's inequality with $p\left(C_k \geq \frac{|V_d|}{\delta}\sqrt{|V_d|^{\log\left[\frac{1}{p_g}\right]|V_d|+1}}\right) \leq \frac{\delta}{|V_d|}$ and utilize the union bound over n choices of k:

$$\sum_k C_k \leq \sum_k \frac{|V_d|}{\delta}\sqrt{|V_d|^{\log\left[\frac{1}{p_g}\right]|V_d|+1}} = \frac{1}{\delta}\sqrt{|V_d|^{\log\left[\frac{1}{p_g}\right]|V_d|+5}}$$

with probability $1-\delta$. □

Now that we have bounded both the number of cliques, as well as the sizes of the maximal cliques with high probability, we now consider the mutual information of the kernel constructed by summing the kernels corresponding to the maximal cliques of the dependency graph.

Lemma B.3. *Define* $I(\mathbf{y}_A; v) \triangleq \mathrm{H}(\mathbf{y}_A) - \mathrm{H}(\mathbf{y}_A \mid v)$ *as the mutual information between* $\mathbf{y}_A$ *and* v *with* $\mathrm{H}(\mathcal{N}(\mu,\Sigma)) \triangleq \frac{1}{2}\log|2\pi e\Sigma|$ *as the entropy function. Define* $\gamma_T^k \geq \max_{A\subset\Theta:|A|=T} I(\mathbf{y}_A; v)$ *when* $v \sim GP(0, k(\theta,\theta'))$. *Let* $[k_i]_{i=1,\ldots,M}$ *be arbitrary kernels defined on the domain* Θ *with upper bounds on mutual information* $[\gamma_T^{k_i}]_{i=1,\ldots,M}$, *then the following holds true:*

$$\gamma_T^{\sum_i k_i} \leq M^2 \max\left[\gamma_T^{k_i}\right]_{i=1,\ldots,M}.$$

To prove the above, we first state Weyl's inequality for convenience:

Lemma B.4. *Let $H, P \in \mathbb{R}^{n\times n}$ be two Hermitian matrices and consider the matrix $M = H + P$. Let $\mu_i, \nu_i, \rho_i, i = 1, \ldots, n$ be the eigenvalues of M, H, and P respectively in decreasing order. Then, for all $i \geq r + s - 1$ we have*

$$\mu_i \leq \nu_r + \rho_s.$$

The above has an immediate Corollary as noted by Rolland et al. 2018:

Corollary B.5. *Let $K_i \in \mathbb{R}^{n\times n}$ be Hermitian matrices for $i = 1, \ldots, M$ with $K \triangleq \sum_i^M K_i$. Let $[\lambda_\ell^{K_i}]_{\ell=1,\ldots,n}$ denote the eigenvalues of K_i in decreasing order. Then for all $\ell \in \mathbb{N}_0$ such that $\ell M + 1 \leq n$ we have*

$$\lambda_{\ell M+1}^{K} \leq \sum_{i=1}^{M} \lambda_{\ell+1}^{K_i}.$$

We are now ready to prove Lemma B.3 using Weyl's inequality and its corollary as a key tool.

Proof. Given the definition of $I(\mathbf{y}_A; v) \triangleq \frac{1}{2}\log|\mathbf{I} + \sigma^{-2}\mathbf{K}_A^k|$ (Srinivas et al. 2010) we bound the eigenvalues of $M\mathbf{I} + \sigma^{-2}\sum_i^M \mathbf{K}_A^{k_i}$ using the eigenvalues of $[I + \sigma^{-2}\mathbf{K}_A^{k_i}]_{i=1,\ldots,M}$ where $k \triangleq \sum_{i=1}^M k_i$. Using the above Corollary we see that:

$$\lambda_\ell^{M\mathbf{I}+\sigma^{-2}K} \leq \sum_{i=1}^{M} \lambda_{\lceil \frac{\ell}{M} \rceil}^{I+\sigma^{-2}K_i}.$$

Given the above, we see that $M^2 \max[\gamma_T^{k_i}]_{i=1,\ldots,M} \geq \frac{1}{2}\log|\mathbf{I}+\sigma^{-2}\mathbf{K}_A^k|$ as $\sum_i^M M\gamma_T^{k_i} \geq \frac{1}{2}\log|M\mathbf{I} + \sigma^{-2}\sum_i^M \mathbf{K}_A^{k_i}|$.

□

Finally, we require an additional helper lemma to bound the supremum and infimum of a function sampled from a GP. This helper lemma helps bound the regret during the first phase of HA-GP-UCB where we randomly sample the Hessian over the domain.

Lemma B.5. *Let $k(\theta, \theta')$ be four times differentiable on the continuous domain $\Theta \triangleq [0, r]^D$ for some bounded r (i.e., compact and convex) with $f \sim GP(0, k(\theta, \theta'))$ then for all $\delta \in (0, 1)$ the following holds true:*

$$\sup_{\theta \in [0,r]^D} f \leq c_b \sqrt{D \log \delta^{-1}} = \mathcal{O}\left(\sqrt{D \log \delta^{-1}}\right).$$

$$\inf_{\theta \in [0,r]^D} f \geq -c_b \sqrt{D \log \delta^{-1}} = \Omega\left(-\sqrt{D \log \delta^{-1}}\right).$$

for some constant c_b dependent on δ and r, with probability $1 - \delta$.

Proof. The proof of the above is contained in Srinivas et al. 2010 Lemma 5.8. We restate the key parts of the proof herein. In the setting of Srinivas et al. 2010 there exist constants a, b_i, such that

$$P\left[\sup_{\theta \in \Theta} \left|\frac{\partial v}{\partial \theta_i}\right| > L\right] \leq a e^{-b_i L^2}.$$

Letting $L = [\log(Da2/\delta)/\min_i b_i]^{1/2}$, we have that $ae^{-b_i L^2} \leq \delta/(2D)$ for all $i = 1, \ldots, D$, so that for $K_1 = D^{1/2} L$ by the mean value theorem, we have

$$P\left[|v(\theta) - v(\theta')| \ \leq \ K_1 ||\theta - \theta'|| \ \ \forall\ \theta, \theta' \in \Theta\right] \geq 1 - \delta/2.$$

□

The proof is complete by noting that $K_1 = \mathcal{O}((\log \delta^{-1})^{1/2})$ and picking an appropriate c_b dependent on δ and r.

We are now ready to prove Theorem 5.2.

Theorem 5.2. *Let k be the kernel as in Assumption 5.1, and Theorem 5.1. Let $\gamma_T^k(d) : \mathbb{N} \to \mathbb{R}$ be a monotonically increasing upper bound function on the* mutual information *of kernel k taking d arguments. The cumulative regret of HA-GP-UCB is bounded with high probability as follows:*

$$R_T = \widetilde{\mathcal{O}}\left(\sqrt{T\beta_T D^{\log D+5}\gamma_T^k(4\log D + c_\gamma)}\right) \tag{5.3}$$

where c_γ is an appropriately picked constant and the base of the logarithm is $\frac{1}{p_g}$.

We restate the above theorem with more precision:

Theorem 5.2. *Let k be the kernel as in Assumption 5.1, and Theorem 5.1 and for some constants a, b,*

$$P\left[\sup_{\theta\in\Theta}\left|\frac{\partial v}{\partial\theta_i}\right| > L\right] \leq ae^{-(L/b)^2}, i = 1, \ldots, D.$$

Let $\gamma_T^k(d) : \mathbb{N} \to \mathbb{R}$ be a monotonically increasing upper bound function on the mutual information *of kernel k taking d arguments. Let $k(\theta, \theta')$ be four times differentiable on the continuous domain $\Theta \triangleq [0, r]^d$ for some bounded r (i.e., compact and convex). For any $\delta_1, \delta_2, \delta_3, \delta_4, \delta_5, \delta_6 \in (0, 1)$. Let, $\tilde{t} \triangleq t - T_0C_1$ and let*

$$\beta_t = 2\log(\tilde{t}^2 2\pi^2/3\delta_6^2) + 2D\log(\tilde{t}^2 Dbr\sqrt{\log(4Da/\delta_6)})$$

The cumulative regret of HA-GP-UCB is bounded:

$$P\left[R_T \leq 2C_1^2 c_b\sqrt{D\log\delta_5^{-1}} + \sqrt{C_2 T\beta_T\gamma_T} + 2 \quad \forall T \geq 1\right] \geq 1-\delta_1-\delta_2-\delta_3-\delta_4-\delta_5-\delta_6$$

when $C_1 = \frac{16D^2}{p_h\delta_1^2}\log\frac{2D^2}{\delta_1}\frac{\sigma_n^2}{\sigma_h^2} + \frac{D^2}{2\delta_2} + 1$, $C_2 = 8/\log(1+\sigma^{-2})$, and $\gamma_T = \frac{1}{\delta_4^2}D^{\log_{1/p_g}D+5}\gamma_T^k\left(2\log_{1/p_g}D + 2\sqrt{\log_{1/p_g}D/\delta_3 + 1}\right)$ where c_b is some constant dependent on δ_5.

Proof. The proof is a consequence of the helper lemmas and theorems we have proved. First we consider Phase 1 of HA-GP-UCB where $t \leq T_0$. By Theorem 5.1, at most $T_0C_1 = C_1^2$ queries will be made during Phase 1, and Lemma B.5 indicates

the maximum regret for any query. Consulting the respective Theorem and Lemma, we are able to bound the cumulative regret during Phase 1 by:

$$2C_1^2 c_b \sqrt{D \log \delta_5^{-1}} = \mathcal{O}(D^{4.5} \log^2 D).$$

Considering Phase 2, we utilize Lemma B.1, Lemma B.2, Lemma B.3 to bound the mutual information of the sampled kernel with high probability. The number of cliques is given by:

$$\frac{1}{\delta_4} \sqrt{D^{\log_{1/p_g} D+5}}.$$

The size of the largest clique is given by:

$$2\log_{1/p_g} D + 2\sqrt{\log_{1/p_g} D/\delta_3 + 1} \leq 2(\log_{1/p_g} D + \log_{1/p_g} D/\delta_3 + 1) \leq 4\log_{1/p_g} D/\delta_3 + 2$$

Following Lemma B.3, we see that

$$\gamma_T^{\sum_i k_i} \leq M^2 \max\left[\gamma_T^{k_i}\right]_{i=1,\ldots,M} \leq \frac{1}{\delta_4^2} D^{\log_{1/p_g} D+5} \gamma_T^k (4\log_{1/p_g} D/\delta_3 + 2).$$

Let $c_\gamma = 4\log_{\frac{1}{p_g}} 1/\delta_3 + 2$ which yields the following information on the mutual information:

$$\gamma_T^{\sum_i k_i} \leq D^{\log_{1/p_g} D+5} \gamma_T^k (4\log_{1/p_g} D + c_\gamma).$$

The proof is complete by leveraging the connection between mutual information and cumulative regret as shown by Srinivas et al. 2010 where $\tilde{\mathcal{O}}$ is the same as $\mathcal{O}$ with the log factors suppressed. □

B.7 On the Surrogate Hessian

In Section 5.2.5 we remarked that although we cannot observe $\mathbf{H}_v$, we can observe a surrogate hessian, $\mathbf{H}_\pi$ which is related to $\mathbf{H}_v$ by the chain rule. We justify our choice here with showing how $\mathbf{H}_\pi$ is an important sub-component of $\mathbf{H}_v$ (Skorski 2019). Although the reasoning we give is in one dimension, an analogous argument can be made in arbitrary dimensions using the chain rule for vector-valued functions yielding the Hessian tensor (Magalhães 2020). We have $v : \Theta \to \mathbb{R}$ is a function of the policy π and can be expressed as a composition of functions:

$$v : \Theta \to \mathbb{R} = \hat{v}\left(\pi\left(\theta\right)\right). \tag{B.1}$$

In the above we use $\pi\left(\theta\right)$ as shorthand for $\pi\left(\mathbf{s}^\alpha, \mathbf{a}^\alpha; \theta\right)$ with $\hat{v}$ representing some unknown function. Using the definition of the Hessian we have:

$$\mathbf{H}_v \triangleq \left[\frac{\partial^2 v}{\partial\theta^a \partial\theta^b}\right]_{a,b=1,\ldots,D} = \left[\frac{\partial^2}{\partial\theta^a \partial\theta^b}\ \hat{v}\left(\pi\left(\theta\right)\right)\right]_{a,b=1,\ldots,D}$$

Where the above identity follows from the definition of v in Eq. B.1. We can now apply chain rule to express:

$$\frac{\partial^2}{\partial\theta^a \partial\theta^b}\ \hat{v}\left(\pi\left(\theta\right)\right) = \underbrace{\left[\mathbf{H}_{\hat{v}}(\pi(\theta))\frac{\partial\pi}{\partial\theta^a}(\theta)\right] \cdot \frac{\partial\pi}{\partial\theta^b}(\theta)}_{r(\theta)} + \underbrace{\frac{\partial^2\pi}{\partial\theta^a\partial\theta^b}\left(\theta\right)}_{\mathbf{H}_\pi(\theta)} \cdot \underbrace{\nabla\hat{v}(\pi(\theta))}_{g(\theta)} \tag{B.2}$$

As we see in the above as a consequence of the chain rule, $\frac{\partial^2\pi}{\partial\theta^a\partial\theta^b}$ forms an important sub-component $\frac{\partial^2 v}{\partial\theta^a\partial\theta^b}$. Given the above, we can simplify the above in the following manner:

$$\mathbf{H}_v = r + \mathbf{H}_\pi \circ g$$

where r, g, and $\mathbf{H}_\pi$ arise from the corresponding highlighted terms in Eq. B.2 with r representing some unknown remainder term and $\circ$ representing the Hadamard product. Given the above, it is straightforward to see how $\mathbf{H}_\pi$ serves as a surrogate hessian for $\mathbf{H}_v$. Indeed if $r \neq -\mathbf{H}_\pi \circ g$ and g has no zero entries then $\mathbf{H}_\pi \neq 0 \implies \mathbf{H}_v \neq 0$. In our use case, we are most concerned with non-zero entries in the Hessian, $\mathbf{H}_v$, and the surrogate Hessian, $\mathbf{H}_\pi$ is well served for determining $\mathbf{H}_v \neq 0$ due to the above.

Since $\pi(\theta)$ is shorthand for $\pi(\mathbf{s}^\alpha, \mathbf{a}^\alpha; \theta)$, to approximate $\mathbf{H}_\pi$ we average $\mathbf{H}_{\pi(\mathbf{s}^\alpha, \mathbf{a}^\alpha; \theta)}$ over state action pairs, $(\mathbf{s}^\alpha, \mathbf{a}^\alpha)$ formed through interaction of the policy with the unknown task environment.

A possible avenue of overcoming this limitation is considering Hessian estimation through zero'th order queries. Several works along this direction have recently appeared using Finite Differences (Cheng, G. Wu, and J. Zhu 2021), as well as Gaussian Processes (Müller, Rohr, and Trimpe 2021). We consider removing this dependency on the surrogate Hessian for future work.

B.8 Drone Delivery Task

Our drone delivery task was inspired by recent research work in studying unique problems in drone delivery vehicle routing problems (Dorling et al. 2017).

Drones fly from delivery point to delivery point where completing a delivery gives a large amount of reward, but running out of fuel and collisions give a small amount of negative reward. After completing a delivery, the delivery point is randomly removed within the environment. A collision gives a small amount of negative reward and momentarily stops the drone. Completing a delivery refills the drone fuel and allows it to continue to make more deliveries. The amount of reward given increases quadratically with the distance of the delivery to highly reward long distance deliveries which require long term planning. To compound this requirement for long term planning, fuel consumption also dramatically increases at high velocities to encourage long-term fuel efficiency planning. In this complex scenario requiring long term planning, RL approaches can easily fall into local minima of completing short distance, low reward deliveries and fail to sufficiently explore (under sparse reward) policies which complete long distance deliveries with careful planning.

Appendix C

Appendix for Chapter 6

C.1 Proof of Lemma 6.1

We restate Lemma 6.1 for clarity.

Lemma 6.1. *Let the reward function be defined on the compact domain* $[-b, +b]$. *Let the function* $M(\theta)$ *denote the number of local maxima in the interval* $(-\theta, +\theta)$. *We have* $\mathbb{E}_{\theta \sim \mathcal{U}(-b,+b)}[M(\theta)] = \Omega(b)$.

Proof. To prove the above Lemma, we first begin by differentiating the reward function. After this differentiation, we establish the number of zeros using the Intermediate Value Theorem. We analyze the critical points to isolate the local maximas by considering the change in sign of the first derivative. Lastly, we use Markov's inequality to complete the lemma.

By the application of the chain rule and division rule we have:

$$\frac{\partial}{\partial a}\frac{c_1 cos(a)}{e^{a^2}} = -c_1 e^{-a^2}(\sin a + 2a\cos a).$$

For local optima we require $-c_1 e^{-a^2}(\sin a + 2a\cos a) = 0$, which, for $c_1 > 0$ occurs when $\sin a + 2a\cos a = 0$.

For ease of analysis, we focus on a subset of the above zeros, specifically when $\sin a = 0$. This occurs when $a = n\pi$ for $n \in \mathbb{Z}$

Consider $n > 0$, we have for $n = 2k$ where $k \in \mathbb{N}$ $\sin n\pi + 2n\pi\cos n\pi > 0$ and for $n = 2k+1$ where $k \in \mathbb{N}$ we have $\sin n\pi + 2n\pi\cos n\pi < 0$.

By the Intermediate Value Theorem we have $\exists v \in (2k\pi, (2k+1)\pi)$ such that $-c_1 e^{-v^2}(\sin v + 2v\cos v) = 0$ and furthermore $\frac{\partial}{\partial v} - c_1 e^{-v^2}(\sin v + 2v\cos v) < 0$ which indicates a local maxima.

Similar reasoning applies for the case where $k < 0 \cap k \in \mathbb{Z}$. Therefore, in the interval $(-(2k+1)\pi, (2k+1)\pi)$ there exist $2k$ local maximas in the reward function.

To proceed with the analysis, we assert that we need to show $\mathbb{E}_{\theta\sim\mathcal{U}}(-b,+b)\left[M(\theta)\right] = \Omega(b)$. We will apply a transformation to prove the above.

Given that $\theta \sim \mathcal{U}(-b,+b)$, we have $P(\theta \leq \frac{-b}{2}) \cup P(\theta \geq \frac{b}{2}) = \frac{1}{2}$. Applying the transformation M to this equality and given that M is monotonically increasing, we have $P(M(\theta) \geq M(\frac{b}{2})) = \frac{1}{2}$. Given the bound on the number of local maximas we established earlier, we see that:

$$M\left(\frac{b}{2}\right) \geq \left\lfloor \frac{b-2\pi}{2\pi} \right\rfloor \tag{C.1}$$

where the factor of $\frac{1}{2}$ disappears due to considering the entire interval $(\frac{-b}{2}, \frac{b}{2})$. Substituting into the original probability equality we have:

$$P\left(M\left(\theta\right) \geq \left\lfloor \frac{b-2\pi}{2\pi} \right\rfloor\right) \geq \frac{1}{2} \tag{C.2}$$

We restate Markov's inequality for convenience, which we will then apply:

$$\mathbb{E}\left[M(\theta)\right] \geq aP(M(\theta) \geq a). \tag{C.3}$$

Substituting $a = \left\lfloor \frac{b-2\pi}{2\pi} \right\rfloor$ we have,

$$\mathbb{E}\left[M(\theta)\right] \geq \left\lfloor \frac{b-2\pi}{2\pi} \right\rfloor P\left(M(\theta) \geq \left\lfloor \frac{b-2\pi}{2\pi} \right\rfloor\right) \tag{C.4}$$

and finally, given that $P(M\left(\theta\right) \geq \left\lfloor \frac{b-2\pi}{2\pi} \right\rfloor) \geq \frac{1}{2}$ we have:

$$\mathbb{E}\left[M(\theta)\right] \geq \frac{1}{2}\left\lfloor \frac{b-2\pi}{2\pi} \right\rfloor \tag{C.5}$$

where the lemma is complete by choosing a suitable constant for the Ω function.

□

C.2 Proof of Lemma 6.2

We restate Lemma 6.2 for clarity.

Lemma 6.2. *Let the reward function be defined on the compact domain* $[-b, +b]$. *Let* $\hat{\theta}_1, \ldots, \hat{\theta}_n$ *be any sequence of local maxima of* $J(\cdot)$ *that are monotonically approaching the global maxima. That is* $sign(\hat{\theta}_1) =, \ldots, = sign(\hat{\theta}_n)$ *and* $|\hat{\theta}_1| >, \ldots, > |\hat{\theta}_n|$, $\frac{\partial J}{\partial \theta}(\hat{\theta}_1) =, \ldots, = \frac{\partial J}{\partial \theta}(\hat{\theta}_n) = 0$ *and* $\frac{\partial^2 J}{\partial \theta^2}(\hat{\theta}_1) < 0, \ldots, \frac{\partial^2 J}{\partial \theta^2}(\hat{\theta}_n) < 0$. *Then* $J(\hat{\theta}_i) < J(\hat{\theta}_j)$ *when* $i < j$.

Proof. To prove the above, we first analyze the critical points to isolate the local maximas, then we show that these local maximas are monotonically increasing as the origin is approached.

Recall from the proof of Lemma 6.1, the local optima require $\sin a + 2a \cos a = 0$. Using the identity $\sin^2 a + \cos^2 a = 1$ we have:

$$\sin a + 2a \cos a = 0 \implies \sin a = -2a \cos a \implies \sqrt{1 - cos^2 a} = -2a \cos a$$

$$\implies 1 - \cos^2 a = 4a^2 \cos^2 a \implies 1 = 4a^2 \cos^2 a + \cos^2 a \implies 1 = (1 + 4a^2) \cos^2 a$$

$$\implies cos^2 a = \frac{1}{1 + 4a^2} \implies \cos a = \pm \frac{1}{\sqrt{1 + 4a^2}}$$

We now show that the positive portion of the square root correspond exactly to local maxima using the second derivative test. Recall from Lemma 6.1 the first derivative is derived to be $-c_1 e^{-a^2}(\sin a + 2a \cos a)$. Therefore:

$$\frac{\partial}{\partial a} - c_1 e^{-a^2}(\sin a + 2a \cos a) = c_1 e^{-a^2} \left(4a^2 \cos a + 4a \sin a - 3 \cos a\right). \tag{C.6}$$

Using the equality $\sin a + 2a\cos a = 0$ to isolate the local optima we have:

$$c_1 e^{-a^2}\left(4a^2\cos a + 4a\sin a - 3\cos a\right) = c_1 e^{-a^2}\left(4a^2\cos a + 2a\sin a + 2a\sin a - 3\cos a\right)$$

$$= c_1 e^{-a^2}\left(2a(2a\cos a + \sin a) + 2a\sin a - 3\cos a\right) = c_1 e^{-a^2}\left(2a\sin a - 3\cos a\right).$$

Once again, at the local optima, we use the equality, $\sin a = -2a\cos a$:

$$c_1 e^{-a^2}\left(2a\sin a - 3\cos a\right) = c_1 e^{-a^2}\left(2a(-2a\cos a) - 3\cos a\right) =$$

$$c_1 e^{-a^2}\left(-4a^2\cos a - 3\cos a\right) = -c_1 e^{-a^2}\left((4a^2+3)\cos a\right).$$

Now that we have isolated the values of the second derivative at the local optima, recall that at the local optima we have:

$$\cos a = \pm\frac{1}{\sqrt{1+4a^2}}. \tag{C.7}$$

Note that for $a \in \mathbb{R}$, $\frac{1}{\sqrt{1+4a^2}} > 0$. Now, if $\cos a > 0$ (i.e., the positive portion of the square root), then $-c_1 e^{-a^2}\left((4a^2+3)\cos a\right) < 0$ which establishes a local maxima.

Therefore, by using the first and second derivative tests, we have established that at all local optima we have $\cos a = \frac{1}{\sqrt{1+4a^2}}$. We can substitute into the reward equation $\frac{c_1\cos a}{e^{a^2}}$. At all local optima we have:

$$J(\theta) = \frac{c_1}{e^{\theta^2}\sqrt{1+4\theta^2}}. \tag{C.8}$$

Suppose that $|\hat{\theta}_i| > |\hat{\theta}_j|$, then it is clear that since $e^{\hat{\theta_i^2}} > e^{\hat{\theta_j^2}}$ along with $\sqrt{1+4\hat{\theta}_i^2} > \sqrt{1+4\hat{\theta}_j^2}$ which naturally implies that $J(\hat{\theta}_i) < J(\hat{\theta}_j)$ which completes the lemma. □

C.3 Proof of Theorem 6.1

We restate Theorem 6.1 for convenience.

Theorem 6.1. *Let the reward function, and the policy space be defined as earlier, then the cumulative regret of the BO Algorithm GP-UCB suffers when searching for the optimal policy is* $\widetilde{O}(\sqrt{T}(c_1^2 \log T + \log^2 T))$.

We begin by restating Theorem 6.1 with more precision:

Theorem 6.1. *Given the reward function as defined earlier defined on some compact domain* $[-b, +b]$*, and using the RBF kernel with lengthscale* 1 *as the reference kernel. The policy function can be queried to yield a noisy observation* $y_t \triangleq J(\theta_t) + \epsilon_t$ *with the noise* ϵ_t *having zero mean conditioned on the history and is bounded by some constant* σ *almost surely. Running GP-UCB with prior* $GP(0, k(x, x'))$ *where* k *is the RBF kernel with lengthscale* 1*, noise model* $\mathcal{N}(0, \sigma^2)$*, and an appropriately picked* β_t *(exploration factor) we obtain a regret bound of* $\widetilde{O}(\sqrt{T}(c_1^2 \log T + \log^2 T))$.

Proof. The proof quickly follows from the bound on the RKHS norm defined earlier. Given that the RKHS norm on the unbounded (infinite) domain of the real line is established to be: $||J||^2_{\mathcal{H}_k} = 2c_1^2(1+e)\sqrt{\frac{1}{2e}}$, and from the domain restriction theorem for RKHS (Aronszajn 1950), the RKHS on the compact domain $[-b, +b]$ is upper bounded by $||J||^2_{\mathcal{H}_k}$. The theorem immediately follows as a consequence of Theorem 3 in (Srinivas et al. 2010) which establishes a straightforward link between the RKHS of the function being optimized, and the regret of the GP-UCB algorithm. □