DEAKIN UNIVERSITY

DOCTORAL THESIS

Thesis Title

Author:
Dat PHAN TRONG

Supervisor:
Prof. Sunil GUPTA

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

November 19, 2024

Declaration of Authorship

I, Dat PHAN TRONG, declare that this thesis titled, "Thesis Title" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism."

Dave Barry

Acknowledgements

The acknowledgments and the people to thank go here, don't forget to include your project advisor…

Contents

Declaration of Authorship **i**

Acknowledgements **iii**

Abbreviations **ix**

Abstract **1**

1 Introduction **3**
1.1 Aims and Approaches 6
1.2 Significant Contributions 6
1.3 Structure of this Thesis 7

2 Background **9**
2.1 Essential Mathematics Backgrounds 9
2.1.1 Sub-Gaussian Random Variables 9
2.1.2 Kernels 9
2.1.3 Maximum Information Gain 9
Entropy 9
Information Gain 9
Information Gain in Bayesian Optimization 10
Maximum Information Gain in Bayesian Optimization 10
2.1.4 Reproducing Kernel Hilbert Space (RKHS) 10
Hilbert Spaces and Inner Products 10
Kernel Functions and Positive Definiteness 11
Constructing RKHS: The Moore-Aronszajn Theorem 11
Inner Product and Norm in RKHS 11
Reproducing Property and Pointwise Evaluation 12
Representer Theorem 12
Regularization and Smoothness in RKHS 12
Examples of Kernels and Corresponding RKHS 12
2.2 Introduction to Optimization 13
2.2.1 Gradient-Based Optimization 13
2.2.2 Derivative-Free Optimization 13
2.3 Black-box Optimization 13
2.3.1 Introduction to Black-box Optimization 13
2.3.2 Surrogates Models 13
Gaussian Process 13
Deep Neural Network 13
2.3.3 Utility-Based Acquisition Functions 13
Upper Confidence Bound 13
Thompson Sampling 13
Probability of Improvement 13

Expected Improvement 13
2.3.4 Information-Theoretic Acquisition Functions 13
Entropy Search 13
Predictive Entropy Search 13
Max-value Entropy Search 13
Knowledge Gradient 13
2.3.5 Bandit-based Optimization 13
Multi-armed Bandits 13
Contextual Bandits 13
Linear Bandits 13
Non-Linear Bandits 13
2.3.6 Performance Metrics in Black-box Optimization 13
Simple Regret 13
Cumulative Regret 14
Comparison and Relevance 14
2.4 Black-box Optimization with Unknown Black-box Constraints 14
2.5 Black-box Optimization with Physics Information 14
2.6 Further Research in Black-box Optimisation 14

3 Neural-BO: A Black-box Optimization Algorithm using Deep Neural Networks **15**
3.1 Problem Setting 16
3.2 Proposed Neural-BO Method 16
3.3 Theoretical Analysis 18
3.4 Proof of the Main Theorem 19
3.5 Experiments 21
3.5.1 Experimental Setup 21
3.5.2 Synthetic Benchmarks 22
3.5.3 Real-world Applications 26
Designing Sensitive Samples for Detection of Model Tampering 26
Unknown target document retrieval 27
Optimizing control parameters for robot pushing 28
3.6 Conclusion 29

4 Black-box Optimization with Unknown Black-box Constraints via Overparametrized Deep Neural Networks **30**
4.1 Proposed Neural-CBO Method 31
4.1.1 The Deep Neural Network for an Arbitrary Function f_a 31
4.1.2 Neural-CBO Algorithm 34
4.2 Theoretical Analysis 35
4.2.1 Detailed Assumptions for Objective Function and Constraints . 35
4.3 Experiments 36
4.3.1 Baselines 36
4.3.2 Synthetic Benchmark Functions 36
4.3.3 Gas Transmission Compressor Design 37
4.3.4 Speed Reducer Design 38
4.3.5 Designing Sensitive Samples for Detection of Model Tampering 39
4.4 Conclusion 40

5 PINN-BO: A Black-Box Optimization Algorithm Using Physics-Informed Neural Networks 41
5.1 Problem Setting 41
5.2 Proposed PINN-BO Method 41
5.3 Theoretical Analysis 43
5.4 Experiments 47
5.4.1 Baselines 47
5.4.2 Synthetic Benchmark Functions 48
5.4.3 Real-world Applications 49
Optimizing Steady-State Temperature 49
Optimizing Beam Displacement 50
5.5 Conclusion 50

6 Conclusion 51
6.1 Contributions 51
6.2 Future Directions 52

A Supplementary Material of Chapter 3 53
A.1 Proof of Theoretical Analysis in Chapter 3 53
A.2 Proof of Auxiliary Lemmas 60
A.2.1 Proof of Lemma A.1.5.1 60
A.2.2 Proof of Lemma A.1.5.2 62
A.2.3 Proof of Lemma A.1.5.3 64
A.2.4 Proof of Lemma A.1.10.1 64

B Supplementary Material of Chapter 4 65

C Supplementary Material of Chapter 5 66
C.1 Proof of Theoretical Analysis in Chapter 5 66
C.1.1 Proof of Lemma 5.3.4 66
C.1.2 Proof of Lemma C.1.2 69

Bibliography 77

List of Figures

3.1 The plots show the minimum true value observed after optimizing several synthetic functions over 2000 iterations of our proposed algorithm and 6 baselines. The dimension of each function is shown in the parenthesis. . . . 23
3.2 The results of benchmarking our Neural-BO and the baselines with COCO framework on 24 BBOB noiseless objective functions with four different dimensions {2,3,5,10}. . . . 25
3.3 The plot shows the **detection rates** corresponding to the number of samples on the MNIST dataset. The larger the number of sensitive samples, the higher the detection rate. As shown in the figure, Neural-BO can generate sensitive samples that achieve nearly 90% of the detection rate with at least 8 samples. . . . 26
3.4 We search for the most related document for a specified target document in **Amazon product reviews** dataset and report the maximum **hierachical F1 score** found by all baselines. All methods show similar behaviour and Neural-BO performs comparably and much better than GP-based baselines. . . . 27
3.5 Optimization results for control parameters of 14D robot pushing problem. The X-axis shows iterations, and the y-axis shows the median of the best reward obtained. . . . 28

4.1 The plots show (Log10 of) the Best Positive Regret plus Violation up to step t, which is $\min_{t\in[T]}[f(\mathbf{x}_t) - f^*]^+ + \sum_{k=1}^{K}[c_k(\mathbf{x}_t)]^+]$, comparing our proposed algorithm and four baselines. The dimension of each objective function is shown in the parenthesis. The left group is four synthetic functions introduced in Section 4.3.2, while the right group is the optimization results of Gas Transmission Compressor Design and Speed Reducer Design, described in Section 4.3.3 and 4.3.4. . . . 38
4.2 **Detection Rates** corresponding to the number of samples on the MNIST dataset. As shown in the figure, Neural-CBO can generate sensitive samples that achieve nearly 85% of the detection rate with at least 10 samples. . . . 39

5.1 The optimization results for synthetic functions comparing the proposed PINN-BO with the baselines. The standard errors are shown by color shading. . . . 47
5.2 The optimization results for real-world applications comparing the proposed PINN-BO with the baselines. . . . 49

List of Tables

3.1 The p-values of KS-test "whether the data obtained from running our methods Neural-BO and all baselines are normally distributed". 24
3.2 One-sided t-tests were employed to assess whether the baseline achieves a lower function value compared to our proposed method, Neural-BO. The null hypothesis $H_0 : \mu_{\text{baseline}} \leq \mu_{\text{Neural-BO}}$ and the alternative hypothesis: $H_a : \mu_{\text{baseline}} > \mu_{\text{Neural-BO}}$. The p-value corresponding to each test is provided as the first value in each cell. Moreover, to account for multiple hypotheses testing, the Benjamini-Hochberg correction was applied and is reported as the second value in each cell. In the outcome, a "T" indicates that the null hypothesis is rejected, whereas an "F" signifies that it is not rejected. 25

4.1 The input dimension and number of constraints for each synthetic objective function. 37
4.2 One-sided t-tests to evaluate whether the baseline outperforms Neural-CBO in terms of best positive regret plus violation. 37

List of Abbreviations

BO	Black-box Optimization
DNN	Deep Neural Network
EI	Expected Improvement
ES	Entropy Search
PES	Predictive Entropy Search
MES	Max-value Entropy Search
KG	Knowledge Gradient
GP	Gaussian Process
LCB	Lower Confidence Bound
NTK	Neural Tangent Kernel
RKHS	Reproducing Kernel Hilbert Space
PDE	Partial Differential Equation
PINN	Physics-Informed Neural Network
TS	Thompson Sampling
UCB	Upper Confidence Bound
RF	Random Forest
BNN	Bayesian Neural Network
Deep GP	Deep Gaussian Process
MIG	Maximum Information Gain
ReLU	Rectified Linear Unit

List of Symbols

a	distance	m
P	power	W (J s^{-1})
ω	angular frequency	rad

For/Dedicated to/To my…

Abstract

Optimization has become an essential task across numerous fields, as many systems and processes require fine-tuning to achieve maximum efficiency. In many cases, these systems are "black-box", meaning that the underlying mechanisms are unknown or too complex to model explicitly; we can only provide inputs and observe the resulting outputs. The challenge with optimizing black-box systems is that evaluating them can often be costly, both in terms of time and resources. This creates a demand for optimization methods that are not only effective but also sample-efficient, minimizing the number of evaluations needed to find optimal solutions.

Bayesian optimization or Black-box Optimization (BO) has emerged as a leading framework for optimizing expensive and unknown functions in various fields, including materials science, biomedical research, and machine learning. BO typically relies on two core components: (1) a surrogate model that approximates the underlying function based on available observations, and (2) an acquisition function that balances the trade-off between exploration (searching unexplored regions with high uncertainty) and exploitation (refining known promising regions). Traditionally, Gaussian Process (GP) has been the preferred choice for the surrogate model due to its ability to quantify uncertainty and its analytic expression, hence being convenient to provide theoretical guarantees. However, GPs suffer from cubic computational complexity in relation to the number of observations, making them impractical for large-scale applications.

First of all, investigating alternative surrogate models to enhance the performance of standard BO, especially in scenarios where GP are computationally prohibitive, is focused. The ability of Deep Neural Networks (DNNs) to scale linearly with the number of data points and capture complex patterns makes them more suitable for tasks involving structural data (like images and text). We introduce **Neural-BO**, a DNN-based optimization algorithm that leverages recent advances in neural network theory to estimate uncertainty and uses Thompson Sampling (TS) for next point selection. This method maintains the flexibility of DNNs while achieving convergence guarantees through the Neural Tangent Kernel (NTK)-based theory, showing improved sample efficiency and faster convergence.

However, optimizing real-world functions often goes beyond simple maximization or minimization, requiring the incorporation of constraints. These constraints ensure that solutions not only optimize the objective but also meet critical feasibility criteria, which is especially important in domains like engineering and physics. Hence, we extend the DNN-based approach to handle optimization problems with expensive, unknown constraints through **Neural-CBO**. In this method, both the objective function and the constraints are modeled using DNNs, and the acquisition function is designed to balance exploration and exploitation while ensuring constraint satisfaction. By using Lower Confidence Bound (LCB) conditions to guarantee feasibility and the Expected Improvement (EI) acquisition function to guide the search, Neural-CBO efficiently explores the feasible region, even when it is significantly smaller than the overall search space. Our method provides upper bounds

on both regret and constraint violations, offering a theoretically sound and scalable approach for constrained black-box optimization tasks.

Moreover, in fields like physics, engineering, and biology, there is often domain-specific knowledge available, typically in the form of physical laws or models, which can be expressed as Partial Differential Equations (PDEs). Incorporating physical knowledge into the optimization process enhances function approximation by providing more accurate guidance, improving surrogate model predictions, and reducing uncertainty in regions that follow known physical laws. Therefore, we propose PINN-BO, an approach that integrates Physics-Informed Neural Network (PINN) into the BO framework. By incorporating these physical constraints into the optimization process, PINN-BO significantly improves sample efficiency. The method employs PINN to model the objective function, utilizing known physical principles by embedding the PDE into the network's training process. This ensures that the optimization process not only converges quickly but also remains aligned with the underlying physical system. Experimental results on both synthetic and real-world tasks demonstrate the superior performance of PINN-BO in domains where physical knowledge is crucial.

In summary, this thesis proposed three novel methods, each leveraging DNNs-based surrogate models to enhance computation complexity and sample efficiency in BO as well as its extended settings.

Chapter 1

Introduction

Science has made a great impact on human prosperity. From healthcare to food production to modern communication, scientific breakthroughs have transformed nearly every facet of life. Many of these breakthroughs arise from experiments conducted on complex, unknown systems, with the aim of discovering which inputs yield the most favorable outputs. Some well-known examples are the synthesis of short polymer fiber materials, alloy design, 3D bio-printing, molecule design, etc, (Greenhill et al., 2020; Shahriari et al., 2015). In many cases, experts rely on informed guesses or prior knowledge to decide which inputs to test. This trial-and-error process typically requires numerous experiments to identify the optimal input, a task that can be both time-consuming and expensive. The high costs, in terms of both financial resources and labor, restrict the extent to which these systems can be optimized, thereby slowing progress in research and industry. Therefore, strategies that help find optimal solutions with fewer trials could significantly speed up innovation.

Bayesian Optimization is one of the most effective methods for optimizing expensive, black-box systems (Mockus, Tiesis, and Zilinskas, 1978; Streltsov and Vakili, 1999). It works by using all previous inputs and their corresponding outputs, along with any known information about the system, to build a statistical model of the function. Typically, a probabilistic model such as a GP is trained on available observations. This model helps generate an acquisition function that balances exploration (trying new inputs) and exploitation (refining known good inputs). The acquisition function is optimized to suggest the next input to evaluate, and the resulting output is used to update the model. By iterating this process, Bayesian Optimization can identify optimal inputs with far fewer experiments than traditional methods, leading to significant cost savings in both time and resources.

Mathematically, we can formalize this task as a global optimization problem to optimize $f(\mathbf{x})$ subject to $\mathbf{x} \in \mathcal{D} \subset \mathbb{R}^d$, where d is the number of dimensions, and f is an expensive black-box system that can only be evaluated point-wise. Without any loss in generality, we assume $\mathcal{D}$ to be a d-dimensional space. Further, due to different aspects (e.g., measurement noise), we often only have access to noisy evaluations of f in the form $y = f(\mathbf{x}) + \epsilon$, where ϵ is the noise.

Bayesian Optimization has found widespread use across various disciplines, including materials science, biomedical research, and even other areas of computer science. In materials science, it has contributed to innovations such as the creation of new polymer fibers (Li et al., 2017), the analysis of metal oxide grain boundary structures (Kikuchi et al., 2018), and the enhancement of thermal conductivity in nanostructures (Ju et al., 2017). Within biomedical research, it has been applied to a range of tasks, such as aiding in COVID-19 diagnosis (Nour, Cömert, and Polat, 2020), examining the impact of aging on time perception (Turgeon, Lustig, and Meck, 2016), and designing synthetic genes (Gonzalez et al., 2015). In computer science, Bayesian optimization is frequently utilized to improve robot control (Berkenkamp, Krause, and Schoellig, 2023) and fine-tune hyperparameters in machine learning models (Snoek, Larochelle, and Adams, 2012; Bergstra and Bengio, 2012).

One of the most fundamental components in Bayesian Optimization is the *surrogate model*. As an approximation of the true objective function, the surrogate model provides both predictions of the function's value and estimates of uncertainty for each potential input. This dual output is crucial for designing the acquisition function, which determines the next point to sample in the optimization process. The acquisition function uses the surrogate model's predictions to balance exploration and exploitation. Exploration encourages sampling in regions with high uncertainty, where the model is less confident, in the hope of discovering better solutions. Exploitation, on the other hand, focuses on regions where the surrogate predicts high performance, aiming to refine promising areas. By leveraging the surrogate model's predictions and uncertainty estimates, the acquisition function can suggest new inputs that strategically advance the search for the optimal solution, often reducing the number of expensive evaluations required.

Due to their flexibility, well-calibrated uncertainty estimates, and favorable analytical properties, GP has long been a popular choice for modeling distributions over functions in Bayesian optimization (Osborne, Garnett, and Roberts, 2009). GP provide a probabilistic framework that captures not only the predictions of the objective function but also the uncertainty associated with those predictions, allowing Bayesian optimization to balance exploration and exploitation effectively. Additionally, the ability of GP to compute closed-form posterior distributions and acquisition functions, such as EI or Upper Confidence Bound (UCB), further enhances their appeal for efficient decision-making in optimization tasks. The combination of Bayesian optimization with GP has led to strong theoretical results, particularly in terms of convergence guarantees. For instance, Srinivas et al. (2009) presented rigorous theoretical bounds showing that Bayesian Optimization with GP models can achieve sublinear regret, which means that the performance of the algorithm improves significantly over time as more data is gathered. Moreover, this framework has been successfully extended to more complex settings, including multi-task and multi-objective optimization (Swersky, Snoek, and Adams, 2013), where GP can jointly model several related tasks or objectives, leveraging shared information to improve optimization efficiency across different tasks.

Despite these advantages, GP comes with inherent limitations that have motivated further research into alternative models. One major challenge is the computational complexity associated with GP, particularly the cubic time complexity for inference, which becomes prohibitive as the dataset grows. This computational burden is especially problematic when handling large-scale problems or high-dimensional input spaces, where GP struggle due to the curse of dimensionality. As the number of dimensions increases, the predictive performance of GP worsens, and the inference becomes slower, limiting their scalability for real-world applications involving large, high-dimensional datasets.

While GP remains the dominant choice for modeling in Bayesian optimization, in response to these challenges, researchers are actively exploring more alternative models to GP, such as Random Forest (RF), Bayesian Neural Network (BNN), and Deep Gaussian Process (Deep GP). These models aim to retain some of the desirable properties of GP, such as uncertainty quantification and flexibility while improving scalability and performance in high-dimensional spaces. For instance, Random Forests (RFs) and DNNs have been employed to estimate black-box functions in this context. RFs tend to perform well at making accurate predictions in the vicinity of the training data. However, they struggle with extrapolation, meaning their performance can significantly deteriorate when making predictions outside the range of observed data points (Shahriari et al., 2015). Additionally, hybrid approaches that combine GP with other models or approximate inference techniques, such as variational methods (Tran, Ranganath, and Blei, 2016) and sparse approximations (Snelson and Ghahramani, 2007), are being developed to tackle the computational bottlenecks and extend the practical applicability of GP and Bayesian Optimization to more complex and large-scale problems. The first significant application of DNN in Bayesian Optimization was introduced by Snoek et al. (2015), who employed a DNN to transform high-dimensional inputs into lower-dimensional feature vectors. These feature vectors were then fitted to a Bayesian linear regression surrogate model, allowing for more efficient optimization. Another promising approach was presented in Springenberg et al. (2016), where the objective function was modeled directly using a Bayesian Neural Network (BNN). This method aimed to capture the complex relationships within the data while providing uncertainty estimates for the predictions. Despite the empirical success of these neural network approaches in Bayesian optimization tasks, a significant gap remains in the theoretical understanding of their convergence properties. Unlike GPs, which have well-established theoretical frameworks demonstrating their convergence guarantees, the algorithms leveraging neural networks lack similar assurances. This absence of theoretical backing raises questions about the reliability and robustness of these methods in practice, particularly in high-dimensional applications. Moreover, the use of Bayesian linear regression or BNNs introduces substantial computational challenges. Both approaches require maintaining an ensemble of models to derive the posterior distribution over the parameters, leading to increased complexity and potentially prohibitive computational costs. As the dimensionality of the input space or the size of the dataset grows, the demand for computational resources can become a critical limitation. Consequently, while alternative models like random forests and neural networks present exciting avenues for exploration in Bayesian optimization, further research is *necessary to develop* methods that not only demonstrate empirical efficacy but also provide the theoretical guarantees and computational efficiency needed for practical application.

Real-world optimization problems often come with *unknown constraints*. In this setting, black-box optimization with black-box constraints becomes crucial, as both the objective function and constraints need to be optimized simultaneously while ensuring that evaluations are made within feasible regions. Traditional BO methods, which focus solely on optimizing the objective function, fail to capture the interplay between the unknown constraints and the objective, potentially leading to infeasible or suboptimal solutions. Hence, considering only standard BO is insufficient to address such tasks effectively. Many attempts have been made to tackle this challenge by incorporating constraint handling into the BO framework, resulting in methods specifically designed for constrained BO(Gelbart, Snoek, and Adams, 2014; Ariafar et al., 2019; Nguyen et al., 2023). However, despite these advancements, all of these methods still rely on GPs as the surrogate model, which introduces drawbacks similar to those seen in standard GP-based BO, such as poor scalability and increasing computational costs as the number of observations grows. This limitation motivates further *exploration* into more scalable surrogate models, such as DNNs, to better handle the complexities of black-box optimization with black-box constraints.

Additionally, these constraints could represent physical laws, safety conditions, or other complex requirements that must be satisfied during the optimization process. In many scientific and engineering domains, the objective function and its constraints are governed by well-established physical principles, such as conservation laws, thermodynamics, or fluid dynamics. These principles can often be expressed in the form of PDEs or other mathematical models that describe the underlying system behavior. *Integrating this prior knowledge into the BO framework can significantly improve both the efficiency and accuracy of the optimization process*. By leveraging physical knowledge, we not only reduce the need for expensive function evaluations but also ensure that the optimization process remains within physically feasible regions, thereby avoiding solutions that violate fundamental laws. Following these motivations, we introduce our main aims and approaches

1.1 Aims and Approaches

The main objective of this thesis is to improve the performance and scalability of BO, especially on structural data. More specifically, we focus on these targets:

1. As the number of optimization iterations increases, GPs-based black-box optimization faces computational costs, leading to poor scalability. To address this, we propose replacing the traditional GP surrogate model with a more scalable alternative – DNN.

2. Applying the use of DNN surrogate models into black-box optimization with black-box, expensive constraints.

3. Improving the performance of black-box optimization by integrating useful physical information in the form of PDEs.

To realize these aims, we developed our methods around the idea of using DNN as the surrogate model. Our works can be summarized as:

- To achieve Aim 1, we replace the conventional GP model by a DNN surrogate model. We apply a Thompson Sampling based strategy for choosing the next evaluation. We named this method as Neural-BO.

- To achieve Aim 2, we apply the Expected Improvement (EI) acquisition function to select the next samples within a feasible region, determined by Lower Confidence Bound (LCB) conditions for all constraints. The LCB-based approach guarantees constraint feasibility, while EI efficiently balances exploration and exploitation, especially when the feasible regions are much smaller than the overall search space. We called this method as Neural-CBO.

- To achieve Aim 3, we model the objective function using a Physics-Informed Neural Network (PINN). By incorporating physical knowledge, expressed through PDEs, into the PINN training process, we enhance the sample efficiency of the black-box optimization. We called this method PINN-BO.

1.2 Significant Contributions

The work presented in this thesis is important as it addresses the scalability limitations found in GPs-based Black-box Optimization (BO). As BO is designed for high-cost problems, our approach significantly lowers the computational cost across a wide range of practical tasks. Specifically:

- As GP scale poorly with the number of data points, GPs-based BO have to deal with the increasing computational cost when the number of optimization iterations increases. DNNs-based approaches, such as methods, that rely on BNNs to manage predictive uncertainty, continue to face computationally efficient challenges. Our Neural-BO algorithm, in contrast, incorporated recent advances in neural network theory through the use of NTK to estimate a confidence interval around the neural network model and employ Thompson Sampling technique to determine the next evaluation point. Our theoretical analysis of the algorithm's regret bound demonstrates that it is both convergent and more sample-efficient than existing methods. Furthermore, we show the potential of our Neural-BO algorithm in optimizing complex structured data, such as images and text, while maintaining computational scalability by growing linearly with the number of data points. The experimental results on synthetic benchmarks and real-world optimization tasks highlight that our algorithm outperforms the current state-of-the-art in BO.
- In real-world life, BO tasks often come along with unknown, expensive constraints. Our Neural-CBO has extended the DNN-based idea to this unknown, expensive constraints BO problem. By inheriting the good scalability of DNNs, we modeled both the objective function and constraints using DNNs, and EI is used as the acquisition function. The feasible region is determined using LCB conditions, ensuring constraint satisfaction while balancing exploration and exploitation. Our theoretical analysis shows that cumulative regret and constraint violations have upper bounds comparable to Gaussian Process-based methods. Importantly, the convergence of our model only requires network width to scale linearly with the number of observations. Benchmarking experiments on synthetic and real-world tasks demonstrate that Neural-CBO performs competitively with state-of-the-art techniques.
- Many natural phenomena are governed by physical laws, which can be viewed as constraints when modeling these phenomena as underlying black-box functions. Our contribution focuses on the BO problem, where the black-box function is subject to physical constraints expressed through PDEs. To address this, we introduce the PINN-BO algorithm, which offers several advantages: enhanced sample efficiency in optimization, scalable computation that grows linearly with the number of function evaluations, and the ability to handle a wide variety of PDEs. We demonstrate that PINN-BO outperforms existing methods that lack physical knowledge across both synthetic and real-world problems.

1.3 Structure of this Thesis

- Chapter 2 provides the foundational concepts for our works. We begin by introducing the optimization problem and its variants, from gradient-based approaches to derivative-free strategies under black-box assumptions on the objective functions. Then we describe black-box optimization and its core components, including surrogate models and acquisition functions. The surrogate models relied on key mathematical tools, such as GP and the associated kernels, along with DNN, particularly in over-parameterized settings. We then link these models through the lens of RKHS. Next, we introduce the trade-off between exploration-exploitation through acquisition functions, with a focus on utility-based acquisition functions (e.g., TS, EI, and UCB). Additionally, we discuss other types of acquisition functions, such as those based on information theory (Entropy Search (ES), Predictive Entropy Search (PES), Max-value Entropy Search (MES)). Finally, we discuss Bandit-based Optimization and establish a connection with Black-box Optimization.

- Chapter 3 describes our DNN-based black-box optimization method named Neural-BO. We begin by discussing the motivations behind the approach and explain how to mathematically integrate a DNN model with TS. Next, we present proof of regret-bound convergence for our proposed method under canonical assumptions. Finally, we provide experimental results on both real-world and synthetic problems, demonstrating that our method outperforms standard approaches, especially in high-dimensional structural data.

- Chapter 4 introduces Neural-CBO method. In this chapter, we utilized the benefit of DNN-based surrogate model into black-box optimization under unknown, black-box constraints. We detail how to use DNN to model, control the uncertainty and balance the exploration-exploitation of both objective function and constraints. Finally, we provide the theoretical guarantee not only for the cumulative regret of the objective function but also for the constraint violation. Similar to Chapter 3, we conduct a wide range of experiments, from synthetic to real-world benchmarks, to demonstrate the effectiveness of our method.

- Chapter 5 focuses on the PINN-BO method, which addresses optimization problems where the objective function is governed by physical knowledge, expressed as physics-based constraints. We begin by outlining the physical information in the form of PDEs and then describe how to incorporate this information into the optimization process. The chapter also provides a theoretical analysis of the convergence of this framework, leveraging the properties of this PINN-BO framework, leveraging the properties of PINN under NTK perspective.

- Chapter 6 concludes this thesis with a summary of our research and a discussion of potential future works.

Chapter 2

Background

2.1 Essential Mathematics Backgrounds

2.1.1 Sub-Gaussian Random Variables

2.1.2 Kernels

2.1.3 Maximum Information Gain

In the context of machine learning and information theory, Maximum Information Gain (MIG) is a principle used to select features, queries, or actions that yield the highest amount of information about an unknown variable of interest. The concept of information gain is rooted in Shannon's entropy theory and measures the reduction in uncertainty about a random variable after observing another variable.

Entropy

Entropy is a measure of the uncertainty or unpredictability in a random variable. For a discrete random variable X with possible outcomes $\mathbf{x} \in \mathcal{X}$ and probability distribution $p(X)$, the entropy $H(X)$ is defined as:

$$H(X) = -\sum_{\mathbf{x}\in\mathcal{X}} p(\mathbf{x}) \log p(\mathbf{x}).$$

This entropy value represents the average number of bits required to encode information about the variable X. When we consider a second random variable Y, we can examine how much observing Y reduces uncertainty about X, which is the basis for defining information gain.

The *conditional entropy* of X given Y, denoted $H(X|Y)$, measures the remaining uncertainty about X after observing Y. It is defined as:

$$H(X|Y) = -\sum_{\mathbf{y}\in\mathcal{Y}} p(\mathbf{y}) \sum_{\mathbf{x}\in\mathcal{X}} p(\mathbf{x}|\mathbf{y}) \log p(\mathbf{x}|\mathbf{y}).$$

The conditional entropy quantifies the uncertainty in X given that we know Y. If observing Y provides information about X, we expect $H(X|Y) < H(X)$.

Information Gain

The *information gain* $IG(X;Y)$ between two random variables X and Y is defined as the reduction in entropy of X after observing Y, and is given by:

$$IG(X;Y) = H(X) - H(X|Y).$$

This quantity represents the average reduction in uncertainty about X provided by knowledge of Y. Alternatively, it can also be expressed in terms of the mutual information $I(X;Y)$ between X and Y:

$$IG(X;Y) = I(X;Y),$$

where mutual information is defined as:

$$I(X;Y) = \sum_{\mathbf{x}\in\mathcal{X}} \sum_{\mathbf{y}\in\mathcal{Y}} p(\mathbf{x},\mathbf{y}) \log \frac{p(\mathbf{x},\mathbf{y})}{p(\mathbf{x})p(\mathbf{y})}.$$

Mutual information is symmetric and non-negative, i.e., $I(X;Y) = I(Y;X) \geq 0$, and measures the amount of shared information between X and Y.

Information Gain in Bayesian Optimization

Assume we have a GP prior over f with mean function $\mu(\mathbf{x})$ and covariance $k(\mathbf{x},\mathbf{x}')$. After t steps, the model receives an input sequence $\mathcal{X}_t = (\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_t)$ and observes noisy rewards $\mathbf{y}_t = (y_1, y_2, \ldots, y_t)$. The *information gain* at step t, quantifies the reduction in uncertainty about f after observing $\mathbf{y}_t$, defined as the mutual information between $\mathbf{y}_t$ and f:

$$I(\mathbf{y}_t; f) := H(\mathbf{y}_t) - H(\mathbf{y}_t|f),$$

where H denotes the entropy. To obtain the closed-form expression of information gain, one needs to introduce a GP model where f is assumed to be a zero mean GP indexed on $\mathcal{X}$ with kernel k. Let $\mathbf{f}_t = (f(\mathbf{x}_1), f(\mathbf{x}_1), \ldots, f(\mathbf{x}_t)])$ be the corresponding true function values. From Cover (1999), the mutual information between two multivariate Gaussian distributions is:

$$I(\mathbf{y}_t; f) = I(\mathbf{y}_t; \mathbf{f}_t) = \frac{1}{2} \log\det\Big(\mathbf{I} + \lambda^{-1}\mathbf{K}_t\Big),$$

where $\lambda > 0$ is a regularization parameter, $\mathbf{K}_t$ is the covariance kernel matrix of the points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, and $\mathbf{I}$ is the identity matrix.

Maximum Information Gain in Bayesian Optimization

To further achieve input-independent and kernel-specific bounds, the *Maximum Information Gain*, denoted as γ_t, after t steps, defined as:

$$\gamma_t := \max_{\mathcal{A}\subset\mathcal{X}, |A|=t} I(\mathbf{y}_A; \mathbf{f}_A)$$

2.1.4 Reproducing Kernel Hilbert Space (RKHS)

RKHSs provide a rigorous framework for dealing with functions in high-dimensional spaces using kernel methods, which are central in machine learning, functional analysis, and statistics. Formally, RKHSs are Hilbert spaces associated with a positive definite kernel, allowing efficient manipulation of functions via inner products and enabling regularization in infinite-dimensional settings.

Hilbert Spaces and Inner Products

A *Hilbert space* $\mathcal{H}$ is a complete vector space equipped with an inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ such that every Cauchy sequence in $\mathcal{H}$ converges in $\mathcal{H}$. Completeness is crucial for ensuring that limits of function sequences (e.g., solutions to optimization problems) lie within the space.

For an inner product space $\mathcal{H}$, the norm induced by the inner product is given by:

$$\|f\|_{\mathcal{H}} = \sqrt{\langle f, f \rangle_{\mathcal{H}}}.$$

Kernel Functions and Positive Definiteness

Let $\mathcal{X}$ be an input space, typically $\mathbb{R}^d$, and let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a *kernel function*. A function k is called *positive definite* if for any finite set of points $\{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subset \mathcal{X}$ and any $\mathbf{c} = (c_1, \dots, c_n) \in \mathbb{R}^n$, we have:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0.$$

This property ensures that k can serve as a similarity measure and enables the construction of a unique RKHS associated with k.

Constructing RKHS: The Moore-Aronszajn Theorem

A central result in RKHS theory is the *Moore-Aronszajn Theorem*, which guarantees the existence of a unique RKHS for any positive definite kernel k. The theorem states:

Theorem 2.1.1 (Moore-Aronszajn). *For any positive definite kernel k on $\mathcal{X} \times \mathcal{X}$, there exists a unique Hilbert space $\mathcal{H}$ of functions $f : \mathcal{X} \to \mathbb{R}$ such that:*

1. *$k(\mathbf{x}, \cdot) \in \mathcal{H}$ for all $\mathbf{x} \in \mathcal{X}$,*
2. *For every $f \in \mathcal{H}$ and $\mathbf{x} \in \mathcal{X}$, $f(\mathbf{x}) = \langle f, k(\mathbf{x}, \cdot) \rangle_{\mathcal{H}}$, called the reproducing property.*

This theorem allows us to construct $\mathcal{H}$ as the *completion* of the span of the functions $\{k(\mathbf{x}, \cdot) : \mathbf{x} \in \mathcal{X}\}$ with respect to the inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$.

Inner Product and Norm in RKHS

Given two functions $f, g \in \mathcal{H}$ represented as finite linear combinations of kernel functions, i.e., $f = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \cdot)$ and $g = \sum_{j=1}^{m} \beta_j k(\mathbf{y}_j, \cdot)$, the inner product in $\mathcal{H}$ can be computed as:

$$\langle f, g \rangle_{\mathcal{H}} = \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_i \beta_j k(\mathbf{x}_i, \mathbf{y}_j).$$

The corresponding norm is:

$$\|f\|_{\mathcal{H}} = \sqrt{\langle f, f \rangle_{\mathcal{H}}} = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j)}.$$

This norm represents the “smoothness” or “complexity” of functions in $\mathcal{H}$, with higher values corresponding to more complex functions.

Reproducing Property and Pointwise Evaluation

A crucial property in RKHS is the *reproducing property*, which allows pointwise evaluation of functions $f \in \mathcal{H}$ using the inner product. For any $f \in \mathcal{H}$ and $\mathbf{x} \in \mathcal{X}$,

$$f(\mathbf{x}) = \langle f, k(\mathbf{x}, \cdot) \rangle_{\mathcal{H}}.$$

This property simplifies function evaluations to inner products, making it possible to evaluate functions at any point without explicitly knowing the form of f.

Representer Theorem

In many machine learning applications, we wish to minimize a regularized empirical risk functional of the form:

$$\min_{f \in \mathcal{H}} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) + \lambda \|f\|_{\mathcal{H}}^2,$$

where L is a loss function (e.g., squared loss for regression) and $\lambda > 0$ is a regularization parameter. The *Representer Theorem* asserts that the solution to this optimization problem can be represented as a linear combination of kernel evaluations at the training points:

$$f^*(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x}),$$

where the coefficients α_i can be found by solving a finite-dimensional problem. This reduces the complexity of optimization from the infinite-dimensional space $\mathcal{H}$ to a finite-dimensional space of size n, greatly simplifying computation.

Regularization and Smoothness in RKHS

The RKHS norm $\|f\|_{\mathcal{H}}$ provides a measure of the smoothness of f. When we regularize by minimizing $\|f\|_{\mathcal{H}}^2$, we control the complexity of the function, effectively penalizing highly oscillatory or rough functions. This form of regularization is particularly relevant in *kernel ridge regression* and *support vector machines*, where the objective function is minimized subject to a smoothness constraint in $\mathcal{H}$.

Examples of Kernels and Corresponding RKHS

- **Linear Kernel** $k(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \mathbf{y}$: The RKHS corresponding to this kernel is the space of linear functions $f(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$.
- **Polynomial Kernel** $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y} + c)^p$: This kernel represents polynomial functions of degree p, enabling polynomial feature spaces.
- **Gaussian (RBF) Kernel** $k(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}\right)$: The RKHS for this kernel includes all smooth functions, making it particularly effective for non-linear, smooth function approximation.

2.2 Introduction to Optimization

2.2.1 Gradient-Based Optimization

2.2.2 Derivative-Free Optimization

2.3 Black-box Optimization

2.3.1 Introduction to Black-box Optimization

2.3.2 Surrogates Models

Gaussian Process

Deep Neural Network

2.3.3 Utility-Based Acquisition Functions

Upper Confidence Bound

Thompson Sampling

Probability of Improvement

Expected Improvement

2.3.4 Information-Theoretic Acquisition Functions

Entropy Search

Predictive Entropy Search

Max-value Entropy Search

Knowledge Gradient

2.3.5 Bandit-based Optimization

Multi-armed Bandits

Contextual Bandits

Linear Bandits

Non-Linear Bandits

2.3.6 Performance Metrics in Black-box Optimization

Evaluating the performance of a Black-box Optimization (BO) algorithm involves quantifying its efficiency in identifying the optimal solution of the objective function f. Two commonly used performance metrics in the literature are *simple regret* and *cumulative regret*, which capture different aspects of the optimization process.

Simple Regret

The *simple regret* measures the difference between the best function value obtained after t evaluations and the true global maximum. Formally, it is defined as:

$$r_t = f(\mathbf{x}^*) - f(\mathbf{x}_t^*),$$

where:

- $f(\mathbf{x}^*) = \max_{\mathbf{x}\in\mathcal{D}} f(\mathbf{x})$ is the global maximum of the objective function f,
- $\mathbf{x}_t^* = \arg\max_{i=1,\dots,t} f(\mathbf{x}_i)$ is the best input queried among the t evaluations.

Simple regret evaluates the quality of the best solution found so far without considering the total cost incurred during the optimization process. This metric is particularly relevant for applications where the goal is to identify a high-quality solution at the end of the optimization rather than during the intermediate steps.

Cumulative Regret

The *cumulative regret* quantifies the total loss incurred due to querying suboptimal points during the optimization process with optimization budget T. It is defined as:

$$R_T = \sum_{t=1}^{T} \left(f(\mathbf{x}^*) - f(\mathbf{x}_t) \right),$$

where $\mathbf{x}_t$ denotes the point queried at the t-th step.

Cumulative regret provides a measure of how efficiently the algorithm balances exploration and exploitation over t iterations. A lower cumulative regret indicates that the algorithm has made better use of its queries to approach the global optimum efficiently.

Comparison and Relevance

Simple regret and cumulative regret capture different trade-offs in BO. Simple regret focuses solely on the quality of the final solution and is particularly suited for offline optimization problems, where the cost of intermediate evaluations is less critical. In contrast, cumulative regret is more relevant in settings where every evaluation incurs a cost, such as online optimization, and emphasizes the efficiency of the entire optimization process.

Both metrics are essential for evaluating the performance of BO algorithms, offering complementary insights into their behavior and effectiveness across various applications.

2.4 Black-box Optimization with Unknown Black-box Constraints

2.5 Black-box Optimization with Physics Information

2.6 Further Research in Black-box Optimisation

Chapter 3

Neural-BO: A Black-box Optimization Algorithm using Deep Neural Networks

Gaussian Process (GP) is a natural and often the most prominent probabilistic model choice for BO. However, they are not well-suited for optimizing functions in complex, structured, high-dimensional spaces, such as those encountered in the optimization of images and text documents. This limitation arises because GPs rely on kernels to model functions, and widely used kernels, such as the Square Exponential and Matérn kernels, struggle to accurately capture similarities within complex structured data. Consequently, BO using GPs is limited in its applicability to fields like computer vision and natural language processing. Moreover, GPs face significant scalability challenges due to the cubic computational complexity of kernel inversion.

Deep Neural Networks (DNNs), by contrast, have demonstrated exceptional scalability and generalization capabilities, particularly for high-dimensional and structured data. These qualities make them a potential alternative to GPs for optimization tasks. Recent efforts to leverage DNNs in black-box function optimization and contextual bandits have shown promise. However, many existing approaches depend on computationally expensive Bayesian Neural Networks (BNNs), which require sampling-based methods to model predictive uncertainty, or they lack theoretical guarantees for convergence and sample efficiency.

To address these challenges, this chapter introduces Neural-BO, a novel framework for black-box optimization. In Neural-BO, the unknown function is modeled using an over-parameterized DNN, eliminating the need for a *Bayesian Neural Network (BNN)* and making the algorithm computationally favorable. The framework leverages recent advances in neural network theory, particularly Neural Tangent Kernel (NTK), to estimate confidence intervals around the neural network model. These confidence intervals are used within Thompson Sampling to draw random samples of the function, which are then maximized to determine the next evaluation point.

We analyze the theoretical properties of Neural-BO, providing a regret bound that establishes its convergence and improved sample efficiency compared to existing methods. This chapter begins with an outline of the problem setting and assumptions, followed by a detailed description of the Neural-BO framework and its theoretical analysis.

3.1 Problem Setting

We consider a global optimization problem to maximize $f(\mathbf{x})$ subject to $\mathbf{x} \in \mathcal{D} \subset \mathbb{R}^d$, where $\mathbf{x}$ is the input with d dimensions. The function f is a noisy black-box function that can only be evaluated via noisy evaluations of f in the form $y = f(x) + \epsilon$, where ϵ is a sub-Gaussian noise which we will discuss more clearly in Assumption 2 in our regret analysis section. In our work, we consider input space $\mathcal{D}$ of the form: $a \leq \|\mathbf{x}\|_2 \leq b$, where a, b are absolute constants and $a \leq b$. We note that this assumption is mild compared to current works in neural contextual bandits (Zhou, Li, and Gu, 2020; Zhang et al., 2021; Xu et al., 2020; Kassraie and Krause, 2022) that required $\|\mathbf{x}\|_2 = 1$.

To imply the smoothness of function f, we assume the objective function f is from an RKHS (See Section 2.1.4) corresponding to a positive definite NTK k_{NTK}. In particular, $\|f\|_{\mathcal{H}_{k_{\text{NTK}}}} \leq B$, for a finite $B > 0$. These assumptions are regular and have been typically used in many previous works (Chowdhury and Gopalan, 2017; Srinivas et al., 2009; Vakili, Khezeli, and Picheny, 2021).

3.2 Proposed Neural-BO Method

In this section, we present our proposed DNN-based black-box algorithm: Neural-BO. Following the principle of BO, our algorithm consists of two main steps: (1) Build a model of the black-box objective function, and (2) Use the black-box model to choose the next function evaluation point at each iteration. For the first step, unlike traditional BO algorithms which use GP to model the objective function, our idea is to use a fully connected neural network $h(\mathbf{x}; \boldsymbol{\theta})$ to learn the function f as follows:

$$h(\mathbf{x}; \boldsymbol{\theta}) = \sqrt{m}\mathbf{W}_L \phi(\mathbf{W}_{L-1}\phi(\cdots \phi(\mathbf{W}_1 \mathbf{x})),$$

where $\phi(x) = \max(x, 0)$ is the Rectified Linear Unit (ReLU) activation function, $\mathbf{W}_1 \in \mathbb{R}^{m \times d}, \mathbf{W}_i \in \mathbb{R}^{m \times m}, 2 \leq i \leq L-1, \mathbf{W}_L \in \mathbb{R}^{1 \times m}$, and $\boldsymbol{\theta} = (vec(\mathbf{W}_1), \cdots, vec(\mathbf{W}_L)) \in \mathbb{R}^p$ is the collection of parameters of the neural network, $p = md + m^2(L-2) + m$. To keep the analysis convenient, we assume that the width m is the same for all hidden layers. We also denote the gradient of the neural network by $\mathbf{g}(\mathbf{x}; \boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} h(\mathbf{x}; \boldsymbol{\theta}) \in \mathbb{R}^p$.

For the second step, we choose the next sample point $\mathbf{x}_t$ by using a Thompson Sampling strategy. In particular, given each $\mathbf{x}$, our algorithm maintains a Gaussian distribution for $f(\mathbf{x})$. To choose the next point, it samples a random function $\widetilde{f}_t$ from the posterior distribution $\mathcal{N}(h(\mathbf{x}, \boldsymbol{\theta}_{t-1}), \nu_t^2 \sigma_t^2(\mathbf{x}))$, where

$$\sigma_t^2(\mathbf{x}) = \lambda \mathbf{g}(\mathbf{x}; \boldsymbol{\theta}_0) \mathbf{U}_{t-1}^{-1} \mathbf{g}(\mathbf{x}; \boldsymbol{\theta}_0)/m,$$

and $\nu_t = \sqrt{2}B + \frac{R}{\sqrt{\lambda}}(\sqrt{2\log(1/\delta)}$. From here, $\sigma_t(\mathbf{x})$ and ν_t construct a confidence interval, where for all $\mathbf{x} \in \mathcal{D}$, we have $|f(\mathbf{x}) - f(\mathbf{x}, \boldsymbol{\theta})| \leq \nu_t \sigma_t(\mathbf{x})$ that holds with probability $1 - \delta$. We also note the difference from Zhang et al. (2021) that uses $\mathbf{g}(\mathbf{x}; \boldsymbol{\theta}_t)/\sqrt{m}$ as dynamic feature map, we here use a fixed feature map $\mathbf{g}(\mathbf{x}; \boldsymbol{\theta}_0)/\sqrt{m}$ which can be viewed as a finite approximation of the feature map $\phi(\mathbf{x})$ of k_{NTK}. This allows us to eliminate $\sqrt{m}$ away the regret bound of our algorithm. Then, the next evaluation is selected as $\mathbf{x}_t = \operatorname{argmax}_{\mathbf{x} \in \mathcal{D}} \widetilde{f}_t(\mathbf{x})$.

Besides, to initialize the network $h(\mathbf{x}; \boldsymbol{\theta}_0)$, we randomly generate each element of $\boldsymbol{\theta}_0$ from an appropriate Gaussian distribution: for each $1 \leq l \leq L-1$, each entry of $\mathbf{W}$ is generated independently from $\mathcal{N}(0, 2/m)$; while each entry of the last layer $\mathbf{W}_L$ is set to zero to make $h(\mathbf{x}, \boldsymbol{\theta}_0) = 0$. Here we inherit so-called *He initialization* (He et al., 2015), which ensures the stability of the expected length of the output vector at each hidden layer.

The proposed Neural-BO is summarized using Algorithm 1 and Algorithm 2. In section 3.3, we will demonstrate that our algorithm can achieve a sub-linear regret bound, and works well in practice (See our section 3.5).

Discussion A simple way to have a DNN-based black-box algorithm is to extend the existing works of Zhou, Li, and Gu (2020) and Zhang et al. (2021) or Kassraie and Krause (2022) to our setting where the search space is continuous. However, this is difficult because these algorithms are designed for a finite set of actions. A simple adaptation can yield non-vanishing bounds on cumulative regret. For example, the regret bound in Kassraie and Krause (2022) depends on the size of finite action spaces $|\mathcal{A}|$. If $|\mathcal{A}| \to \infty$ then the regret bound goes to infinity. In Zhou, Li, and Gu (2020) and Zhang et al. (2021), the regret bounds are built on gram matrix $\mathbf{H}$ which is defined through the set of actions. However, such a matrix cannot even be defined in our setting where the search space (action space) is infinite. We solve this challenge by using a discretization of the search space at each iteration as mentioned in Section 3.3. In addition, for our confidence bound calculation, using $\mathbf{g}(\mathbf{x}; \boldsymbol{\theta}_0)/\sqrt{m}$ instead of $\mathbf{g}(\mathbf{x}; \boldsymbol{\theta}_t)/\sqrt{m}$ as in Zhou, Li, and Gu (2020) and Zhang et al. (2021) allows us to reduce a factor $\sqrt{m}$ from our regret bound.

Our algorithm is also different from Chowdhury and Gopalan (2017) in traditional Bayesian optimization which uses a Gaussian process to model the function with posterior computed in closed form. In contrast, our algorithm computes posterior by gradient descent.

Algorithm 1 Neural Black-box Optimization (Neural-BO)

Input: The input space $\mathcal{D}$, the number of rounds T, exploration parameter ν_t, network width m, RKHS norm bound B, regularization parameter λ, $\delta \in (0, 1)$.

```
1: Set $\mathbf{U}_0 = \lambda \mathbf{I}$
2: for $t = 1$ to $T$ do
3:     $\sigma_t^2(\mathbf{x}) = \lambda \mathbf{g}(\mathbf{x}; \boldsymbol{\theta}_0)^\top \mathbf{U}_{t-1}^{-1} \mathbf{g}(\mathbf{x}; \boldsymbol{\theta}_0)/m$
4:     $\widetilde{f}_t(\mathbf{x}) \sim \mathcal{N}(h(\mathbf{x}; \boldsymbol{\theta}_{t-1}), \nu_t^2 \sigma_t^2(\mathbf{x}))$
5:     Choose $\mathbf{x}_t = \text{argmax}_{\mathbf{x} \in \mathcal{D}} \widetilde{f}_t(\mathbf{x})$ and receive observation $y_t = f(\mathbf{x}_t) + \epsilon_t$
6:     Update $\boldsymbol{\theta}_t \leftarrow \text{TrainNN}(\{\mathbf{x}_i\}_{i=1}^t \{y_i\}_{i=1}^t, \boldsymbol{\theta}_0)$
7:     $\mathbf{U}_t \leftarrow \mathbf{U}_{t-1} + \frac{\mathbf{g}(\mathbf{x}_t; \boldsymbol{\theta}_0) \mathbf{g}(\mathbf{x}_t; \boldsymbol{\theta}_0)^\top}{m}$
8: end for
```

Algorithm 2 TrainNN

Input: Chosen points $\{\mathbf{x}_i\}_{i=1}^t$, noisy observations $\{y_i\}_{i=1}^t$, initial parameter $\boldsymbol{\theta}_0$, number of gradient descent steps J, regularization parameter λ, network width m, learning rate η.

```
1: Define objective function $\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2} \sum_{i=1}^T (f(\mathbf{x}_i; \boldsymbol{\theta}) - y_i)^2 + \frac{1}{2} m\lambda \left\| \boldsymbol{\theta} - \boldsymbol{\theta}^{(0)} \right\|_2^2$
2: for $k = 0, \cdots, J-1$ do
       $\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \eta \nabla \mathcal{L}(\boldsymbol{\theta}^{(k)})$
3: end for
4: return $\boldsymbol{\theta}^{(J)}$.
```

3.3 Theoretical Analysis

In this section, we provide a regret bound for the proposed Neural-BO algorithm. Our regret analysis is built upon the recent advances in NTK theory (Allen-Zhu, Li, and Song, 2019; Cao and Gu, 2019) and proof techniques of GP-TS (Chowdhury and Gopalan, 2017). Unlike most previous neural-based contextual bandits works (Zhang et al., 2021; Zhou, Li, and Gu, 2020) that restrict necessarily the search space to a set $\mathbb{S}^{d-1} = \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\|_2 = 1\}$, we here derive our results for a flexible condition, where the norm of inputs are bounded: $a \leq \|\mathbf{x}\|_2 \leq b$, where $0 < a \leq b$.

For theoretical guarantees, we need to use the following assumptions on the observations $\{\mathbf{x}_i\}_{i=1}^T$.

Assumption 3.3.1. There exists $\lambda_0 > 0$, $\mathbf{H}_T \geq \lambda_0 \mathbf{I}_T$.

This is a common assumption in the literature (Zhou, Li, and Gu, 2020; Xu et al., 2020; Zhang et al., 2021), and is satisfied as long as we sufficiently explore the input space such that no two inputs $\mathbf{x}$ and $\mathbf{x}'$ are identical.

Assumption 3.3.2. We assume the noises $\{\epsilon_i\}_{i=1}^T$ where $\epsilon_i = y_i - f(\mathbf{x}_i)$, are i.i.d and sub-Gaussian with parameter $R > 0$ and are independent with $\{\mathbf{x}_i\}_{i=1}^T$.

This assumption is mild and similar to the work of Vakili et al. (2021) where the assumption is used to prove an optimal order of the regret bound for Gaussian process bandits. We use it here to provide a regret bound with the sublinear order for our proposed algorithm.

Now we are ready to bound the regret of our proposed Neural-BO algorithm. To measure the regret of the algorithm, we use the cumulative regret which is defined as follows: $R_T = \sum_{t=1}^T r_t$ after T iterations, where $\mathbf{x}^* = \mathrm{argmax}_{\mathbf{x} \in \mathcal{D}} f(\mathbf{x})$ is the maximum point of the unknown function f and $r_t = f(\mathbf{x}^*) - f(\mathbf{x}_t)$ is instantaneous regret incurred at time t. For further details on cumulative regret and its role as a performance metric, refer to Section 2.3.6. We present our main theoretical result.

Theorem 3.3.3. *Let $\delta \in (0,1)$. Assume that the true underlying f lies in the RKHS $\mathcal{H}_{k_{\mathrm{NTK}}}(\mathcal{D})$ corresponding to the NTK kernel $k_{\mathrm{NTK}}(\mathbf{x},\mathbf{x}')$ with RKHS norm bounded by B. Set the parameters in Algorithm 1 as $\lambda = 1 + \frac{1}{T}$, $\nu_t = \sqrt{2}B + \frac{R}{\sqrt{\lambda}}(\sqrt{2\log(1/\delta)}$, where R is the noise sub-Gaussianity parameter. Set $\eta = (m\lambda + mLT)^{-1}$ and $J = (1 + LT/\lambda)\left(1 + \log\left(T^3 L \lambda^{-1} \log\left(1/\delta\right)\right)\right)$. If the network width m satisfies:*

$$m \geq \mathrm{poly}\left(\lambda, T, \log(1/\delta), \lambda_0^{-1}\right),$$

then with probability at least $1-\delta$, the regret of Algorithm 1 is bounded as

$$\begin{aligned} R_T \leq\ & C_1(1+c_T)\nu_T\sqrt{L}\sqrt{\frac{\lambda B T}{\log(B+1)}(2\gamma_T+1)} \\ & + (4 + C_2(1+c_T)\nu_T L)\sqrt{2\log(1/\delta)T} + \frac{(2B+1)\pi^2}{6} + \frac{b(a^2+b^2)}{a^3} \end{aligned}$$

where C_1, C_2 are absolute constants, a and b are lower and upper norm bounds of $\mathbf{x} \in \mathcal{D}$ and $c_T = \sqrt{4\log T + 2\log|\mathcal{D}_t|}$.

Remark 3.3.4. There exists a component $\frac{b(a^2+b^2)}{a^3}$ in the regret bound R_T. This component comes from the condition of the search space. If we remove the influence of constants a, b, B and R, Theorem 3.3.3 implies that our regret bound follows an order $\widetilde{O}(\sqrt{T\gamma_T})$. This regret bound has the same order as the one of the Maximum Variance Reduction algorithm in Vakili et al. (2021), but is significantly better than traditional BO algorithms (e.g., (Srinivas et al., 2009; Chowdhury and Gopalan, 2017)) and existing neural contextual bandits (e.g., (Zhou, Li, and Gu, 2020; Zhang et al., 2021)) that have the order $\widetilde{O}(\gamma_T\sqrt{T})$. We note that the regret bounds in Zhou, Li, and Gu (2020) and Zhang et al. (2021) are expressed through the effective dimension $\widetilde{d} = \frac{\log\det(\mathbf{I}+\mathbf{H}/\lambda)}{\log(1+TK/\lambda)}$, where K is the number of arm in contextual bandits setting. However, γ_T and the effective dimension are of the same order up to a ratio of $1/(2\log(1+TK/\lambda))$. The improvement of factor $\sqrt{\gamma_T}$ is significant because for NTK kernel as shown by Kassraie and Krause (2022), γ_T follows order $\mathcal{O}(T^{1-1/d})$, where d is the input dimension. For $d \geq 2$, existing bounds become vacuous and are no longer sub-linear. In contrast, our regret bound remains sub-linear for any $d > 0$.

3.4 Proof of the Main Theorem

To perform analysis for continuous actions as in our setting, we use a discretization technique. At each time t, we use a discretization $\mathcal{D}_t \subset \mathcal{D}$ with the property that $|f(\mathbf{x}) - f([\mathbf{x}]_t)| \leq \frac{1}{t^2}$ where $[\mathbf{x}]_t \in \mathcal{D}_t$ is the closest point to $\mathbf{x} \in \mathcal{D}$. Then we choose $\mathcal{D}_t$ with size $|\mathcal{D}_t| = \left((b-a)BC_{\text{lip}}t^2\right)^d$ that satisfies $\|\mathbf{x} - [\mathbf{x}]_t\|_1 \leq \frac{b-a}{(b-a)BC_{\text{lip}}t^2} = \frac{1}{BC_{\text{lip}}t^2}$ for all $\mathbf{x} \in \mathcal{D}$, where $C_{\text{lip}} = \sup_{\mathbf{x}\in\mathcal{D}}\sup_{j\in[d]} \left(\frac{\partial^2 k_{\text{NTK}}(\mathbf{p},\mathbf{q})}{\partial \mathbf{p}_j \mathbf{q}_j} | \mathbf{p} = \mathbf{q} = \mathbf{x}\right)$. This implies, for all $\mathbf{x} \in \mathcal{D}$:

$$|f(\mathbf{x}) - f([\mathbf{x}]_t)| \leq \|f\|_{\mathcal{H}_{k_{\text{NTK}}}} C_{\text{lip}} \|\mathbf{x} - [\mathbf{x}]_t\|_1 \leq BC_{\text{lip}} \frac{1}{BC_{\text{lip}}t^2} = 1/t^2, \tag{3.1}$$

is Lipschitz continuous of any $f \in \mathcal{H}_{k_{\text{NTK}}}(\mathcal{D})$ with Lipschitz constant BC_{lip}, where we have used the inequality $\|f\|_{\mathcal{H}_{k_{\text{NTK}}}} \leq B$ which is our assumption about function f. We bound the regret of the proposed algorithm by starting from the instantaneous regret $r_t = f(\mathbf{x}^*) - f(\mathbf{x}_t)$ can be decomposed to $r_t = [f(\mathbf{x}^*) - f([\mathbf{x}^*]_t)] + [f([\mathbf{x}^*]_t - f(\mathbf{x}_t)]$. While the first term is bounded by Eqn.(3.1), we bound the second term in Lemma A.1.9 provided in our Appendix. The proof of Lemma A.1.9 requires a few steps - we introduce a saturated set $\mathcal{S}_t$ (see Definition A.2), we then combine it with the results of Lemma A.1.3 (deriving a bound on $|\widetilde{f}_t(\mathbf{x}) - h(\mathbf{x};\boldsymbol{\theta}_{t-1})|$) necessitated due to the Thompson sampling and Lemma A.1.4 (deriving a bound on $|f(\mathbf{x}) - h(\mathbf{x};\boldsymbol{\theta}_{t-1})|$) utilizing the generalization result of the over-parameterized neural networks. It is noted that in our proof, we consider the effect of our general search space setting (where input $\mathbf{x} \in \mathcal{D}$ is norm-bounded, i.e., $0 < a \leq \|\mathbf{x}\|_2 \leq b$) on the output of the over-parameterized deep neural network at each layer in Lemma A.1.2 and on the gradient of this network utilized in Lemma A.1.5.1. These results contribute to the appearance of the lower and the upper norm bounds a and b, the first in the generalization property of over-parameterized neural networks in Lemma A.1.4 and the latter affect the cumulative regret bound in Theorem 3.3.3. Our proof follows the proof style of Lemma 4.1 of Cao and Gu (2019), Lemma B.3 of Cao and Gu (2019) and Theorem 5 of Allen-Zhu, Li, and Song (2019).

On the other hand, Lemma A.1.5.2, based on Assumption 3.3.2, provides a tighter bound on the confidence interval that eliminated the factor $\sqrt{\gamma_T}$, in comparison with previous relevant works Chowdhury and Gopalan (2017) and Zhou, Li, and

Gu (2020), leads to the sub-linear regret bound in Theorem 3.3.3. By these arguments, we achieve a cumulative regret bound for our proposed algorithm in terms of the maximum information gain γ_T (Lemma A.1.8, A.1.9, A.1.10). **Our detailed proof is provided in Appendix A.1.**

Proof Sketch for Theorem 3.3.3. With probability at least $1-\delta$, we have

$$\begin{aligned}
R_T &= \sum_{t=1}^{T} f(\mathbf{x}^*) - f(\mathbf{x}_t) \\
&= \sum_{t=1}^{T} [f(\mathbf{x}^*) - f([\mathbf{x}^*]_t)] + [f([\mathbf{x}^*]_t) - f(\mathbf{x}_t)] \\
&\le 4T\epsilon(m) + \frac{(2B+1)\pi^2}{6} + \bar{C}_1(1+c_T)\nu_T\sqrt{L}\sum_{i=1}^{T}\min(\sigma_t(\mathbf{x}_t), B) \\
&\quad + (4 + \bar{C}_2(1+c_T)\nu_T L + 4\epsilon(m))\sqrt{2\log(1/\delta)T} \\
&\le \bar{C}_1(1+c_T)\nu_T\sqrt{L}\sqrt{\frac{\lambda BT}{\log(B+1)}(2\gamma_T+1)} + \frac{(2B+1)\pi^2}{6} + 4T\epsilon(m) \\
&\quad + 4\epsilon(m)\sqrt{2\log(1/\delta)T} + (4 + \bar{C}_2(1+c_T)\nu_T L)\sqrt{2\log(1/\delta)T} \\
&= \bar{C}_1(1+c_T)\nu_t\sqrt{L}\sqrt{\frac{\lambda BT}{\log(B+1)}(2\gamma_T+1)} + \frac{(2B+1)\pi^2}{6} \\
&\quad + \epsilon(m)(4T + \sqrt{2\log(1/\delta)T}) + (4 + \bar{C}_2(1+c_T)\nu_t L)\sqrt{2\log(1/\delta)T}
\end{aligned}$$

The first inequality is due to Lemma A.1.9, which provides the bound for cumulative regret R_T in terms of $\sum_{t=1}^{T}\min(\sigma_t(\mathbf{x}_t), B)$. The second inequality further provides the bound of term $\sum_{t=1}^{T}\min(\sigma_t(\mathbf{x}_t), B)$ due to Lemma A.1.10, while the last equality rearranges addition. Picking

$$\begin{aligned}
&\eta = (m\lambda + mLT)^{-1} \text{ and} \\
&J = (1 + LT/\lambda)\left(\log(C_{\epsilon,2}) + \log\left(T^3 L\lambda^{-1}\log(1/\delta)\right)\right),
\end{aligned}$$

we have

$$\begin{aligned}
&\frac{b}{a}C_{\epsilon,2}(1-\eta m\lambda)^J\sqrt{TL/\lambda}\left(4T + \sqrt{2\log(1/\delta)T}\right) \\
=&\frac{b}{a}C_{\epsilon,2}\left(1 - \frac{1}{1+LT/\lambda}\right)^J\left(4T + \sqrt{2\log(1/\delta)T}\right) \\
=&\frac{b}{a}C_{\epsilon,2}e^{-\left(\log(C_{\epsilon,2}) + \log\left(T^3L\lambda^{-1}\log(1/\delta)\right)\right)}\left(4T + \sqrt{2\log(1/\delta)T}\right) \\
=&\frac{b}{a}\frac{1}{C_{\epsilon,2}}.T^{-3}L^{-1}\lambda\log^{-1}(1/\delta)\left(4T + \sqrt{2\log(1/\delta)T}\right) \le \frac{b}{a}
\end{aligned}$$

Then choosing m that satisfies:

$$\left(\frac{b}{a}C_{\epsilon,1}m^{-1/6}\lambda^{-2/3}L^3\sqrt{\log m}+\left(\frac{b}{a}\right)^3 C_{\epsilon,3}m^{-1/6}\sqrt{\log m}L^4T^{5/3}\lambda^{-5/3}(1+\sqrt{T/\lambda})\right)\left(4T+\sqrt{2\log(1/\delta)T}\right)\leq\left(\frac{b}{a}\right)^3$$

We finally achieve the bound of R_T as:

$$R_T \leq \bar{C}_1(1+c_T)\nu_T\sqrt{L}\sqrt{\frac{\lambda BT}{\log(B+1)}(2\gamma_T+1)} + (4+\bar{C}_2(1+c_T)\nu_T L)\sqrt{2\log(1/\delta)T}+\frac{(2B+1)\pi^2}{6}+\frac{b(a^2+b^2)}{a^3}$$

□

3.5 Experiments

In this section, we demonstrate the effectiveness of our proposed Neural-BO algorithm compared to traditional BO algorithms and several other BO algorithms on various synthetic benchmark optimization functions and various real-world optimization problems that have complex and structured inputs. The real-world optimization problems consist of (1) generating sensitive samples to detect tampered model trained on MNIST (LeCun and Cortes, 2010) dataset; (2) optimizing control parameters for robot pushing considered in Wang and Jegelka (2017); and (3) optimizing the number of evaluations needed to find a document that resembles an intended (unknown) target document.

3.5.1 Experimental Setup

For all experiments, we compared our algorithm with common classes of surrogate models used in black-box optimization, including GPs, RFs and DNNs. For GPs, we have three baselines: GP-UCB (Srinivas et al., 2009), GP-TS (Chowdhury and Gopalan, 2017) and GP-EI (Jones, Schonlau, and Welch, 1998) with the common Squared Exponential (SE) kernel. Our implementations for GP-based Bayesian Optimization baselines utilize public library GPyTorch [1] and BOTorch [2]. Next, we consider SMAC (Hutter, Hoos, and Leyton-Brown, 2011) as a baseline using RF as a surrogate model. Lastly, we also include two recent DNNs-based works for black-box optimization: DNGO (Snoek et al., 2015) and Neural Greedy (Paria et al., 2022). We further describe the implementations for RF and DNN-based baselines as follows:

- RF-based surrogate model is implemented using public library GPyOpt [3] and optimization is performed with EI acquisition function.
- DNGO (Snoek et al., 2015) models the black-box function by a Bayesian linear regressor, using low-dimensional features (extracted by DNN) as inputs.

[1] `https://gpytorch.ai/`
[2] `https://botorch.org/`
[3] `https://github.com/SheffieldML/GPyOpt`

We run DNGO algorithm with the implementation of AutoML [4] with default settings.

- NeuralGreedy (Paria et al., 2022) fits a neural network to the current set of observations, where the function values are randomly perturbed before learning the neural network. The learned neural network is then used as the acquisition function to determine the next query point. Our re-implementation of NeuralGreedy follows the setting described in Appendix F.2 (Paria et al., 2022).

For our proposed Neural-BO algorithm, we model the functions using two-layer neural networks $f(\mathbf{x};\boldsymbol{\theta}) = \sqrt{m}\mathbf{W}_2\sigma(\mathbf{W}_1\mathbf{x})$ with network width $m = 500$. The weights of the network are initialized with independent samples from a normal distribution $\mathcal{N}(0, 1/m)$. To train the surrogate neural network models, we use SGD optimizer with batch size 50, epochs 50, learning rate $\eta = 0.001$ and regularizer $\lambda = 0.01$. We set exploration parameter in Algorithm 1: $\nu_t = \nu$ and do a grid search over $\{0.1, 1, 10\}$.

3.5.2 Synthetic Benchmarks

We optimized several commonly used synthetic benchmark functions: Ackley, Levy, and Michalewics. The exact expression of these functions can be found at: `http://www.sfu.ca/~ssurjano/optimization.html`. To emphasize the adaptation of our method to various dimensions, for each function, we performed optimization for four different dimensions, including 10, 20, 50, and 100. This is done to evaluate the methods across a range of dimensions. The function evaluation noise follows a normal distribution with zero mean and the variance is set to 1% of the function range.

[4] `https://github.com/automl/pybnn`

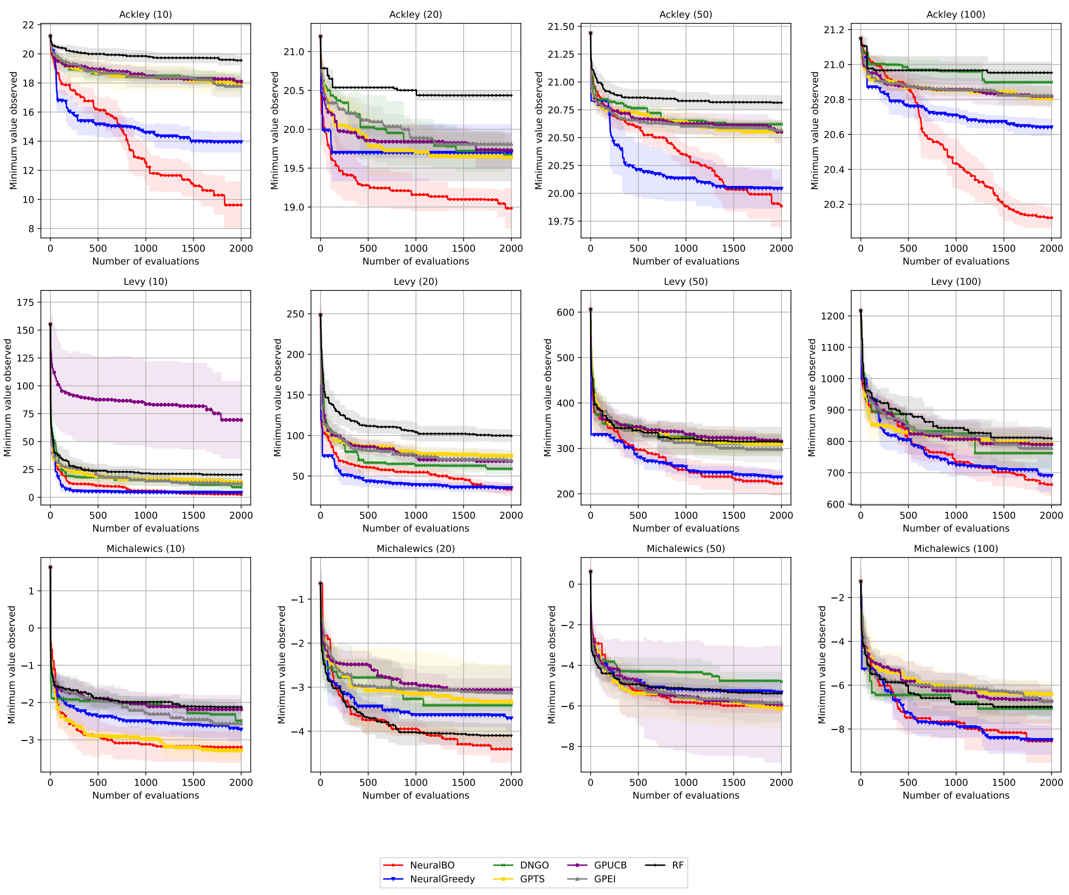


FIGURE 3.1: The plots show the minimum true value observed after optimizing several synthetic functions over 2000 iterations of our proposed algorithm and 6 baselines. The dimension of each function is shown in the parenthesis.

All reported experiments are averaged over 10 runs, each using a random initialization. All the methods start with the same set of initial points. As seen from Figure 3.1, Neural-BO is comparable to or better than all other baselines, including GP-based BO algorithms, both NN-based BO algorithms (DNGO and NeuralGreedy) and one algorithm using random forest. Moreover, our algorithm is also promising for high dimensions.

In order to assess the statistical significance of whether our proposed method outperforms all baselines, we conducted a series of tests. First, a Kolmogorov-Smirnov (KS) test was performed to check if the sample sets for the hypothesis test follow a normal distribution. The null hypothesis assumes no difference between the observed and normal distribution. The p-values obtained from the KS test are presented in Table 3.1. As all p-values exceeded 0.05, we failed to reject the null hypothesis, indicating that all data can be considered as normally distributed. Subsequently, one-sided t-tests were employed (as the variance is unknown), along with the Benjamini-Hochberg correction for multiple hypotheses testing, for each pair of (baseline, Neural-BO). These tests aimed to determine whether the baseline achieves a lower objective function value than our proposed method, Neural-BO. The alternative hypothesis $H_a : \mu_{\text{baseline}} > \mu_{\text{Neural-BO}}$ was tested against the null hypothesis $H_0 : \mu_{\text{baseline}} \leq \mu_{\text{Neural-BO}}$. Detailed test results are provided in Table 3.2, where each cell contains two values: the first value represents the p-value obtained from the t-test, and the second value (T or F) indicates the outcome of the Benjamini-Hochberg

		Neural -BO	Neural Greedy	GP -UCB	GP -TS	GP -EI	DNGO	RF
Ackley	D=10	0.85	0.83	0.95	0.95	0.45	0.23	0.79
	D=20	0.31	0.46	0.91	0.90	1.00	1.00	0.93
	D=50	0.95	0.38	0.81	0.99	0.52	0.85	0.67
	D=100	0.84	0.77	0.94	0.82	0.82	0.38	0.97
Levy	D=10	0.75	0.45	0.79	0.72	0.72	0.58	0.63
	D=20	0.84	1.00	1.00	0.79	0.79	0.91	0.89
	D=50	0.76	0.34	0.75	0.75	0.98	0.77	0.37
	D=100	0.76	0.59	0.87	0.90	0.99	0.93	0.95
Micha-lewics	D=10	0.54	0.56	0.95	0.88	0.83	0.77	0.55
	D=20	0.60	0.99	0.43	0.33	0.31	0.56	0.85
	D=50	0.90	0.80	0.64	0.39	0.68	0.99	0.95
	D=100	0.70	0.58	0.96	0.57	0.96	0.93	0.52

TABLE 3.1: The p-values of KS-test "whether the data obtained from running our methods Neural-BO and all baselines are normally distributed".

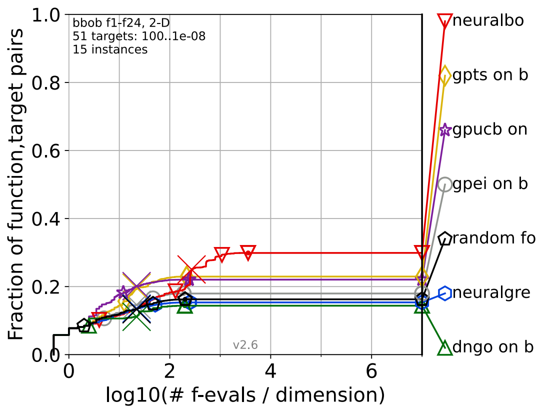
correction where "T" indicates that the null hypothesis is rejected, whereas an "F" indicates that it is not rejected.

Next, we used the COmparing Continuous Optimizers (COCO) framework[5] to compare the performance of our method with baseline methods. COCO is a widely used benchmarking platform for continuous optimization algorithms, providing researchers with a standardized set of test functions to evaluate different optimization methods. The framework includes a collection of test functions that are designed to be challenging to optimize and exhibit various properties, such as multimodality, ill-conditioning, or weak global structure. To evaluate our algorithms, we used the BBOB test functions, which consist of 24 noiseless, single-objective, and scalable functions. The quality indicator needs to reach or exceed a predefined target value t for COCO to consider a single run over a problem successful. The performance of different optimizers is measured by the *Fraction of function, target pairs*, which indicates the ratio of problems solved at optimization step T. For more details on how COCO measures the performance of different optimizers, please refer to Hansen et al. (2021). We set the number of evaluations for our optimizers to be 20 times the dimension of the problem.

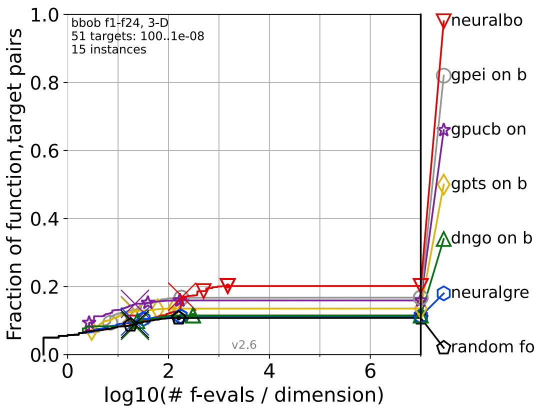
[5] `https://github.com/numbbo/coco`

		Neural Greedy	GPUCB	GPTS	GPEI	DNGO	RF
Ackley	D=10	(2.98e-07, T)	(3.11e-12, T)	(1.64e-11, T)	(2.3e-11, T)	(6.15e-09, T)	(2.09e-13, T)
	D=20	(6.39e-05, T)	(1.61e-06, T)	(4.72e-05, T)	(4.19e-08, T)	(0.000111, T)	(1.29e-05, T)
	D=50	(0.0467, T)	(2.23e-08, T)	(1.89e-08, T)	(8.5e-09, T)	(1.86e-08, T)	(2.11e-10, T)
	D=100	(3.92e-14, T)	(7.98e-16, T)	(2.43e-16, T)	(4.7e-16, T)	(6.44e-16, T)	(5.78e-17, T)
Levy	D=10	(0.000171, T)	(7.98e-06, T	(8.81e-09, T)	(6.6e-07, T)	(1.68e-05, T)	(1.62e-11, T)
	D=20	(0.278, F)	(3.28e-09, T)	(4.32e-11, T)	(1.8e-07, T)	(4.75e-05, T)	(8.01e-14, T)
	D=50	(0.0971, F)	(5.37e-09, T)	(2.72e-07, T)	(6.47e-06, T)	(0.000188, T)	(1.99e-06, T)
	D=100	(0.096, F)	(1.09e-06, T)	(4.85e-08, T)	(9.87e-05, T)	(0.00456, T)	(5.57e-07, T)
Micha-lewics	D=10	(5.9e-06, T)	(7.79e-08, T)	(0.472, F)	(1.68e-05, T)	(7.49e-06, T)	(1.87e-11, T)
	D=20	(2.63e-05, T)	(3e-09, T)	(0.00108, T)	(1.02e-05, T)	(0.000163, T)	(0.0161, T)
	D=50	(0.000344, T)	(3.47e-06, T)	(0.315, F)	(0.101, F)	(0.000871, T)	(0.000132, T)
	D=100	(0.45, F)	(8.2e-05, T)	(8.49e-06, T)	(0.000126, T)	(0.0404, T)	(0.00579, T)

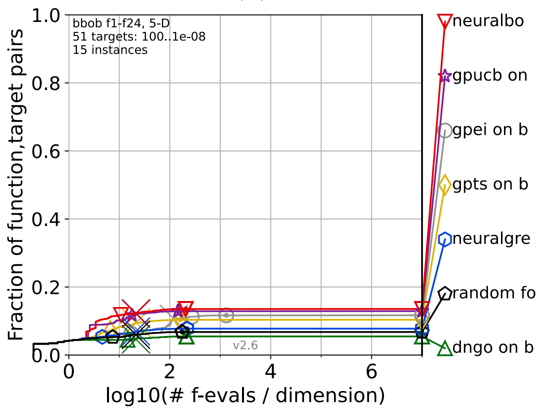
TABLE 3.2: One-sided t-tests were employed to assess whether the baseline achieves a lower function value compared to our proposed method, Neural-BO. The null hypothesis $H_0 : \mu_{\text{baseline}} \leq \mu_{\text{Neural-BO}}$ and the alternative hypothesis: $H_a : \mu_{\text{baseline}} > \mu_{\text{Neural-BO}}$. The p-value corresponding to each test is provided as the first value in each cell. Moreover, to account for multiple hypotheses testing, the Benjamini-Hochberg correction was applied and is reported as the second value in each cell. In the outcome, a "T" indicates that the null hypothesis is rejected, whereas an "F" signifies that it is not rejected.



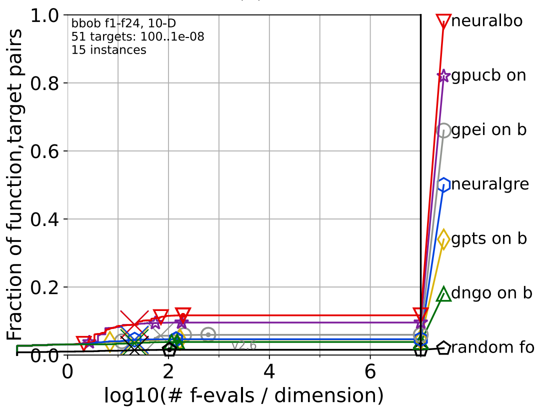
(A) 2D



(B) 3D



(C) 5D



(D) 10D

FIGURE 3.2: The results of benchmarking our Neural-BO and the baselines with COCO framework on 24 BBOB noiseless objective functions with four different dimensions {2,3,5,10}.

In Figure 3.2, we present the results of our experiment using the COCO benchmarking framework to evaluate all methods. The benchmarking comprised 24 noiseless functions with 15 instances, where each instance represented a unique condition of the function, such as the location of the optimum. Our method was found to outperform other baselines when assessed using the well-designed and publicly available COCO framework.

3.5.3 Real-world Applications

Designing Sensitive Samples for Detection of Model Tampering

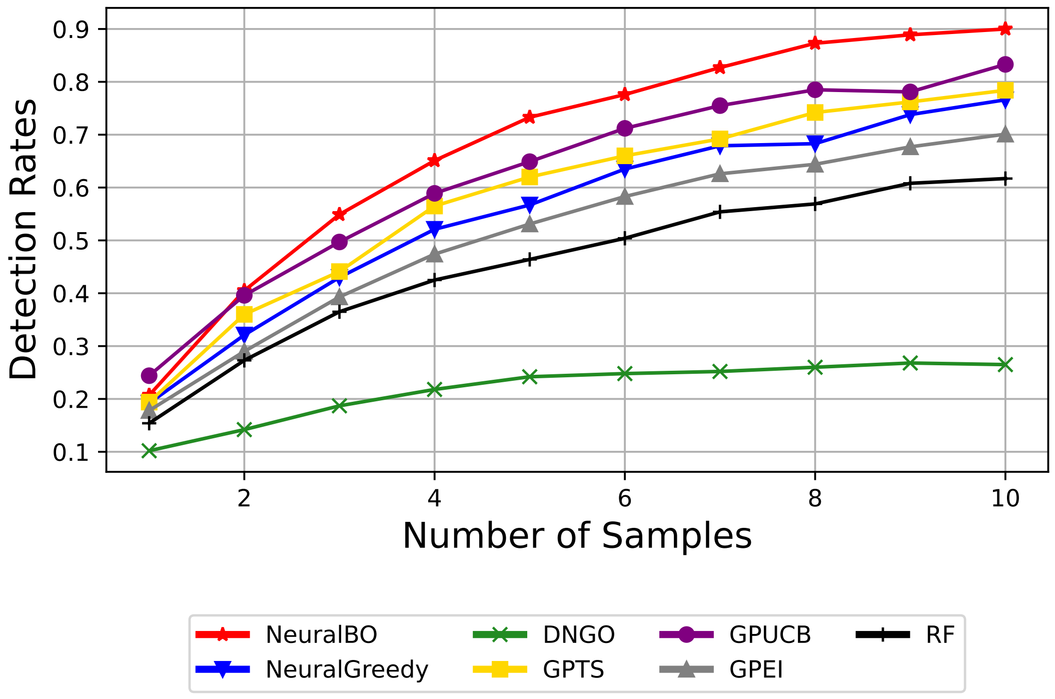


FIGURE 3.3: The plot shows the **detection rates** corresponding to the number of samples on the MNIST dataset. The larger the number of sensitive samples, the higher the detection rate. As shown in the figure, Neural-BO can generate sensitive samples that achieve nearly 90% of the detection rate with at least 8 samples.

We consider an attack scenario where a company offering Machine Learning as a service (MLaaS) hosts its model on a cloud. However, an adversary with backdoor access may tamper with the model to change its weight. This requires the detection of model tampering. To deal with this problem, He, Zhang, and Lee (2018) suggests generating a set of test vectors named *Sensitive-Samples* $\{v_i\}_{i=1}^n$, whose outputs predicted by the compromised model will be different from the outputs predicted by the original model. As formalized in He, Zhang, and Lee (2018), suppose we suspect a pre-trained model $f_\theta(x)$ of having been modified by an attacker after it was sent to the cloud service provider. Finding sensitive samples for verifying the model's integrity is equivalent to optimizing task: $v = \text{argmax}_x \left\| \frac{\partial f_\theta(x)}{\partial \theta} \right\|_F$, where $\|\cdot\|_F$ is the Frobenius norm of a matrix. A *successful detection* is defined as "given N_S sensitive samples, there is at least one sample, whose top-1 label predicted by the compromised model is different from the top-1 label predicted by the correct model". Clearly, optimizing this expensive function requires a BO algorithm to be able to work with high-dimensional structured images, unlike usual inputs that take values in hyper-rectangles.

We used a hand-written digit classification model (pre-trained on MNIST data) as the original model and tampered it by randomly adding noise to each weight of this model. We repeat this procedure 1000 times to generate 1000 different tampered models. The top-1 accuracy of the MNIST original model is 93% and is reduced to 87.73% $\pm$ 0.08% after modifications.

To reduce the computations, we reduce the dimension of images from 28×28 to 7×7 and do the optimization process in 49-dimensional space. After finding the optimal points, we recover these points to the original dimension by applying an upscale operator to get sensitive samples. We compare our method with all other baselines by the average detection rate with respect to the number of sensitive samples. From Figure 4.2, it can be seen that our method can generate samples with better detection ability than other baselines. This demonstrates the ability of our method to deal with complex structured data such as images.

Unknown target document retrieval

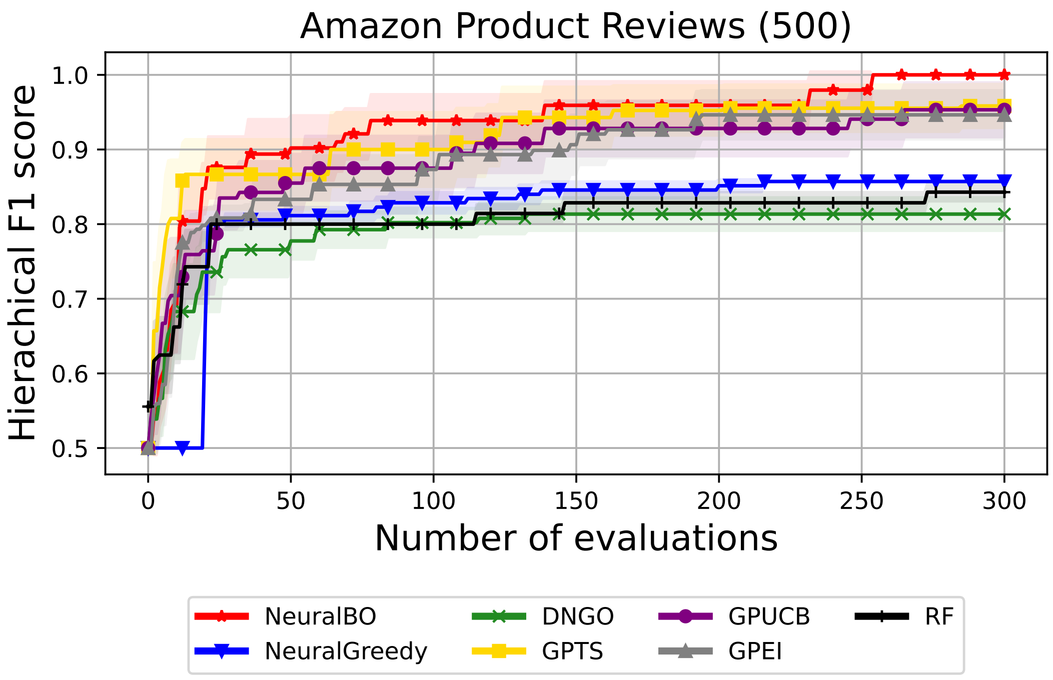


FIGURE 3.4: We search for the most related document for a specified target document in **Amazon product reviews** dataset and report the maximum **hierachical F1 score** found by all baselines. All methods show similar behaviour and Neural-BO performs comparably and much better than GP-based baselines.

Next, we evaluate our proposed method on a retrieval problem where our goal is to retrieve a document from a corpus that matches user's preference. The optimization algorithm works as follows: it retrieves a document, receives user's feedback score and then updates its belief and attempts again until it reaches a high score, or its budget is depleted. The objective target function is defined as the user's evaluation of each document, which usually has a complex structure. It is clear that evaluations are expensive since the user must read each suggestion. Searching for the most relevant document is considered as finding document d in the dataset S_{text} that maximizes the match to the target document d_t: $d = \text{argmax}_{d \in S_{text}} \text{Match}(d, d_t)$, where $\text{Match}(d, d_t)$ is a matching score between documents d and d_t. We represent each document by a word frequency vector $x_n = (x_{n1}, \cdots, x_{nJ})$, where x_{nj} is the number of occurrences of the j-th vocabulary term in the n-th document, and J is the vocabulary size.

Our experiment uses Amazon product reviews dataset [6], which are taken from users' product reviews from Amazon's online selling platform. This dataset has hierarchical categories, where the category classes are sorted from general to detail. The dataset are structured as: 6 classes in "level 1", 64 classes in "level 2" and 464 classes in "level 3". The number of users' reviews was originally 40K, which was reduced to 37738 after ignoring reviews with "unknown" category. We choose the size of vocabulary to be 500 and use the hierarchical F1-score introduced in Kiritchenko, Matwin, Famili, et al. (2005) as a scoring metric for the target and retrieved documents. We report the mean and standard deviation of hierarchical F1-score between target and retrieved documents over ten runs for all methods in Figure 3.4. Figures 3.4 indicate that our method shows a better performance for the Amazon product review dataset in comparison with other approaches.

Optimizing control parameters for robot pushing

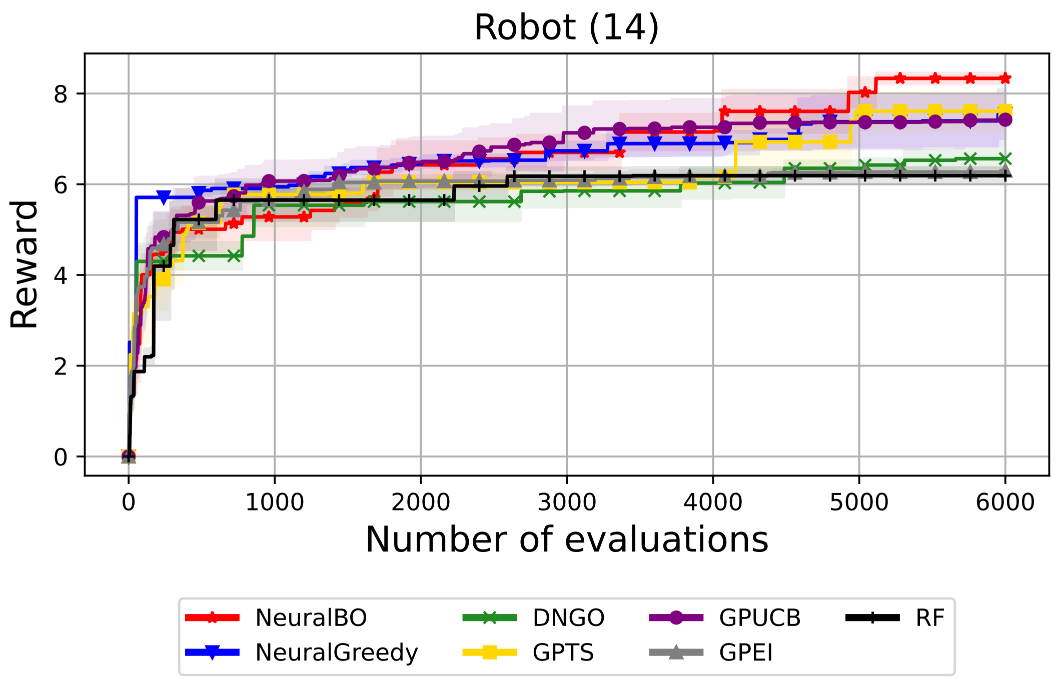


FIGURE 3.5: Optimization results for control parameters of 14D robot pushing problem. The X-axis shows iterations, and the y-axis shows the median of the best reward obtained.

Finally, we evaluate our method on tuning control parameters for the robot pushing problem considered in Wang and Jegelka (2017). We run each method for a total of

[6] https://www.kaggle.com/datasets/kashnitsky/hierarchical-text-classification

6000 evaluations and repeat ten times to take average optimization results. Neural-BO and all other methods are initialized with 15 points. Figure 3.5 summarizes the median of the best rewards achieved by all methods. It can be seen that Neural-BO achieves the highest reward after 5K optimization steps.

3.6 Conclusion

We proposed a new algorithm for Black-box Optimization using Deep Neural Networks. A key advantage of our algorithm is that it is computationally efficient and performs better than traditional Gaussian Process based methods, especially for complex structured design problems. We provided rigorous theoretical analysis for our proposed algorithm and showed that its cumulative regret converges with a sub-linear regret bound. Using both synthetic benchmark optimization functions and a few real-world optimization tasks, we showed the effectiveness of our proposed algorithm.

Chapter 4

Black-box Optimization with Unknown Black-box Constraints via Overparametrized Deep Neural Networks

Black-box Optimization with unknown constraints (CBO) methods modify the acquisition function to account for constraints, ensuring feasible solutions that satisfy these conditions while optimizing the objective function. Among these methods, Expected Improvement with Constraints (cEI), first introduced by Schonlau, Welch, and Jones (1998) and later extended by Gardner et al. (2014) and Gelbart, Snoek, and Adams (2014), has been particularly influential. By incorporating feasibility into the acquisition process, cEI directs the optimization toward regions likely to contain feasible solutions. Recent advancements, such as the quasi-Monte Carlo approximation by Letham et al. (2019), have further enhanced cEI's robustness in noisy environments.

Despite the effectiveness of EI-based methods, they encounter challenges in specific scenarios, such as when no feasible points are available or in high-dimensional search spaces. These limitations have driven the development of alternative approaches like Predictive Entropy Search with Constraints (PESC), introduced by Hernández-Lobato et al. (2015), which focuses on uncertainty reduction within feasible regions. However, computational complexities associated with quadrature sampling have restricted its practical applicability. Recent innovations, such as the Min-Value Entropy Search method proposed by Takeno et al. (2022), have simplified sampling procedures, offering a more tractable solution.

Numerical optimization has also emerged as a promising tool for addressing black-box constraints by reformulating constrained problems into simpler unconstrained ones. Techniques like Augmented Lagrangian Bayesian Optimization (ALBO) by Gramacy et al. (2016) and its variant Slack-AL by Picheny et al. (2016) use Augmented Lagrangian Functions (ALF) to iteratively solve reformulated surrogate problems. Recently, Ariafar et al. (2019) introduced ADMMBO, which applies the Alternating Direction Method of Multipliers (ADMM) technique to transform constrained problems into equivalent unconstrained optimization problems. While effective, these methods often require additional variables, increasing computational costs and limiting scalability.

To further enhance CBO, penalty functions and primal-dual methods have been employed to handle constraint violations during optimization. For instance, Lu and Paulson (2022) proposed a penalty-function approach that adds a penalty term to the objective function for violations, converting the constrained problem into an unconstrained one. Similarly, Zhou and Ji (2022) introduced a primal-dual approach that balances optimizing the objective with minimizing violations. However, these methods often rely on careful parameter tuning, making their implementation more challenging.

In parallel with empirical advancements, theoretical research has begun addressing the lack of formal guarantees in CBO. For example, Lu and Paulson (2022) proposed a penalty-based regret bound that combines regret from the objective function and penalties for constraint violations. Xu et al. (2023) further expanded this analysis by separately evaluating cumulative regret and constraint violations. In contrast, Nguyen et al. (2023) provided theoretical performance guarantees for CBO under unknown constraints in a decoupled setting, where cumulative regret is defined as the sum of both objective regret and constraint violations.

While Gaussian Processes (GPs) have been a traditional selection for modeling in CBO, their poor computational scalability remains a significant limitation. The cubic complexity of kernel matrix inversion grows prohibitive with increasing constraints. In contrast, Deep Neural Networks (DNNs) offer a scalable alternative, excelling in feature extraction and maintaining linear complexity with dataset size. Although DNNs have seen success in unconstrained black-box optimization (Snoek et al., 2015), contextual bandits in discrete search spaces (Zhou, Li, and Gu, 2020; Zhang et al., 2021), and constrained optimization with Augmented Lagrangian methods (Ariafar et al., 2019), applying DNNs to constrained black-box optimization with theoretical guarantees remains an open problem. This gap motivates the exploration of neural network-based approaches for CBO.

In Chapter 3, we inherited a similar motivation as in Chapter 3 to apply for BO with unknown, expensive constraints setting. We provided Neural-CBO, a method to alternate canonical GP by DNN as the surrogate model in BO in CBO.

4.1 Proposed Neural-CBO Method

Here, we will introduce the general DNNs architect used to model objective function and the constraints, followed by basis theory accompanying with these networks. Then, the Neural-CBO algorithm is introduced.

4.1.1 The Deep Neural Network for an Arbitrary Function f_a

Given a black-box, expensive function f_a, we use a fully connected neural network, denoted as $a(\mathbf{x};\mathbf{W})$, to model f_a:

$$a(\mathbf{x};\mathbf{W}) = \frac{\mathbf{q}^\top}{\sqrt{m}}\mathbf{D}^{(L)}(\mathbf{x})\mathbf{W}^{(L)}\dots\frac{1}{\sqrt{m}}\mathbf{D}^{(1)}(\mathbf{x})\mathbf{W}^{(1)}\mathbf{x}, \tag{4.1}$$

where $\mathbf{q} \in \mathbb{R}^m$ is the last layer weight, $\mathbf{W}^{(1)} \in \mathbb{R}^{m\times d}$, $\mathbf{W}^{(l)} \in \mathbb{R}^{m\times m}$ for $2 \le l \le L$ is the weight of the l-th hidden layer. The matrix $\mathbf{D}^{(l)}(\mathbf{x})$ is associated with the ReLU activation function and is defined as:

$$\mathbf{D}^{(l)}(\mathbf{x}) = \text{diag}\{\mathbf{1}_{\{\langle w_i^{(l)},\mathbf{h}^{(l-1)}(\mathbf{x})\rangle\ge 0\}}\} \in \mathbb{R}^{m\times m},$$

with m as the number of neurons in the hidden layer l, and $\mathbf{h}^{(l)}(\mathbf{x})$ is the output of the l-th layer given by

$$\mathbf{h}^{(l)}(\mathbf{x}) = \frac{1}{\sqrt{m}}\mathbf{D}^{(l)}(\mathbf{x})\mathbf{W}^{(l)} \dots \frac{1}{\sqrt{m}}\mathbf{D}^{(1)}(\mathbf{x})\mathbf{W}^{(1)}\mathbf{x},$$

with $\mathbf{h}^{(0)}(\mathbf{x}) = \mathbf{x}$.

At time $t = 0$, each weight matrix $\mathbf{W}^{(l)}, 2 \le l \le L$ is initialized as $\begin{bmatrix} \mathbf{\Psi} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Psi} \end{bmatrix}$, where $\mathbf{\Psi}$ is a Gaussian random matrix with independent and identically distributed (i.i.d.) standard normal entries. Additionally, the outer weights $\mathbf{q} = (\hat{\mathbf{q}}, -\hat{\mathbf{q}})^\top$ are set as random variables, and each entry of $\mathbf{b}$ is set with an equal probability of being either -1 or 1, and remain fixed throughout the training process. This initialization method is commonly employed in the literature, as seen in works like Du et al. (2018) and Arora et al. (2019), and it can be verified that, with this initialization scheme, $a(\mathbf{x}; \mathbf{W}_0) = 0$, for all input $\mathbf{x}$.

The neural network is trained by running the stochastic gradient descent on the streaming data in *one pass*. In particular, given the initialization $\{\mathbf{W}_0^{(l)}\}_{l=1}^{L}$ and last layer weight $\mathbf{q}$, the l-th layer weight matrix at the t-th iteration is updated by minimizing the L_2 loss as:

$$\mathbf{W}_{t+1}^{(l)} = \mathbf{W}_t^{(l)} + \alpha_t(y_t - a(\mathbf{x}_t; \mathbf{W}_t))\frac{\partial a(\mathbf{x}_t; \mathbf{W}_t)}{\partial \mathbf{W}^{(l)}}, \tag{4.2}$$

where α_t is the step size, and $\{\mathbf{x}_t, y_t\}$ is the observation at the t-th optimization iteration.

To estimate the uncertainty of the function f_a modeled by $a(\mathbf{x}; \mathbf{W})$, we adopt the variance formula from recent advances in neural contextual bandits research (Zhou, Li, and Gu, 2020; Kassraie and Krause, 2022):

$$\sigma_{a,t}(\mathbf{x}) = \sqrt{\mathbf{g}_a(\mathbf{x}; \mathbf{W}_0)^\top \mathbf{U}_{a,t-1}^{-1} \mathbf{g}_a(\mathbf{x}; \mathbf{W}_0)}, \tag{4.3}$$

where

$$\begin{aligned} \mathbf{g}_a(\mathbf{x}; \mathbf{W}) &= \nabla_{\mathbf{W}} a(\mathbf{x}; \mathbf{W}), \text{ and} \\ \mathbf{U}_{a,t} &= \mathbf{U}_{a,t-1} + \mathbf{g}_a(\mathbf{x}_t; \mathbf{W}_0)\mathbf{g}_a(\mathbf{x}_t; \mathbf{W}_0)^\top, \end{aligned} \tag{4.4}$$

Next, we present the common and well-established assumptions. The following assumption indicates the smoothness property of the unknown function f_a.

Assumption 4.1.1. We assume that $f_a \in \mathcal{H}_{k_a}(\mathcal{D})$, where $\mathcal{H}_{k_a}(\mathcal{D})$ is the Reproducing Kernel Hilbert Space (RKHS) associated with a real-valued function f_a defined on the domain $\mathcal{D}$. This space is induced by the Neural Tangent Kernel k_a, which arises from a neural network $a(\mathbf{x}; \mathbf{W})$. In particular, the RKHS $\mathcal{H}_{k_a}$ induces an inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}_{k_a}}$ with the reproducing property: for all $f_a \in \mathcal{H}_{k_a}(\mathcal{D})$, we have $f_a(\mathbf{x}) = \langle f_a, k_a(\cdot, \mathbf{x}) \rangle_{\mathcal{H}_{k_a}}$. The induced norm is bounded and serves as a measure of the smoothness of f_a w.r.t the kernel function k_a: $\|f_a\|_{\mathcal{H}_{k_a}} = \sqrt{\langle f_a, f_a \rangle_{\mathcal{H}_{k_a}}} \le B_a$.

To ensure that the noise arising from querying unknown function f_a remains bounded and manageable, we impose the following assumption:

Assumption 4.1.2. We assume the noises $\{\zeta_t\}_{t=1}^{T}$ where $\zeta_t = o_t - f_a(\mathbf{x}_t)$ are conditionally sub-Gaussian with parameter $R_a > 0$, where $\{\zeta_t\}_{t=1}^{T}$ is assumed to capture the noises induced by querying the black-box, expensive function $f_a(\cdot)$.

$$\forall t \geq 0,\ \forall \lambda_a \in \mathbb{R},\ \mathbb{E}[e^{\lambda_a \zeta_t} | \mathcal{F}_{a,t-1}] \leq \exp\left(\frac{\lambda_a^2 R_a^2}{2}\right),$$

where $\mathcal{F}_{a,t-1}$ are the σ-algebra generated by the random variables $\{\mathbf{x}_i, \zeta_i\}_{i=1}^{t-1} \cup \{\mathbf{x}_t\}$.

To manage the approximation error, several technical lemmas impose the following condition on the width of the neural network introduced in Eqn. 4.1.

Condition 4.1.3. *Throughout the section, the width of each hidden layer m satisfies is assumed to satisfy:*

$$m \geq d^9 \exp\Big(\Omega(\nu L C^L \log T)\Big), \tag{4.5}$$

for some absolute constant C. Besides, the step size $\alpha_t \leq \frac{\nu}{t+1}$, where ν is a parameter and independent of dimension d and width m.

Before going to our main algorithm, we provide the confidence bound, the fundamental component in almost all BO algorithms to guide algorithm design and ensure theoretical guarantee. The lemma demonstrates that by following the network width condition stated in Condition 4.1.3, the prediction of the trained neural network $a(\cdot; \mathbf{W}_{t-1})$ is concentrated around the actual value of the function $f_a(\cdot)$.

Lemma 4.1.4. *Let Assumptions 4.1.1 and 4.1.2 hold. Using neural network $a(\mathbf{x}; \mathbf{W})$ satisfied Condition 4.1.3 to model an arbitrary function f_a. Setting the step size at training step t as $\alpha_t \leq \frac{\nu}{(T+1)^2}$, then for any $\delta \in (0,1)$, with probability at least $1 - \delta \exp(\Omega(C^{-L} m^{1/36}))$, the following holds for all $\mathbf{x} \in \mathcal{D}$ and $1 \leq t \leq T$:*

$$|f_a(\mathbf{x}) - a(\mathbf{x}; \mathbf{W}_{t-1})| \leq \beta_{a,t} \sigma_{a,t-1}(\mathbf{x}) + \frac{\mathcal{E}(m)}{T+1},$$
$$\beta_{a,t} = \left(B_a + R_a \sqrt{\gamma_{t,a} + 2 + 2\log(1/\delta)}\right),$$
$$\mathcal{E}(m) = \mathcal{O}(C^{2L} L^{3/2} m^{11/36}).$$

Here, the coefficient $\beta_{a,t}$ control the uncertainty of $a(\mathbf{x}; \mathbf{W}_{t-1})$ about $f_a(\mathbf{x})$ at $\mathbf{x}$, while $\mathcal{E}(m)$ indicates the approximation error appeared when using the neural network's output $a(\mathbf{x}; W)$ to learn the underlying function f_a.

To facilitate the following algorithm design and discussion, we introduce the Lower Confidence Bound and Upper Confidence Bound functions w.r.t the *arbitrary* function f_a:

$$\mathrm{LCB}_{a,t}(\mathbf{x}, \mathbf{W}_t) = a(\mathbf{x}, \mathbf{W}_t) - \beta_{a,t} \sigma_{a,t}(\mathbf{x}) - \frac{\mathcal{E}(m)}{T+1},$$
$$\mathrm{UCB}_{a,t}(\mathbf{x}, \mathbf{W}_t) = a(\mathbf{x}, \mathbf{W}_t) + \beta_{a,t} \sigma_{a,t}(\mathbf{x}) + \frac{\mathcal{E}(m)}{T+1},$$

where $\sigma_{a,t}(\mathbf{x})$ is calculated using the formulate given in Eqn. 4.3. Then, with high probability, f_a is bounded by $\mathrm{LCB}_{a,t}(\mathbf{x}, \mathbf{W}_t)$ and $\mathrm{UCB}_{a,t}(\mathbf{x}, \mathbf{W}_t)$ as in the following corollary:

Corollary 4.1.5. *Let Assumptions 4.1.1, 4.1.2 and Condition 4.1.3 hold. Then with probability at least $1 - \delta \exp(\Omega(C^{-L} m^{1/36}))$, the following holds for all $\mathbf{x} \in \mathcal{D}$ and $1 \leq t \leq T$:*

$$f_a(\mathbf{x}) \in [\mathrm{LCB}_{a,t}(\mathbf{x}, \mathbf{W}_t), \mathrm{UCB}_{a,t}(\mathbf{x}, \mathbf{W}_t)].$$

4.1.2 Neural-CBO Algorithm

In the remaining parts of this chapter, we refer to $v(\mathbf{x};\boldsymbol{\theta})$ and $\{u_{c_i}(\mathbf{x};\boldsymbol{\omega}_{c_i})\}_{i=1}^{K}$ as the neural network models for the unknown objective function f and constraints $\{c_i\}_{i=1}^{K}$, respectively.

Our algorithm starts by initializing the neural networks $v(\mathbf{x};\boldsymbol{\theta})$ and $\{u_{c_i}(\mathbf{x};\boldsymbol{\omega}_{c_i})\}_{i=1}^{K}$ using the initialization scheme described in Section 4.1.1. We use the EI acquisition function to identify the next samples within the feasible region, determined by applying LCB-based conditions to all constraints. These conditions ensure that constraints are satisfied, while EI maintains a balance between exploration and exploitation, particularly when the feasible region is much smaller than the overall search space. (Line 4). At each optimization iteration t, the next evaluation point $\mathbf{x}_t$ is determined by maximizing the acquisition function $\mathrm{EI}_{f,t}(\mathbf{x})$ subject to the lower confidence bound constraints for all unknown black-box constraints $\{c_i(\mathbf{x})\}_{i=1}^{K}$:

$$\mathrm{LCB}_{c_i,t}(\mathbf{x},\boldsymbol{\omega}_{c_i,t}) \le 0, \forall i \in [K].$$

To handle noisy observations, we utilized the standard choice of the incumbent, which is the best value of the mean function so far: $\mu_t^+ = \max_{\mathbf{x}_k\in\mathcal{D}_{t-1}} v(\mathbf{x}_k;\boldsymbol{\theta}_{t-1})$, where the evaluations of both objective function and constraints on $\mathbf{x}_k$ yield noisy observations, such as the objective value $y_k = f(\mathbf{x}_k)+\epsilon_k$ and constraint values $\{z_{c_i,k}\}_{i=1}^{K}$, with each constraint observation given by $z_{c_i,k} = c_i(\mathbf{x}_k)+\eta_{c_i,k}$ and $\mathcal{D}_{t-1} = \{\mathbf{x}_k, y_k, z_{c_1,k},\ldots,z_{c_K,k}\}_{k=1}^{t-1}$.

Further, to control the error $\mathcal{E}(m)$ incurred by neural network model approximation, this noisy EI version can be written in closed form, following Tran-The et al. (2022) as:

$$\begin{aligned}\mathrm{EI}_{f,t}(\mathbf{x}) &= \mathbb{E}[\max\{0, v(\mathbf{x};\boldsymbol{\theta}_{t-1}) - \mu_t^+ + \mathcal{E}(m)\}]\\ &= \rho(v(\mathbf{x};\boldsymbol{\theta}_{t-1}) - \mu_t^+ + \mathcal{E}(m), \sigma_{f,t}(\mathbf{x})),\end{aligned}$$

where $\rho(u,v) = \begin{cases} u\Phi(\frac{u}{v}) + v\phi(\frac{u}{v}), & \text{if } v>0,\\ \max\{u,0\}, & \text{if } v=0.\end{cases}$

Then, we updated the dataset $\mathcal{D}_t = \mathcal{D}_{t-1}\cup\{\mathbf{x}_t, y_t, z_{c_1,t},\ldots,z_{c_K,t}\}$. The parameters $\boldsymbol{\theta}$ (for the objective function) and $\{\boldsymbol{\omega}_{c_i}\}_{i=1}^{K}$ (for the constraints) are then updated separately by minimizing the L_2 loss on the new observation using stochastic gradient descent (SGD) described in Eqn. 4.2.

Algorithm 3

Input: The input space $\mathcal{D}$, the optimization budget T, the number of constraints N

```
1: Initialize neural network models parameters $\boldsymbol{\theta}_0$, $\{\boldsymbol{\omega}_{c_i,0}\}_{i=1}^{K}$.
2: Initialize $\mathbf{U}_{f,0} = \mathbf{I}, \mathbf{U}_{c_i,0} = \mathbf{I}, \forall i \in [1\ldots K]$,
3: for $t = 1$ to $T$ do
4:     Choose $\mathbf{x}_t = \mathrm{argmin}_{\mathbf{x}\in\mathcal{D}} \mathrm{EI}_{f,t}(\mathbf{x})$ subject to $\mathrm{LCB}_{c_i,t}(\mathbf{x},\boldsymbol{\omega}_{c_i,t}) \le 0, \forall i \in [K]$
5:     Observe the noisy evaluations of objective function $y_t = f(\mathbf{x}_t)+\epsilon_t$ and constraints $\{z_{c_i,t} = c_i(\mathbf{x}_t)+\eta_{c_i,t}\}_{i=1}^{K}$.
6:     Update observations set $\mathcal{D}_t = \mathcal{D}_{t-1}\cup\{\mathbf{x}_t, y_t, z_{c_1,t},\ldots,z_{c_K,t}\}$
7:     Update the neural network parameters $\boldsymbol{\theta}_0$, $\{\boldsymbol{\omega}_{c_i,0}\}_{i=1}^{K}$ using Eqn 4.2.
8:     Update $\mathbf{U}_{f,t}$ and $\mathbf{U}_{c_i,t}, \forall i \in [1\cdots K]$ separately using Eqn. 4.4.
9: end for
```

4.2 Theoretical Analysis

To evaluate the performance of black-box optimization methods, much of the prior research on unconstrained BO has focused on minimizing cumulative regret. The cumulative regret after T iterations is defined as:

$$R_T = \sum_{t=1}^{T} r_t,$$

where $r_t = f(\mathbf{x}_t) - f(\mathbf{x}^*)$ represents the instantaneous regret, quantifying the difference between the value of the unknown function f at the optimal point, $\mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathcal{D}} f(\mathbf{x})$, and the value of the function at point $\mathbf{x}_t$, which is selected by the algorithm at iteration t. However, since $f(\mathbf{x}^*)$ represents the optimal value under constraints, the algorithm may sometimes sample infeasible points with lower objective values than $f(\mathbf{x}^*)$. To account for this, following Xu et al. (2023) and Nguyen et al. (2023), we inherited the *positive regret* definition as $r_t^+ = [f(\mathbf{x}_t) - f(\mathbf{x}^*)]^+$, where $[\cdot]^+ := \max\{0, \cdot\}$. Additionally, to measure constraint satisfaction, constraint *violation* is defined as $v_{c_i,t} = [c_i(\mathbf{x}_t)]^+$. Then, we introduce the *cumulative positive regret* for the objective function, R_T^+, and the *cumulative violation* for each constraint, $V_{c_i,T}$. These metrics measure the additional cost incurred due to sub-optimal decisions and violations of the constraints over time by running the algorithm.

Definition 4.2.1 (Cumulative Positive Regret and Cumulative Violation).

$$R_T^+ = \sum_{t=1}^{T} [f(\mathbf{x}_t) - f(\mathbf{x}^*)]^+,$$
$$V_{c_i,T} = \sum_{t=1}^{T} [c_i(\mathbf{x}_t)]^+, \forall i \in [K]$$

4.2.1 Detailed Assumptions for Objective Function and Constraints

We apply the general assumption stated in the Assumption 4.1.1 and 4.1.2 on both objective function and constraints:

- **Objective function**: $f \in \mathcal{H}_{k_f}(\mathcal{D})$, $\|f\|_{\mathcal{H}_{k_f}} \leq B$, where k_f is corresponding to $v(\cdot, \boldsymbol{\theta})$. The noisy observation at step t is $y_t = f(\mathbf{x}_t) + \epsilon_t$, where $\{\epsilon_i\}_{i=1}^t$ is sub-Gaussian with parameter R_f and variance λ_f.
- **Constraint**: $c_i \in \mathcal{H}_{k_{c_i}}(\mathcal{D}), \|c_i\|_{\mathcal{H}_{k_{c_i}}} \leq S_i$, where k_{c_i} is corresponding to $u_{c_i}(\cdot, \boldsymbol{\omega}_{c_i}), \forall i = 1, \ldots, K$. The noisy observation at step t is $z_{c_i,t} = c_i(\mathbf{x}_t) + \eta_{c_i,t}$, where $\{\eta_{c_i,t}\}_{i=1}^t$ is sub-Gaussian with parameter R_{c_i} and variance λ_{c_i}.

We can now state our main theorem:

Theorem 4.2.2. *Under Assumption 4.1.1, Assumption 4.1.2 and Condition 4.1.3, set the step size used to train the neural networks in Alg. 3 as $\alpha_t \leq \frac{\nu}{(T+1)^2}$, then for any $\delta \in (0,1)$, with probability at least $1 - \delta \exp(\Omega(C^{-L} m^{1/36}))$, the Cumulative Regret R_T and Cumulative*

Violation $V_{c_i,T}$ after T iterations are bounded as:

$$V_{c_i,T} \leq 2\beta_{c_i,T}\sqrt{\frac{S_i T}{\log(S_i+1)}(2\gamma_{c_i,T}+1)} + 2\mathcal{E}(m),$$

$$R_T \leq R_T^+ \leq 2\beta_{f,T}\sqrt{\frac{BT}{\log(B+1)}(2\gamma_{f,T}+1)} + 2\mathcal{E}(m),$$

where $\mathcal{E}(m) = \mathcal{O}(C^{2L}L^{3/2}m^{11/36})$. Especially, by choosing $m = \Omega(d^9 \exp(\nu LC^L \log T))$, the Cumulative Regret and Cumulative Violation enjoy the following results:

$$V_{c_i,T} = \mathcal{O}(\gamma_{c_i,T}\sqrt{T}), \quad R_T = \mathcal{O}(\gamma_{f,T}\sqrt{T}).$$

Remark 4.2.3. Unlike previous works (Zhou, Li, and Gu, 2020; Zhang et al., 2021) that require a neural network width of $m = \Omega(T^6)$ for convergence when modeling the objective function, our analysis builds on recent analyses from Xu and Zhu (2024), which show that only a linear condition of $m = \Omega(T)$ is needed. Furthermore, while Xu and Zhu (2024) focus on the input domain $\mathbb{S}^{d-1}$, we can adapt to inputs $\mathbf{x} \in \mathbb{R}^d$ with $0 < n_l < \|\mathbf{x}\| < n_b$ (where n_l and n_b are positive constants) without changing the order of T in the width condition for m. Similar arguments are noted in (Du et al., 2018; Cao and Gu, 2020).

4.3 Experiments

In this section, we outline and discuss our experimental results. We apply our method to synthetic, real-world functions.

4.3.1 Baselines

For all experiments, we compared our algorithm with well-known Constrained EI (cEI), the extension of EI into constrained BO from Gardner et al. (2014). Besides, we also compare our algorithm with recent state-of-the-art algorithms in unknown constrained BO, including ADMMBO (Ariafar et al., 2019), UCB-C (Nguyen et al., 2023) and ConfigOpt (Xu et al., 2023). For our proposed Neural-CBO algorithm, we employ fully connected deep neural networks as the surrogate models for both objective function and constraints. Implementation details of our Neural-CBO algorithm as well as baselines are provided in the Supplementary Material.

4.3.2 Synthetic Benchmark Functions

We conducted optimization experiments on four synthetic objective functions: Branin, Ackley, Simionescu and Hartmann. The input dimension of each objective function and the corresponding number of constraints are summarized in Table 4.1. Due to space limitations, we present the expression of each function and its constraints in the Supplementary material. The noise in function evaluations follows a normal distribution with zero mean, and the variance is set to 1% of the function range. All experiments reported here are averaged over 20 runs, each with random initialization. We report the (Log10 of) the Best Positive Regret plus Violation in Figure 4.1

TABLE 4.1: The input dimension and number of constraints for each synthetic objective function.

Obj	Branin	Simionescu	Ackley	Hartmann
Dim	2	2	5	6
Constraints	1	1	2	1

To ensure statistical significance, we performed one-sided t-tests to assess whether a baseline outperforms Neural-CBO in terms of the best positive regret plus violation. The null hypothesis is $H_0 : \mu_{\text{baseline}} \leq \mu_{\text{Neural-CBO}}$, and the alternative hypothesis is $H_a : \mu_{\text{baseline}} > \mu_{\text{Neural-CBO}}$, where μ_{baseline} and $\mu_{\text{Neural-CBO}}$ represent the means of the (Log10 of) Best Positive Regret plus Violation values of the baseline and our proposed Neural-CBO, respectively. Note that lower values indicate better performance. We present the statistical test results for four synthetic benchmark functions and two real-world tasks (described in Section 4.3.3 and 4.3.4) in Table 4.2. Each cell in the table shows the p-value from the t-test as the first value. To account for multiple comparisons, the Benjamini-Hochberg correction was applied, with the corrected value provided as the second value. A result is labeled as "T" if the null hypothesis is rejected, meaning that Neural-CBO is statistically better to the compared baselines. Conversely, a result is labeled "F" if we cannot reject the null hypothesis, meaning that the baselines and the Neural-CBO are comparable.

TABLE 4.2: One-sided t-tests to evaluate whether the baseline outperforms Neural-CBO in terms of best positive regret plus violation.

	ConfigOpt	**cEI**	**UCB-C**	**ADMMBO**
Branin	(8.25e-06, T)	(6.41e-20, T)	(4.68e-73, T)	(6.86e-102, T)
Simionescu	(6.19e-07, T)	(2.42e-13, T)	(2.13e-25, T)	(1.32e-78, T)
Ackley	(6.95e-29, T)	(0.13, F)	(6.4e-97, T)	(0.21, F)
Hartmann	(0.00594, T)	(1.54e-59, T)	(4.88e-118, T)	(1.2e-87, T)
Gas Transmission	(1.52e-46, T)	(1.78e-48, T)	(1.66e-25, T)	(1.54e-73, T)
Speed Reducer	(0.77, F)	(6.99e-74, T)	(5.22e-77, T)	(0.38, F)

We analyze three real-world constrained black-box optimization tasks: gas transmission compressor and speed reducer designs from Kumar et al. (2020), and a third inspired by He, Zhang, and Lee (2018). Details of each task will follow in the upcoming sections.

4.3.3 Gas Transmission Compressor Design

The main objective is to minimize operational costs or energy consumption. This requires identifying the optimal configuration of the compressor by optimizing four design variables. The problem involves $d = 4$ input dimensions and includes $K = 1$ constraint. The detailed mathematic formula is provided in Supplementary Material.

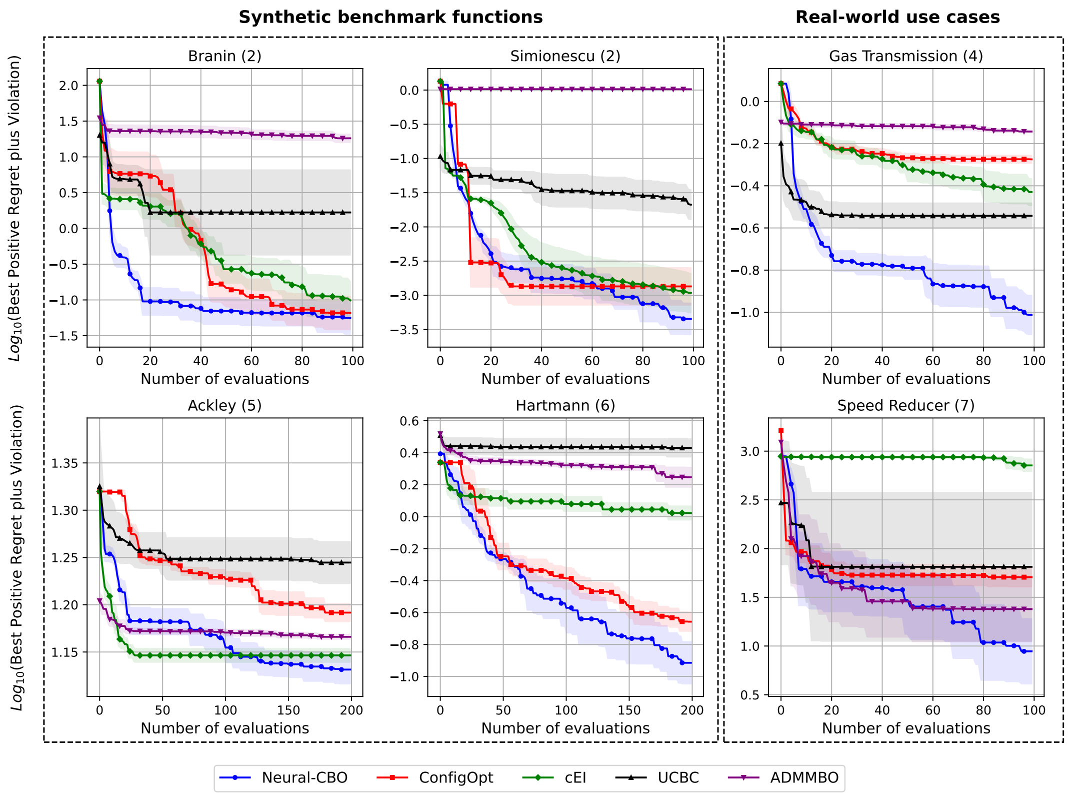


FIGURE 4.1: The plots show (Log10 of) the Best Positive Regret plus Violation up to step t, which is $\min_{t\in[T]}[f(\mathbf{x}_t) - f^*]^+ + \sum_{k=1}^{K}[c_k(\mathbf{x}_t)]^+]$, comparing our proposed algorithm and four baselines. The dimension of each objective function is shown in the parenthesis. The left group is four synthetic functions introduced in Section 4.3.2, while the right group is the optimization results of Gas Transmission Compressor Design and Speed Reducer Design, described in Section 4.3.3 and 4.3.4.

4.3.4 Speed Reducer Design

This task involves designing a speed reducer for a small aircraft engine, focusing on minimizing weight while meeting several constraints, including bending stress on gear teeth, surface stress, transverse deflections of shafts, and shaft stresses. The problem includes 7 decision variables and 11 constraints, resulting in an input dimension of $d = 7$ and $K = 11$ constraints. The detailed mathematical formulation is provided in the Supplementary Material. We report numerical results of Section 4.3.3 and 4.3.4 in Figure 4.1 and Table 4.2.

4.3.5 Designing Sensitive Samples for Detection of Model Tampering

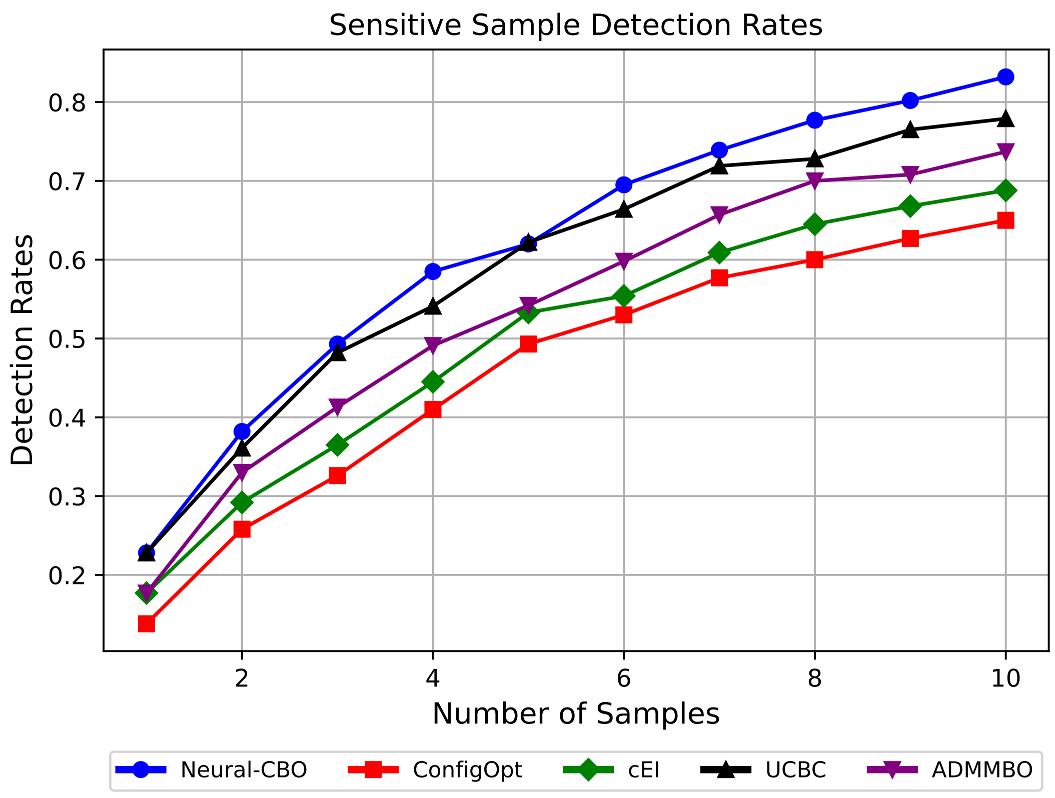


FIGURE 4.2: **Detection Rates** corresponding to the number of samples on the MNIST dataset. As shown in the figure, Neural-CBO can generate sensitive samples that achieve nearly 85% of the detection rate with at least 10 samples.

We examine a scenario where a company offers Machine Learning as a Service (MLaaS) and hosts its model in the cloud. In this context, an attacker with system access could alter the model by changing its weights. To detect such tampering, He, Zhang, and Lee (2018) propose generating a set of test inputs, called *Sensitive Samples* $\{v_i\}_{i=1}^n$, whose outputs from the modified model will differ from those of the original. Assuming a pre-trained model $s_\varphi(\mathbf{x})$ may have been altered after being uploaded, the goal is to find sensitive samples by solving the optimization problem:

$$v = \underset{\mathbf{x}}{\operatorname{argmax}} \left\| \frac{\partial s_\varphi(\mathbf{x})}{\partial \varphi} \right\|_F,$$

where $\|\cdot\|_F$ denotes the Frobenius norm. A detection is *successful* if at least one of the N_S sensitive samples shows a different top-1 prediction between the tampered and original models. To prevent attackers from evading detection, sensitive samples must resemble normal inputs. Therefore, a human-in-the-loop process is employed, where reviewers rate the realism of each sample on a scale of $(0, 1)$; higher scores indicate more realistic samples. These scores serve as constraints in the optimization, where obtaining human feedback can be costly.

We utilized a pre-trained MNIST hand-written digit classification model and compared our method's performance against several baselines based on average detection rates for sensitive samples. Feasible samples were chosen based on their realistic scores. Figure 4.2 shows the detection rates of (feasible) sensitive samples generated by our method compared to four baselines, demonstrating that our samples achieved higher detection rates. As expected, the detection rate improves with more samples and our method is consistently competitive.

4.4 Conclusion

We proposed a novel algorithm for black-box optimization with unknown constraints, utilizing deep neural networks as surrogate models for both the objective function and constraints. Our algorithm leverages the bounded nature of constraint values by applying LCB conditions at each iteration to ensure feasibility. We also employ EI as the acquisition function to balance exploration and exploitation, especially in scenarios where feasible regions are significantly smaller than the search space. Our theoretical analysis shows that, under mild conditions regarding neural network width, our algorithm achieves upper bounds on cumulative regret and constraint violations comparable to previous GPs-based methods. We validate our approach through experiments on synthetic and real-world benchmark tasks involving structural data, with results demonstrating competitive performance against state-of-the-art methods.

Chapter 5

PINN-BO: A Black-Box Optimization Algorithm Using Physics-Informed Neural Networks

In Chapter 3 and Chapter 4, we explored DNN-based BO and its extension to settings with unknown constraints. In practical applications, these constraints often appear from physical principles and can be represented by PDEs. This chapter introduces a novel BO framework that incorporates physical knowledge directly into the optimization process, enhancing model fidelity and constraint handling in scenarios governed by known physics.

5.1 Problem Setting

In this chapter, we consider a global optimization problem setting where the objective function $f\colon \mathcal{D} \to \mathbb{R}$ is associated with a PDE:

$$\min_{\mathbf{x}\in\mathcal{D}} f(\mathbf{x}) \text{ s.t. } \mathcal{N}[f](\mathbf{x}) = g(\mathbf{x}),$$

where $\mathcal{D} \subset \mathbb{R}^d$ is a d-dimensional bounded domain and $\mathcal{N}[f]$ denotes a differential operator of the function f with respect to the input $\mathbf{x}$. The function f is an expensive, black-box function, and its evaluations are obtainable only through noisy measurements in the form of $y = f(\mathbf{x}) + \epsilon$, where ϵ represents sub-Gaussian noise.

5.2 Proposed PINN-BO Method

In this section, we present our proposed Physics-Informed Neural Network based Black-box Optimization (PINN-BO). PINN-BO algorithm combines optimization and machine learning techniques to efficiently optimize an unknown black-box function over a given input space while leveraging physics-informed constraints described by a PDE. Following the fundamental principles of Bayesian optimization, our algorithm consists of two primary steps: (1) Constructing a model of the black-box objective function, and (2) Employing this model to select the next function evaluation point during each iteration. In the first step, our approach diverges from traditional Bayesian Optimization algorithms that typically utilize Gaussian processes to model the objective function. Instead, we employ a fully connected neural network denoted as $h(\mathbf{x};\boldsymbol{\theta})$ to learn the function f as follows:

$$h(\mathbf{x};\boldsymbol{\theta}) = \frac{1}{\sqrt{m}}\mathbf{W}_L\psi(\mathbf{W}_{L-1}\psi(\cdots\psi(\mathbf{W}_1\mathbf{x})),$$

where $\psi: \mathbb{R} \to \mathbb{R}$ is a coordinate-wise smooth activation function (e.g., ReLU, Tanh), $\mathbf{W}_1 \in \mathbb{R}^{m\times d}, \mathbf{W}_i \in \mathbb{R}^{m\times m}, 2 \le i \le L-1, \mathbf{W}_L \in \mathbb{R}^{1\times m}$, and $\boldsymbol{\theta} \in \mathbb{R}^p$ is the collection of parameters of the neural network, $p = md + m^2(L-2) + m$ and d is the dimension of inputs, i.e., $\mathbf{x} \in \mathcal{D} \subset \mathbb{R}^d$. We initialize all the weights to be independent and identically distributed as standard normal distribution $\mathcal{N}(0,1)$ random variables. To leverage the information embedded within the PDE governing the objective function f, our algorithm generates a set of N_r PDE data points denoted as $\mathcal{R} = \{\mathbf{z}_j, u_j\}_{j=1}^{N_r}$. Here, u_j represents the noisy evaluations of the function g at the corresponding point $\mathbf{z}_j$, where $u_j = g(\mathbf{z}_j) + \eta_j$. Besides, we denote $\mathcal{D}_t = \{\mathbf{x}_i, y_i\}_{i=1}^{t}$ as the set of noisy observations of the unknown function f after t optimization iterations, where $y_t = f(\mathbf{x}_t) + \epsilon_t$. We further define some other notations:

$$\phi(\cdot) = \nabla_{\boldsymbol{\theta}} h(\cdot;\boldsymbol{\theta}_0); \quad \omega(\cdot) = \nabla_{\boldsymbol{\theta}} \mathcal{N}[h](\cdot;\boldsymbol{\theta}_0)$$

where $\phi(\cdot)$ is the gradient of $h(\cdot;\boldsymbol{\theta}_0)$ with respect to the parameter $\boldsymbol{\theta}$, evaluated at initialization $\boldsymbol{\theta}_0$. Similarly, $\omega(\cdot)$ represents the gradient of $\mathcal{N}[h](\cdot;\boldsymbol{\theta}_0)$ with respect to model parameters $\boldsymbol{\theta}$ and evaluated at initialization $\boldsymbol{\theta}_0$, where $\mathcal{N}[h](\cdot;\boldsymbol{\theta}_0)$ is the result of applying differential operator $\mathcal{N}$ (with respect to the input) to $h(\cdot;\boldsymbol{\theta}_0)$. Both $\mathcal{D}_t$ and $\mathcal{R}$ play an important role in the subsequent stages of the algorithm, specifically in the minimization of the loss function associated with learning the network $h(\mathbf{x},\boldsymbol{\theta}_t)$ at optimization iteration t:

$$\mathcal{L}(t) = \textstyle\sum_{i=1}^{t-1}[y_i - \nu_t h(\mathbf{x}_i;\boldsymbol{\theta}_{t-1})]^2 + \sum_{j=1}^{N_r}[u_j - \nu_t \mathcal{N}[h](\mathbf{z}_j;\boldsymbol{\theta}_{t-1})]^2, \tag{5.1}$$

where ν_t is a scale parameter that controls the exploration-exploitation trade-off.

For the second step, we employ a greedy strategy to pick the next sample point $\mathbf{x}_t$. At each iteration t, the algorithm updates the neural network by optimizing the loss function described in Eqn 5.1 by gradient descent, with scaled function value predictions $\nu_t h(\cdot;\boldsymbol{\theta}_{t-1})$ and scaled predictions with respect to the governed PDE $\nu_t\mathcal{N}[h](\cdot;\boldsymbol{\theta}_{t-1})$. In the proof of Section 5.3, we show this action is equivalent to placing the GP prior over function values $f_{1:t}$ and PDE values $g_{1:N_r}$ (Corollary C.1.1.1 in the Appendix C). Then the posterior distribution of the function prediction $\widetilde{f}_t(\mathbf{x}) = h(\mathbf{x},\boldsymbol{\theta}_{t-1})$ at a new data point $\mathbf{x}$ can be viewed as being sampled from a GP with specific posterior mean and variance function (Lemma 5.3.4). This allows us to directly use the network prediction as an acquisition function following the principle of Thompson Sampling. Then, the next evaluation point $\mathbf{x}_t$ is selected by minimizing this acquisition function $\widetilde{f}_t(\mathbf{x}) = h(\mathbf{x},\boldsymbol{\theta}_{t-1})$. Then, the black-box function is queried at point $\mathbf{x}_t$, resulting in a (noisy) observation y_t, which is subsequently used to update the dataset $\mathcal{D}_t$. The PDE observations set $\mathcal{R}$, in combination with the observations in $\mathcal{D}_t$, is integrated into the training process of the neural network by minimizing the squared loss, as described in Eqn 5.1. We provide a concise step-by-step summary of our approach in Algorithm 4.

To enhance the exploration step, it is important to bring the additional information to improve our model of objective function, especially in the regions where optima lies. In our case, this task of exploration is easier as we have access to PDE which provides knowledge about the objective function and reduces the amount of information that is needed to model the function. In Section 5.3 (Theoretical Analysis), we derive a scaling factor $\nu_t = B + \widetilde{R}\sqrt{2\gamma_t - 2I(f;\mathbf{Y}_t;\mathbf{U}_r) + \log(\frac{1}{\delta})}$, which directly reflects this intuition. It reduces the maximum information gain (which can be thought of as the complexity of the function modeling) by the interaction information $I(f;\mathbf{Y}_t;\mathbf{U}_r)$, which is a generalization of the mutual information for three variables: unknown function f, its observations $\mathbf{Y}_t$, and the PDE data $\mathbf{U}_r$. This information can be calculated as $I(f;\mathbf{Y}_t;\mathbf{U}_r) = \frac{1}{2}\log\left(\frac{\det\left(\frac{\mathbf{\Phi}_t^\top\mathbf{\Phi}_t}{\lambda_1}+\mathbf{I}\right)\det\left(\frac{\mathbf{\Omega}_r^\top\mathbf{\Omega}_r}{\lambda_2}+\mathbf{I}\right)}{\det\left(\frac{\mathbf{\Phi}_t^\top\mathbf{\Phi}_t}{\lambda_1}+\frac{\mathbf{\Omega}_r^\top\mathbf{\Omega}_r}{\lambda_2}+\mathbf{I}\right)}\right)$, where $\mathbf{\Phi}_t = [\phi(\mathbf{x}_1)^\top,\dots,\phi(\mathbf{x}_t)^\top]^\top$ and $\mathbf{\Omega}_r = [\omega(\mathbf{z}_1)^\top,\dots,\omega(\mathbf{z}_{N_r})^\top]^\top$. The value of ν_t signifies how our algorithm continues the exploration in regions of search space where the function f has no implicit knowledge through PDE observations. These are the regions indicated by $\gamma_t - I(f;\mathbf{Y}_t;\mathbf{U}_r)$, which is the amount of information about the unknown function f remains after our algorithm interacts with PDE data $\mathbf{U}_r$.

Algorithm 4 Physics-informed Neural Network based Black-box optimization (PINN-BO)

Input: The input space $\mathcal{D}$, the optimization budget T, PDE training set size N_r, $\delta \in (0,1)$, parameters $B, R_1, R_2, \lambda_1, \lambda_2$ (see Assumption 5.3.2 and 5.3.3 in Section 5.3).

```
1: Initialize $\mathcal{D}_0 = \emptyset$ and $\boldsymbol{\theta}_0 \sim \mathcal{N}(\mathbf{0},\mathbf{I})$
2: Generate set $\mathcal{R} = \{\mathbf{z}_j, u_j\}_{j=1}^{N_r}$ from the PDE.
3: for $t = 1$ to $T$ do
4:     Set $\nu_t = B + \widetilde{R}\sqrt{2\gamma_t - 2I(f;\mathbf{Y}_t;\mathbf{U}_r) + \log(\frac{1}{\delta})}$, where $\widetilde{R} = \sqrt{\left(\frac{R_1}{\lambda_1}\right)^2 + \left(\frac{R_2}{\lambda_2}\right)^2}$.
5:     $\widetilde{f}_t(\mathbf{x}) = h(\mathbf{x};\boldsymbol{\theta}_{t-1})$
6:     Choose $\mathbf{x}_t = \text{argmin}_{\mathbf{x}\in\mathcal{D}} \widetilde{f}_t(\mathbf{x})$ and receive observation $y_t = f(\mathbf{x}_t) + \epsilon_t$
7:     Update $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{\mathbf{x}_t, y_t\}$
8:     Update $\boldsymbol{\theta}_t = \text{argmin}_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$ using Eqn. 5.1 by gradient descent with $\nu = \nu_t$.
9: end for
```

5.3 Theoretical Analysis

In this section, we provide a regret bound for the proposed PINN-BO algorithm. To quantify the algorithm's regret, we employ the cumulative regret, defined as $R_T = \sum_{t=1}^{T} r_t$ after T iterations. Here, $\mathbf{x}^* = \text{argmin}_{\mathbf{x}\in\mathcal{D}} f(\mathbf{x})$ represents the optimal point of the unknown function f, and $r_t = f(\mathbf{x}^*) - f(\mathbf{x}_t)$ denotes the instantaneous regret incurred at time t. Our regret analysis is built upon the recent NTK-based theoretical work of Wang, Yu, and Perdikaris (2022) and proof techniques of GP-TS Chowdhury and Gopalan (2017). Before proceeding with the theoretical analysis, we now introduce a set of definitions and assumptions. They clarify our proof and set up the basis and conditions for our analysis. **A detailed proof can be found in Section 5.3 of the Appendix.**

Definition 5.3.1. We define matrix $\mathbf{K}_{\text{NTK}-\text{PINN}}$ as the *neural tangent kernel of a Physics-Informed Neural Network (NTK of PINNs)*:

$$\mathbf{K}_{\text{NTK}-\text{PINN}} = \begin{bmatrix} \mathbf{K}_{uu} & \mathbf{K}_{ur} \\ \mathbf{K}_{ru} & \mathbf{K}_{rr} \end{bmatrix}, \tag{5.2}$$

where $(\mathbf{K}_{uu})_{ij} = \langle\phi(\mathbf{x}_i),\phi(\mathbf{x}_j)\rangle$, $(\mathbf{K}_{ur})_{ij} = \langle\phi(\mathbf{x}_i),\omega(\mathbf{z}_j)\rangle$, $(\mathbf{K}_{rr})_{ij} = \langle\omega(\mathbf{z}_i),\omega(\mathbf{z}_j)\rangle$ and $\mathbf{K}_{ru} = \mathbf{K}_{ur}^\top$, defined using $\mathcal{D}_t$ and $\mathcal{R}$.

Assumption 5.3.2. We assume the noises $\{\epsilon_i\}_{i=1}^{T}$ where $\epsilon_i = y_i - f(\mathbf{x}_i)$ and $\{\eta_j\}_{j=1}^{N_r}$ where $\eta_j = u_j - g(\mathbf{z}_j)$ are conditionally sub-Gaussian with parameter $R_1 > 0$ and $R_2 > 0$, where $\{\epsilon_i\}_{i=1}^{T}$ and $\{\eta_j\}_{j=1}^{N_r}$ is assumed to capture the noises induced by querying the black-box, expensive function $f(\cdot)$ and cheap-to-evaluate PDE-related function $g(\cdot)$ respectively.

$$\forall i \geq 0, \forall \lambda_1 \in \mathbb{R},\ \mathbb{E}[e^{\lambda_1 \epsilon_i} | \mathcal{F}_{t-1}] \leq e^{\frac{\lambda_1^2 R_1^2}{2}}$$
$$\forall j \geq 0, \forall \lambda_2 \in \mathbb{R},\ \mathbb{E}[e^{\lambda_2 \eta_j} | \mathcal{F}'_{N_r-1}] \leq e^{\frac{\lambda_2^2 R_2^2}{2}}$$

where $\mathcal{F}_{t-1}, \mathcal{F}'_{N_r-1}$ are the σ-algebra generated by the random variables $\{\mathbf{x}_i, \epsilon_i\}_{i=1}^{t-1} \cup \{\mathbf{x}_t\}$ and $\{\mathbf{z}_j, \eta_j\}_{j=1}^{N_r-1} \cup \{\mathbf{z}_{N_r}\}$, respectively.

Assumption 5.3.3. We assume f to be an element of the Reproducing Kernel Hilbert Space (RKHS) associated with real-valued functions defined on the set $\mathcal{D}$. This specific RKHS corresponds to the Neural Tangent Kernel of a physics-informed neural network (NTK-PINN) and possesses a bounded norm denoted as $\|f\|_{\mathcal{H}_{k_{\text{NTK-PINN}}}} \leq B$. Formally, this RKHS is denoted as $\mathcal{H}_{k_{\text{NTK-PINN}}}(\mathcal{D})$, and is uniquely characterized by its kernel function $k_{\text{NTK-PINN}}(\cdot,\cdot)$. The RKHS induces an inner product $\langle \cdot, \cdot \rangle$ that obeys the reproducing property: $f(\mathbf{x}) = \langle f, k_{\text{NTK-PINN}}(\cdot, \mathbf{x}) \rangle$ for all $f \in \mathcal{H}_{k_{\text{NTK-PINN}}}(\mathcal{D})$. The norm induced within this RKHS, $\|f\|_{\mathcal{H}_{k_{\text{NTK-PINN}}}} = \sqrt{\langle f, f \rangle_{\mathcal{H}_{k_{\text{NTK-PINN}}}}}$, quantifies the smoothness of f concerning the kernel function $k_{\text{NTK-PINN}}$, and satisfies: $f \in \mathcal{H}_{k_{\text{NTK-PINN}}}(\mathcal{D})$ if and only if $\|f\|_{k_{\mathcal{H}_{\text{NTK-PINN}}}} < \infty$.

Assumptions 5.3.2 and 5.3.3 represent commonly employed and well-established assumptions in GP-based Bandits and Bayesian Optimization (Chowdhury and Gopalan, 2017; Vakili et al., 2021). We are now prepared to establish an upper bound on the regret incurred by our proposed PINN-BO algorithm.

We begin by presenting key lemmas for establishing the regret bound in Theorem 5.3.11 of the proposed algorithms. The following lemma demonstrates that, given the assumption of an infinitely wide network, the output of the trained physics-informed neural network used to model the unknown function f governed by a linear PDE, after running t optimization iterations in Algorithm 4, can be regarded as sampling from a GP with specific mean and covariance functions.

Lemma 5.3.4. *Conditioned on* $\mathcal{D}_t = \{\mathbf{x}_i, y_i\}_{i=1}^{t}, \mathcal{R} = \{\mathbf{z}_j, u_j\}_{j=1}^{N_r}$, *the acquisition function* $\widetilde{f}_t(\mathbf{x}) = h(\mathbf{x}; \boldsymbol{\theta}_{t-1})$ *can be viewed as a random draw from a* GP $\left(\mu_t^f(\mathbf{x}), \nu_t^2 \left(\sigma_t^f\right)^2(\mathbf{x})\right)$ *with the following mean and covariance functions:*

$$\mu_t^f(\mathbf{x}) = \phi(\mathbf{x})^\top \boldsymbol{\xi}_t^\top \widehat{\mathbf{K}}_{\text{PINN}}^{-1} \begin{bmatrix} \mathbf{Y}_t \\ \mathbf{U}_r \end{bmatrix}$$
$$\left(\sigma_t^f\right)^2(\mathbf{x}) = \langle \phi(\mathbf{x}), \phi(\mathbf{x}) \rangle - \phi(\mathbf{x})^\top \boldsymbol{\xi}_t^\top \widehat{\mathbf{K}}_{\text{PINN}}^{-1} \boldsymbol{\xi}_t \phi(\mathbf{x}),$$

where

$$\begin{aligned}
\widehat{\mathbf{K}}_{\text{PINN}}^{-1} &= \begin{bmatrix} \mathbf{K}_{uu} + \lambda_1 \mathbf{I} & \mathbf{K}_{ur} \\ \mathbf{K}_{ru} & \mathbf{K}_{rr} + \lambda_2 \mathbf{I} \end{bmatrix}^{-1} = \begin{bmatrix} \widetilde{\mathbf{A}} & \widetilde{\mathbf{B}} \\ \widetilde{\mathbf{C}} & \widetilde{\mathbf{D}} \end{bmatrix} \\
\mathbf{\Phi}_t &= [\phi(\mathbf{x}_1)^\top, \dots, \phi(\mathbf{x}_t)^\top]^\top \\
\mathbf{\Omega}_r &= [\omega(\mathbf{z}_1)^\top, \dots, \omega(\mathbf{z}_{N_r})^\top]^\top,\ \xi_t = \begin{bmatrix} \mathbf{\Phi}_t^\top & \mathbf{\Omega}_r^\top \end{bmatrix}^\top \\
\mathbf{K}_{uu} &= \mathbf{\Phi}_t \mathbf{\Phi}_t^\top,\ \mathbf{K}_{ur} = \mathbf{\Phi}_t \mathbf{\Omega}_r^\top, \mathbf{K}_{ru} = \mathbf{K}_{ur}^\top, \mathbf{K}_{rr} = \mathbf{\Omega}_r \mathbf{\Omega}_r^\top \\
\mathbf{Y}_t &= [y_1, y_2, \dots, y_t]^\top, \mathbf{U}_r = [u_1, u_2, \dots, u_{N_r}]^\top
\end{aligned}$$

Generic BO methods (without the extra information from PDE) utilized the *maximum information gain* over search space $\mathcal{D}$ at time t: $\gamma_t := \max_{\mathcal{A} \subset \mathcal{D}: |\mathcal{A}| = t} I(\mathbf{Y}_\mathcal{A}, f_\mathcal{A})$, where $I(\mathbf{Y}_\mathcal{A}, f_\mathcal{A})$ denotes the mutual information between $f_\mathcal{A} = [f(\mathbf{x})]_{\mathbf{x} \in \mathcal{A}}$ and noisy observations $\mathbf{Y}_\mathcal{A}$, which quantifies the reduction in uncertainty about the objective function f after observing y_A. The maximum information gain is the fundamental component when analyzing regret bound for their algorithm (Srinivas et al., 2009; Vakili et al., 2021). However, in our work, the PDE evaluations $\{u_j\}_{j=1}^{N_r}$ of the function g is considered as the second source of information that contributes to reduce the uncertainty of f. Therefore, we introduce the *interaction information* as the generalization of *mutual information* for three random variables:

Definition 5.3.5. The **interaction information** between f, its observations $\mathbf{Y}_\mathcal{A}$, (where $\mathcal{A} \subset \mathcal{D}$), and the PDE data $\mathbf{U}_r$ can be defined as:

$$I(f; \mathbf{Y}_\mathcal{A}; \mathbf{U}_r) = I(f; \mathbf{Y}_\mathcal{A}) - I(f; \mathbf{Y}_\mathcal{A} | \mathbf{U}_r),$$

where $I(f; \mathbf{Y}_\mathcal{A})$ quantifies the reduction in uncertainty in f due to observing $\mathbf{Y}_\mathcal{A}$, while $I(f; \mathbf{Y}_\mathcal{A} | \mathbf{U}_r)$ represents the additional information contributed by $\mathbf{U}_r$ to enhance the mutual information between f and $\mathbf{Y}_\mathcal{A}$.

The next lemma provides the closed-form expression of the interaction information.

Lemma 5.3.6. *The interaction information between f and observation $\mathbf{Y}_t$ and PDE data $\mathbf{U}_r$, for the points chosen from Algorithm 4 can be calculated as:*

$$I(f; \mathbf{Y}_t; \mathbf{U}_r) = \frac{1}{2} \log \left(\frac{\det\left(\frac{\mathbf{\Phi}_t^\top \mathbf{\Phi}_t}{\lambda_1} + \mathbf{I} \right) \det\left(\frac{\mathbf{\Omega}_r^\top \mathbf{\Omega}_r}{\lambda_2} + \mathbf{I} \right)}{\det\left(\frac{\mathbf{\Phi}_t^\top \mathbf{\Phi}_t}{\lambda_1} + \frac{\mathbf{\Omega}_r^\top \mathbf{\Omega}_r}{\lambda_2} + \mathbf{I} \right)} \right)$$

Remark 5.3.7. Following Remark 3.3 in Wang, Yu, and Perdikaris (2022), both matrices $\frac{\mathbf{\Phi}_t^\top \mathbf{\Phi}_t}{\lambda_1}$ and $\frac{\mathbf{\Omega}_r^\top \mathbf{\Omega}_r}{\lambda_2}$ are positive semi-definite. It can be clearly seen that the interaction information given in Lemma 5.3.6 is non-negative:

$$\begin{aligned}
I(f; \mathbf{Y}_t; \mathbf{U}_r) &= \frac{1}{2} \log \left(\frac{\det\left(\frac{\mathbf{\Phi}_t^\top \mathbf{\Phi}_t}{\lambda_1} + \mathbf{I} \right) \det\left(\frac{\mathbf{\Omega}_r^\top \mathbf{\Omega}_r}{\lambda_2} + \mathbf{I} \right)}{\det\left(\frac{\mathbf{\Phi}_t^\top \mathbf{\Phi}_t}{\lambda_1} + \frac{\mathbf{\Omega}_r^\top \mathbf{\Omega}_r}{\lambda_2} + \mathbf{I} \right)} \right) \\
&= \frac{1}{2} \log \left(\frac{\det\left(\frac{\mathbf{\Phi}_t^\top \mathbf{\Phi}_t}{\lambda_1} + \frac{\mathbf{\Omega}_r^\top \mathbf{\Omega}_r}{\lambda_2} + \mathbf{I} + \frac{\mathbf{\Phi}_t^\top \mathbf{\Phi}_t \mathbf{\Omega}_r^\top \mathbf{\Omega}_r}{\lambda_1 \lambda_2} \right)}{\det\left(\frac{\mathbf{\Phi}_t^\top \mathbf{\Phi}_t}{\lambda_1} + \frac{\mathbf{\Omega}_r^\top \mathbf{\Omega}_r}{\lambda_2} + \mathbf{I} \right)} \right) \geq 0
\end{aligned}$$

The inequality uses the identity $\det(\mathbf{A}+\mathbf{B}) \geq \det(\mathbf{A})$, where $\mathbf{A}, \mathbf{B}$ are two positive semi-definite matrices.

Our next result shows how the prediction of the neural network model is concentrated around the unknown reward function f, which is the key to a tighter regret bound.

Lemma 5.3.8. *Assume that* $\|\omega(\cdot)\|_2 \leq L$, *where* $\omega(\cdot) = \nabla_{\boldsymbol{\theta}}\mathcal{N}[h](\cdot;\boldsymbol{\theta}_0)$ *and* $\rho_{min}(\mathbf{K}_{uu})$ *the smallest eigenvalue of kernel matrix* $\mathbf{K}_{uu}$ *defined in lemma 1. Set* $N_r = c_r\left(1+\frac{\rho_{min}(\mathbf{K}_{uu})}{\lambda_1}\right)/L^2$ *for a positive constant* c_r. *Under the same hypotheses as stated in Assumption 5.3.2 and Assumption 5.3.3, and denote* $\widetilde{R} = \sqrt{\left(\frac{R_1}{\lambda_1}\right)^2 + \left(\frac{R_2}{\lambda_2}\right)^2}$. *Let* $\delta \in (0,1)$. *Then, with probability at least* $1-\delta$, *the following confidence bound holds for all* $\mathbf{x} \in \mathcal{D}$ *and* $t \geq 1$*:*

$$\begin{aligned}&|f(\mathbf{x}) - \mu_t^f(\mathbf{x})| \\ &\leq \sigma_t^f(\mathbf{x})\left(B + \widetilde{R}\sqrt{2I(f;\mathbf{Y}_t) - 2I(f;\mathbf{Y}_t;\mathbf{U}_r) + \mathcal{O}(1) + \log(1/\delta)}\right) \\ &\leq \sigma_t^f(\mathbf{x})\left(B + \widetilde{R}\sqrt{2\gamma_t - 2I(f;\mathbf{Y}_t;\mathbf{U}_r) + \mathcal{O}(1) + \log(1/\delta)}\right)\end{aligned}$$

Proof sketch for Lemma 5.3.8 We split the problem into two terms: The prediction error of an element f in the RKHS as assumed in Assumption 5.3.3 with noise-free observations and the noise effect. Our proof differs from most GP-based Bayesian Optimization methods, which use single-block kernel matrices. In contrast, our predictive mean and covariance function involve the inversion of a block matrix $\widehat{\mathbf{K}}_{\text{PINN}}^{-1}$, as stated in Lemma 5.3.4. We employ the block matrix inversion formula (see Appendix A, (Rasmussen, Williams, et al., 2006)) to express $\widehat{\mathbf{K}}_{\text{PINN}}^{-1}$ as four distinct matrices. Subsequently, using equivalent transformations, intermediate matrix identities, and utilizing the expression *Interaction information* provided in Lemma 5.3.6, we derive the final bound.

Remark 5.3.9. The upper bound of the confidence interval presented in Lemma 5.3.8 shares a similar form with the existing confidence interval of GP-TS as outlined in Chowdhury and Gopalan (2017). It is worth emphasizing, however, that our bound offers valuable insights into the significance of integrating partial differential equations (PDEs) to attain a tighter confidence bound. This insight can be summarized as follows: The expression $I(f;\mathbf{Y}_t) - I(f;\mathbf{Y}_t;\mathbf{U}_r)$ equals to $I(f;\mathbf{Y}_t|\mathbf{U}_r)$, which represents the expected mutual information between the function f and the observations $\mathbf{Y}_t$, given $\mathbf{U}_r$. Lemma 5.3.6 quantifies $I(f;\mathbf{Y}_t;\mathbf{U}_r)$ in terms of the kernel Gram matrices induced by black-box function and the PDE observations. As mentioned in Remark 5.3.7, the condition $I(f;\mathbf{Y}_t;\mathbf{U}_r) \geq 0$ implies that $I(f;\mathbf{Y}_t) \geq I(f;\mathbf{Y}_t|\mathbf{U}_r)$. This inequality signifies that knowing the values of the partial differential equation (PDE) component $\mathbf{U}_r$ reduces the statistical information between the observations $\mathbf{Y}_t$ and the unknown function f. In other words, knowing the values of PDE component $\mathbf{U}_r$ can diminish the number of observations $\mathbf{Y}_t$ required to estimate the unknown function f.

Remark 5.3.10. The value of $I(f;\mathbf{Y}_t;\mathbf{U}r)$ depends on the specific problem. For instance, if we assume that f is a function in RKHS with a linear kernel and $\mathcal{N}[f] =$

$\sum_{i=1}^{n} \frac{\partial f}{\partial \mathbf{x}_i}$, the lower bound for the interaction information is:

$$I(f;\mathbf{Y}_t;\mathbf{U}_r) = \Theta\left(\frac{dN_r}{dN_r+1}(1-1/T)\right) = \Theta(1),$$

which is a constant. This is because the linear kernel differential feature map sends all PDE points to the same vector in the RKHS. This example aims to show how to bound the interaction information for a known PDE.

We are now ready to present the main theoretical result of the paper:

Theorem 5.3.11. *Let* $\mathbf{K}_{uu}, \mathbf{K}_{ur}, \mathbf{K}_{ru}$, *and* $\mathbf{K}_{rr}$ *be four matrices as defined in Lemma 5.3.4. Let* $\delta \in (0,1)$. *Assume that* $\|\omega(\cdot)\|_2 \le L$ *and* $\rho_{min}(\mathbf{K}_{uu})$ *be the smallest eigenvalue of kernel matrix* $\mathbf{K}_{uu}$. *Set* $N_r = c_r\left(1+\frac{\rho_{min}(\mathbf{K}_{uu})}{\lambda_1}\right)/L^2$ *for a constant* $c_r > 0$. *Additionally, let* $\widetilde{R} = \sqrt{\left(\frac{R_1}{\lambda_1}\right)^2 + \left(\frac{R_2}{\lambda_2}\right)^2}$ *and* $I_0 = \frac{1}{2}\log\frac{\det(\mathbf{K}_{rr}+\lambda_2\mathbf{I})}{\det\left(\mathbf{K}_{rr}+\lambda_2\mathbf{I}-\mathbf{K}_{ru}\mathbf{K}_{uu}^{-1}\mathbf{K}_{ur}\right)}$. *Then with probability at least* $1-\delta$, *the regret of PINN-BO running for an unknown function* f *governed by a linear PDE, lying in the* $\mathcal{H}_{k_{\text{NTK-PINN}}}$, $\|f\|_{H_{k_{\text{NTK-PINN}}}} \le B$ *as stated in Assumption 5.3.3, after* T *iterations satisfies:*

$$R_T = \mathcal{O}\Bigg(\sqrt{Td\log BdT}\Bigg[B\sqrt{\gamma_T - I_0 + \log(2/\delta)} + \widetilde{R}\sqrt{\gamma_T}\sqrt{\gamma_T - I(f;\mathbf{Y}_T;\mathbf{U}_r) - I_0 + \log(2/\delta)}\Bigg]\Bigg)$$

5.4 Experiments

In this section, we demonstrate the effectiveness of our proposed PINN-BO algorithm through its application of synthetic benchmark optimization functions as well as real-world optimization problems. Our implementations of both problems are available at: `https://github.com/phantrdat/pinn-bo`.

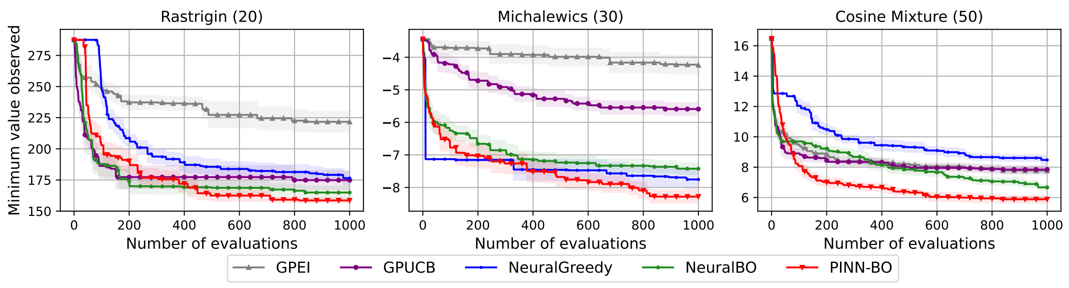


FIGURE 5.1: The optimization results for synthetic functions comparing the proposed PINN-BO with the baselines. The standard errors are shown by color shading.

5.4.1 Baselines

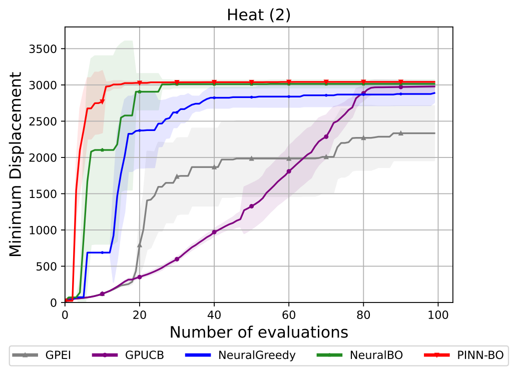
For all experiments, we compared our algorithm with common classes of surrogate models used in black-box optimization, including Gaussian Processes (GPs) and Deep Neural Networks (DNNs). For GPs, we employ the most popular strategy GP-EI Mockus, Tiesis, and Zilinskas, 1978 and GP-UCB Srinivas et al., 2009 with the Matérn Kernel. Our implementations for GP-based Bayesian Optimization baselines utilize public library GPyTorch https://gpytorch.ai/ and BOTorch https://botorch.org/. We also include two recent DNNs-based works for black-box optimization: Neural Greedy Paria et al., 2022 and NeuralBO Phan-Trong, Tran-The, and Gupta, 2023 described below:

- NeuralGreedy Paria et al., 2022 fits a neural network to the current set of observations, where the function values are randomly perturbed before learning the neural network. The learned neural network is then used as the acquisition function to determine the next query point. Since the NeuralGreedy code is not publicly available, we use our own implementation following the setting described in Paria et al., 2022 (see Appendix F.2 therein).
- NeuralBO Phan-Trong, Tran-The, and Gupta, 2023 utilizes the Thompson Sampling strategy for selecting the next evaluation point. In this approach, the mean function is estimated using the output of a fully connected deep neural network. To implement this baseline, we adhere to the configuration outlined in Section 7 in Phan-Trong, Tran-The, and Gupta, 2023.

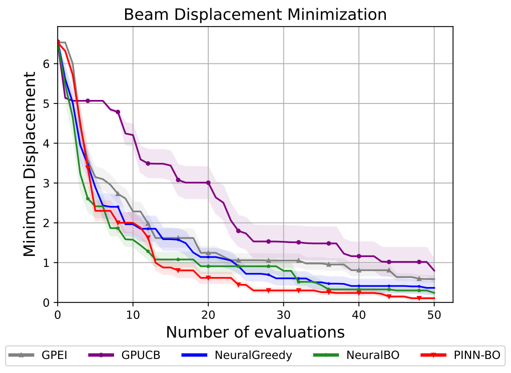
For our proposed PINN-BO algorithm, we employ a fully connected deep neural network (DNN) as the surrogate model. The network's weights are initialized with independent samples drawn from a normal distribution $\mathcal{N}(0, 1)$. The model's hyper-parameters, which include depth, width, and learning rate, are selected as follows: For each function, we perform a grid search for tuning, where each hyper-parameter tuple is trained with 50 initial points. The width is explored within the set $\{100, 200, 500\}$, while the depth and the learning rate are searched across the values $\{2, 3, 4\}$ and $\{0.001, 0.005, 0.01, 0.02, 0.05, 0.1\}$, respectively. Subsequently, we select the tuple of (depth, width, learning rate) associated with the lowest mean-square error during evaluation. To train the surrogate neural network models, we utilize the (stochastic) gradient descent optimizer along with an Exponential Learning Rate scheduler with a factor of $\gamma = 0.95$. To accelerate the training process, we update the parameters $\boldsymbol{\theta}_t$ of surrogate models in Algorithm 4 after every 10 optimization iterations with 100 epochs.

5.4.2 Synthetic Benchmark Functions

We conducted optimization experiments on five synthetic functions: DropWave (2), Styblinski-Tang (10), Rastrigin (20), Michalewics (30), and Cosine Mixture (50), where the numbers in parentheses indicate the input dimensions of each function. We selected them to ensure a diverse range of difficulty levels, as suggested by the difficulty rankings available at https://infinity77.net/global_optimization/test_functions.html. To enhance the optimization process, we incorporated the partial differential equations (PDEs) associated with these objective functions. The detailed expressions of these functions and their corresponding PDEs can be found in Section A.1 of the Appendix. Additionally, the noise in function evaluations follows a normal distribution with zero mean, and the variance is set to 1% of the function range. Results for Rastrigin, Michalewics, and Cosine Mixture functions are presented in Figure 4.1 in the main paper (the results for DropWave and Styblinski-Tang functions can be found in Section A.1 of the Appendix). All experiments reported here are averaged over 10 runs, each with random initialization. All methods begin with the same initial points. The results demonstrate that our PINN-BO is better than all other baseline methods, including GP-based BO algorithms (GP-EI, GP-UCB), and NN-based BO algorithms (NeuralBO, NeuralGreedy).



(A) The optimization results for finding the maximum temperature comparing the proposed PINN-BO with the baselines. It can be seen that, using the PDE heat equation, PINN-BO found the maximum temperature faster than all baselines.



(B) The minimum displacement on the non-uniform Euler beam under given loads $q(x)$, flexural rigidity $EI(x)$, and boundary conditions. In comparison with other baselines, our proposed PINN-BO found the location with the smallest displacement, ensuring stability when placing the load over the beam.

FIGURE 5.2: The optimization results for real-world applications comparing the proposed PINN-BO with the baselines.

5.4.3 Real-world Applications

In this section, we explore two real-world applications where the objective functions are constrained by specific partial differential equations (PDEs). We consider two tasks: (1) optimizing the Steady-State temperature distribution, satisfying the Laplace equation, and (2) optimizing the displacement of a beam element, adhering to the non-uniform Euler-Bernoulli equation. We continue to compare our proposed method with the baselines mentioned in Section 5.4.1. Due to space constraints, we briefly introduce these problems in the following subsections and present figures illustrating the results of one case for each problem. Detailed experimental setup and other results are provided in Section A.2 of the Appendix.

Optimizing Steady-State Temperature

The steady-state heat equation represents a special case of the heat equation when the temperature distribution no longer changes over time. It describes the equilibrium state of a system where the temperature is constant, and no heat is being added or removed. The steady-state heat equation is given by: $\nabla^2 T(x,y) = 0$, where x, y are spatial variables that represent the positions within a two-dimensional space. In this paper, our task is to find the best position (x, y) that maximizes temperature T within the defined domain. Formally, this problem can be defined as:

$$\max_{x,y\in\mathcal{U}} T(x,y) \text{ s.t. } \nabla^2 T(x,y) = 0,$$

We consider three different heat equations, where the solution of each problem is associated with one (unknown) boundary condition. The details and the results of optimizing these heat equation problems are provided in Section A.2. The optimization results of our PINN-BO, in comparison to other baseline methods, for the first case of the heat equation, are depicted in Figure 5.2a.

Optimizing Beam Displacement

Euler-Bernoulli beam theory is a widely used model in engineering and physics to describe the behavior of slender beams under various loads. The governing differential equation for a non-uniform Euler-Bernoulli beam is given by:

$$\frac{d^2}{dx^2}\left(EI(x)\frac{d^2w(x)}{dx^2}\right) = q(x),$$

where $EI(x)$ represents the flexural rigidity of the beam, which can vary with position x, and $w(x)$ represents the vertical displacement of the beam at position x, and $q(x)$ represents the distributed or concentrated load applied to the beam. The lower value deflection leads to a stiffer and more resilient structure, hence reducing the risk of structural failure and serviceability issues. In this paper, we consider the task of finding the position x that minimizes displacement $w(x)$. This problem can be defined as:

$$\min_{x\in\mathcal{S}} w(x) \text{ s.t. } \frac{d^2}{dx^2}\left(EI(x)\frac{d^2w(x)}{dx^2}\right) = q(x)$$

5.5 Conclusion

We introduced a novel black-box optimization scenario incorporating Partial Differential Equations (PDEs) as additional information. Our solution, PINN-BO, is a new algorithm tailored to this challenge. We conducted a theoretical analysis, demonstrating its convergence with a tighter regret bound. Through experiments involving synthetic benchmark functions and real-world optimization tasks, we validated the efficacy of our approach. Our method, due to using a neural network, is applicable to a broad set of optimization problems where the input space is complex (e.g. drug discovery), and can significantly accelerate the optimization by leveraging the auxiliary knowledge via PDEs.

Chapter 6

Conclusion

In this thesis, we have introduced three methods that apply DNNs as an alternative to GPs for the surrogate models in BO. We also extended the use of DNN to different settings of BO. These methods improved the scalability drawback of BO with GP while ensuring the optimization convergence. Further, DNNs-based Black-box Optimization shows the potential application to high-dimensional structural data. We summarize our contributions below and outline some potential future work in this area.

6.1 Contributions

To address the computational costs and scalability limitations caused by the cubic complexity of kernel inversion in GPs used for BO, this thesis focused on developing methods based on DNNs for both standard and extended BO problems.

The key idea was first applied within the standard BO setting. In **Chapter** 3, a novel black-box optimization algorithm was introduced, where the unknown function was modeled using an over-parameterized DNN. NTK theory was employed to control network uncertainty, along with the TS technique for selecting the next point for evaluation. Theoretical analysis of the algorithm's regret bound was demonstrated to show convergence and improved sample efficiency over existing methods. Additionally, the Neural-BO algorithm was shown to be scalable, with computational costs growing linearly with the number of data points, making it suitable for optimizing complex structured data like images and text. Experimental results on synthetic benchmarks and real-world optimization tasks confirmed that the algorithm outperformed current state-of-the-art methods in BO.

In **Chapter** 4, the DNN-based approach was extended to address BO with unknown, expensive constraints. Neural-CBO was introduced, where both the objective function and constraints were modeled by DNNs. EI was used as the acquisition function for the objective, while the feasible region was defined using Lower Confidence Bound (LCB) conditions. Theoretical analysis indicated that cumulative regret and constraint violations had upper bounds comparable to those of GP-based methods, under a more relaxed condition on network width. Experimental verification, conducted on synthetic and real-world tasks, demonstrated that Neural-CBO performed competitively with recent state-of-the-art approaches.

In **Chapter** 5, a novel problem setting for BO was introduced, integrating physical prior knowledge about the underlying black-box function through PDEs, using PINNs. The PINN-BO algorithm enhanced sample efficiency by efficiently handling a broad class of PDEs, thereby promoting the use of physical knowledge in BO. It was ultimately demonstrated that this approach significantly improved practical performance when compared to existing methods across various synthetic and real-world benchmark functions.

6.2 Future Directions

In this thesis, the convergence of DNN-based BO approaches is demonstrated in terms of the maximum information gain, denoted as γ_T, with respect to the NTK. As established by Kassraie and Krause (2022), γ_T is bounded by $\mathcal{O}(T^{1-d^{-1}})$, where T represents the number of observations and d the input dimensionality. In **Chapter** 3, this result was utilized to prove that the Neural-BO algorithm achieves a sub-linear regret bound, which is crucial for ensuring efficient optimization. This result is based on the assumption that the observation noise is independent of the observed values. However, in more complex scenarios where the noise depends on prior observations, deriving a sub-linear regret bound remains an open problem, suggesting an interesting avenue for future research, either by improving the upper bound of γ_T or by designing a more refined algorithm to ensure sub-linear convergence.

A related and equally promising direction for future work is the improvement of the theoretical analysis of PINN-BO, as discussed in **Chapter** 5. The current regret bound for this algorithm includes the interaction information value, which is influenced by the PDE observations, making the regret bound dependent on the specific PDE. While it has been shown that PINN-BO is capable of handling a broad class of PDE constraints, establishing a regret bound for each specific PDE class, such as linear PDEs, remains a significant challenge. In Section 5.3, a regret bound is provided for a simple class of PDEs, yet extending this analysis to more complex classes could significantly improve the understanding of how physical information, expressed through PDEs, positively influences the optimization process. By analyzing the regret bound for each PDE class, the integration of physical knowledge into Black-box Optimization can be deepened, thereby enhancing the efficiency and practicality of algorithms like PINN-BO. This approach would not only strengthen the theoretical foundations of PINN-BO but also broaden its application to real-world optimization problems governed by diverse physical laws.

Appendix A

Supplementary Material of Chapter 3

A.1 Proof of Theoretical Analysis in Chapter 3

In this part, we provide the proof for Theorem 3.3.3. The main lemmas are from Lemma A.1.2 to Lemma A.1.10. Some main lemmas require auxiliary lemmas to complete their proofs. These auxiliary lemmas are provided instantly after the main lemmas and later proved in Section A.2.

To begin, we consider the following condition of the neural network width m

Condition A.1.1. *The network width m satisfies*

$$m \geq C \max\left\{\sqrt{\lambda}L^{-3/2}[\log(TL^2\alpha)]^{3/2}, T^6L^6\log(TL/\alpha)\max\{\lambda_0^{-4}, 1\}]\right\}$$
$$m[\log m]^{-3} \geq CTL^{12}\lambda^{-1} + CT^7\lambda^{-8}L^{18}(\lambda + LT)^6 + CL^{21}T^7\lambda^{-7}(1 + \sqrt{T/\lambda})^6,$$

where C is a positive absolute constant.

Under this flexible condition of the search space as mentioned in Section 3.3, we need some new results. Following Allen-Zhu, Li, and Song (2019), we first define

$$\mathbf{h}_{i,0} = \mathbf{x}, \mathbf{h}_{i,l} = \phi(\mathbf{W}_l\mathbf{h}_{i,l-1}), l \in [L]$$

as the output of the l-th hidden layer. With this definition, we provide a norm bound of $\mathbf{h}_{i,l-1}$ as follows.

Lemma A.1.2 (Lemma 7.1, (Allen-Zhu, Li, and Song, 2019)). *If $\epsilon \in [0, 1]$, with probability at least $1 - \mathcal{O}(nL)e^{-\Omega(m\epsilon^2/L)}$, we have*

$$\forall i \in [T], l \in [L], \|\mathbf{h}_{i,l-1}\| \in [ae^{-\epsilon}, be^{\epsilon}]$$

The following lemma is the concentration property of sampling value $\widetilde{f}_t(\mathbf{x})$ from estimated mean value $h(\mathbf{x}; \boldsymbol{\theta}_{t-1})$.

Lemma A.1.3. *For any $t \in [T]$, and any finite subset $\mathcal{D}_t \subset \mathcal{D}$, pick $c_t = \sqrt{4\log t + 2\log|\mathcal{D}_t|}$. Then*

$$|\widetilde{f}_t(\mathbf{x}) - h(\mathbf{x}; \boldsymbol{\theta}_{t-1})| \leq c_t\nu_t\sigma_t(\mathbf{x}), \forall \mathbf{x} \in \mathcal{D}_t,$$

holds with probability $\geq 1 - t^{-2}$.

To prove Lemma A.1.3, we need a concentration bound on Gaussian distributions (Hoffman, Shahriari, and Freitas, 2013) as follows:

Lemma A.1.3.1 ((Hoffman, Shahriari, and Freitas, 2013)). Consider a normally distributed random variable $X \sim \mathcal{N}(\mu, \sigma^2)$ and $\beta \geq 0$. The probability that X is within a radius of $\beta\sigma$ from its mean can then be written as:

$$\mathbb{P}(|X - \mu| \leq \beta\sigma\ ) \geq 1 - \exp(\beta^2/2)$$

Proof of Lemma A.1.3. Because the sampled value $\widetilde{f}_t(\mathbf{x})$ is sampled from

$$\mathcal{N}(h(\mathbf{x}; \boldsymbol{\theta}_{t-1}), \nu_t^2 \sigma_t^2(\mathbf{x})),$$

applying the concentration property in Lemma A.1.3.1, we have:

$$\mathbb{P}(|\widetilde{f}_t(\mathbf{x}) - h(\mathbf{x}; \boldsymbol{\theta}_{t-1})| \leq c_t \nu_t \sigma_t(\mathbf{x}) | \mathcal{F}_{t-1}) \geq 1 - \exp(-c_t^2/2)$$

Taking union bound over $\mathcal{D}_t$, we have for any t:

$$\mathbb{P}(|\widetilde{f}_t(\mathbf{x}) - h(\mathbf{x}; \boldsymbol{\theta}_{t-1})| \leq c_t \nu_t \sigma_t(\mathbf{x}) | \mathcal{F}_{t-1}) \geq 1 - |\mathcal{D}_t| \exp(-c_t^2/2)$$

Picking $c_t = \sqrt{4 \log t + 2 \log |\mathcal{D}_t|}$, we get the bound:

$$\mathbb{P}(|\widetilde{f}_t(\mathbf{x}) - h(\mathbf{x}; \boldsymbol{\theta}_{t-1})| \leq c_t \nu_t \sigma_t(\mathbf{x}) | \mathcal{F}_{t-1}) \geq 1 - \frac{1}{t^2}$$

□

By combining Lemma A.1.2 with techniques used in Lemma 4.1 of Cao and Gu (2019), Lemma B.3 of Cao and Gu (2019) and Theorem 5 of Allen-Zhu, Li, and Song (2019), we achieve a concentration property of estimated value $h(\mathbf{x}; \boldsymbol{\theta}_{t-1})$ from its true value $f(\mathbf{x})$ as follows:

Lemma A.1.4. *Suppose the width of the neural network m satisfies Condition A.1.1. Given any $\alpha \in (0,1)$ and set $\eta = C(m\lambda + mLT)^{-1}$. Then for any $t \in [T]$, we have*

$$|h(\mathbf{x}; \boldsymbol{\theta}_{t-1}) - f(\mathbf{x})| \leq \nu_t \sigma_t(\mathbf{x}) + \epsilon(m), \forall \mathbf{x} \in \mathcal{D}_t,$$

holds with probability $\geq 1 - \alpha/2$ and

$$\begin{aligned}\epsilon(m) =& \frac{b}{a} C_{\epsilon,1} m^{-1/6} \lambda^{-2/3} L^3 \sqrt{\log m} + \frac{b}{a} C_{\epsilon,2} (1 - \eta m \lambda)^J \sqrt{TL/\lambda} \\ &+ \left(\frac{b}{a}\right)^3 C_{\epsilon,3} m^{-1/6} \sqrt{\log m} L^4 T^{5/3} \lambda^{-5/3} (1 + \sqrt{T/\lambda}),\end{aligned}$$

where $\{C_{\epsilon,i}\}_{i=1}^3$ are positive constants and a, b is lower and upper norm bound of input $\mathbf{x}$ as assumed in Section 3.1.

First, we define some necessary notations about linear and kernelized models.

Definition A.1.5. Let us define terms for the convenience as follows:

$$\begin{aligned}\mathbf{G}_t &= (\mathbf{g}(\mathbf{x}_1; \boldsymbol{\theta}_0), \cdots, \mathbf{g}(\mathbf{x}_t; \boldsymbol{\theta}_0)) \\ \mathbf{f}_t &= (f(\mathbf{x}_1), \cdots, f(\mathbf{x}_t))^\top \\ \mathbf{y}_t &= (y_1, \cdots, y_t)^\top \\ \boldsymbol{\epsilon}_t &= (f(\mathbf{x}_1) - y_1, \cdots, f(\mathbf{x}_t) - y_t)^\top,\end{aligned}$$

where $\boldsymbol{\epsilon}_t$ is the reward noise at time t. We recall that the definition of $\boldsymbol{\epsilon}_t$ is simply from our setting in Section 3.1. It can be verified that $\mathbf{U}_t = \lambda \mathbf{I} + \mathbf{G}_t \mathbf{G}_t^\top / m$. We also define $\mathbf{K}_t = \lambda \mathbf{I} + \mathbf{G}_t^\top \mathbf{G}_t / m$. We next reuse a lemma from Zhang et al. (2021) to bound the difference between the outputs of the neural network and the linearized model.

Lemma A.1.5.1. Suppose the network width m satisfies Condition A.1.1. Then, set $\eta = C_1(m\lambda + mLT)^{-1}$, with probability at least $1 - \alpha$ over the random initialization of $\boldsymbol{\theta}_0$, we have $\forall \mathbf{x} \in \mathcal{D}, 0 < a \leq \|\mathbf{x}\|_2 \leq b$

$$\begin{aligned}&\left| h(\mathbf{x}; \boldsymbol{\theta}_{t-1}) - \langle \mathbf{g}(\mathbf{x}; \boldsymbol{\theta}_0); \mathbf{G}_{t-1}^\top \mathbf{U}_{t-1}^{-1} \mathbf{r}_{t-1} / m \rangle \right| \\ \leq & \frac{b}{a} C_{\epsilon,1} t^{2/3} m^{-1/6} \lambda^{-2/3} L^3 \sqrt{\log m} + \frac{b}{a} C_{\epsilon,2} (1 - \eta m \lambda)^J \sqrt{tL/\lambda} \\ & + \left(\frac{b}{a}\right)^3 C_{\epsilon,3} m^{-1/6} \sqrt{\log m} L^4 t^{5/3} \lambda^{-5/3} \left(1 + \sqrt{t/\lambda}\right),\end{aligned}$$

where $\{C_{\epsilon,i}\}_{i=1}^3$ are positive constants. We provide the proof for this lemma in A.2.1.

Lemma A.1.5.2. Let $\alpha \in (0,1)$. Recall that matrix $\mathbf{U}_{t-1}$ is defined in Algorithm 1, $\mathbf{G}_{t-1}$ is defined in Definition A.1.5 and $\bullet_{t-1}$ is i.i.d sub-Gaussian observation noises up to optimization iteration $t-1$, with parameter R. Then with probability $1 - \alpha$, we have

$$\left| \mathbf{g}(\mathbf{x}; \boldsymbol{\theta}_0)^\top \mathbf{U}_{t-1}^{-1} \mathbf{G}_{t-1} \boldsymbol{\epsilon}_{t-1} / m \right| \leq \sigma_t(\mathbf{x}) \frac{R}{\sqrt{\lambda}} \sqrt{2 \log\left(\frac{1}{\alpha}\right)}$$

The proof for Lemma A.1.5.2 is given in A.2.2. Then, the following lemma provides the upper bound for approximating the NTK with the empirical gram matrix of the neural network at initialization by the maximum information gain associated with the NTK.

Lemma A.1.5.3. Let $\alpha \in (0,1)$. If the network width m satisfies $m \geq CT^6 L^6 \log(TL/\alpha)$, then with probability at least $1 - \alpha$, with the points chosen from Algorithm 1, the following holds for every $t \in [T]$:

$$\log \det\left(\mathbf{I} + \lambda^{-1} \mathbf{K}_t\right) \leq 2\gamma_t + 1,$$

where γ_t is maximum information gain associated with the kernel k_{NTK}. We provided the proof of Lemma A.1.5.3 in A.2.3.

Lemma A.1.5.4 (Lemma D.2, Kassraie and Krause, 2022). Let $\alpha \in (0,1)$. Under Assumption 3.3.1 , if the network width m satisfies $m \geq CT^6 L^6 \log(TL/\alpha)$ and f be a member of $\mathcal{H}_{k_{\text{NTK}}}$ with bounded RKHS norm $\|f\|_{k_{\text{NTK}}} \leq B$, then with probability at least $1 - \alpha$, there exists $\mathbf{w} \in \mathbb{R}^p$ such that

$$f(\mathbf{x}) = \langle \mathbf{g}(\mathbf{x}; \boldsymbol{\theta}_0), \mathbf{w} \rangle, \|\mathbf{w}\|_2 \leq \sqrt{\frac{2}{m}} B$$

We remark that in our proofs, we assume $k_{\text{NTK}}(\mathbf{x}, \mathbf{x}) \leq 1$ for simplicity. Now we are going on to prove Lemma A.1.4

Proof of Lemma A.1.4. First of all, since m satisfies Condition A.1.1, then with the choice of η, the condition required in Lemma A.1.5.1 - A.1.5.3 are satisfied. Thus, taking a union bound, we have with probability at least $1 - 3\alpha$, that the bounds provided by these lemmas hold. As we assume that f is in RKHS $\mathcal{H}_{k_{\text{NTK}}}$ with

NTK kernel, and $\mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0)/\sqrt{m}$ can be considered as finite approximation of $\varphi(\cdot)$, the feature map of the NTK from $\mathbb{R}^d \to \mathcal{H}_{k_{\text{NTK}}}$. From Lemma A.1.5.4 , there exists $\mathbf{w} \in \mathbb{R}^p$ such that $f(\mathbf{x}) = \langle \mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0), \mathbf{w}\rangle = \mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0)^\top \mathbf{w}$. Then for any $t \in [T]$, we will first provide the difference between the target function and the linear function $\langle \mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0); \mathbf{U}_{t-1}^{-1}\mathbf{G}_{t-1}\mathbf{r}_{t-1}/m\rangle$ as:

$$
\begin{aligned}
&\left| f(\mathbf{x}) - \langle \mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0); \mathbf{U}_{t-1}^{-1}\mathbf{G}_{t-1}\mathbf{r}_{t-1}/m\rangle \right| \\
&= \left| f(\mathbf{x}) - \mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0)^\top \mathbf{U}_{t-1}^{-1}\mathbf{G}_{t-1}\mathbf{r}_{t-1}/m \right| \\
&\le \left| f(\mathbf{x}) - \mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0)^\top \mathbf{U}_{t-1}^{-1}\mathbf{G}_{t-1}\mathbf{f}_{t-1}/m \right| + \left| \mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0)^\top \mathbf{U}_{t-1}^{-1}\mathbf{G}_{t-1}\boldsymbol{\epsilon}_{t-1}/m \right| \\
&= \left| \mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0)^\top \mathbf{w} - \mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0)^\top \mathbf{U}_{t-1}^{-1}\mathbf{G}_{t-1}\mathbf{G}_{t-1}^\top \mathbf{w}/m\rangle \right| + \left| \mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0)^\top \mathbf{U}_{t-1}^{-1}\mathbf{G}_{t-1}\boldsymbol{\epsilon}_{t-1}/m \right| \\
&= \left| \mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0)^\top \left(\mathbf{I} - \mathbf{U}_{t-1}^{-1}\mathbf{G}_{t-1}\mathbf{G}_{t-1}^\top/m \right) \mathbf{w} \right| + \left| \mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0)^\top \mathbf{U}_{t-1}^{-1}\mathbf{G}_{t-1}\boldsymbol{\epsilon}_{t-1}/m \right| \\
&= \left| \mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0)^\top \left(\mathbf{I} - \mathbf{U}_{t-1}^{-1}\left(\mathbf{U}_{t-1} - \lambda \mathbf{I}\right) \right) \mathbf{w} \right| + \left| \mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0)^\top \mathbf{U}_{t-1}^{-1}\mathbf{G}_{t-1}\boldsymbol{\epsilon}_{t-1}/m \right| \\
&= \left| \lambda \mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0)^\top \mathbf{U}_{t-1}^{-1}\mathbf{w} \right| + \left| \mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0)^\top \mathbf{U}_{t-1}^{-1}\mathbf{G}_{t-1}\boldsymbol{\epsilon}_{t-1}/m \right| \\
&\le \|\mathbf{w}\|_{k_{\text{NTK}}} \left\| \lambda \mathbf{U}_{t-1}^{-1}\mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0) \right\|_{k_{\text{NTK}}} + \left| \mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0)^\top \mathbf{U}_{t-1}^{-1}\mathbf{G}_{t-1}\boldsymbol{\epsilon}_{t-1}/m \right| \\
&\le \|\mathbf{w}\|_{k_{\text{NTK}}} \sqrt{\lambda \mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0)^\top \mathbf{U}_{t-1}^{-1}\mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0)} + \left| \mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0)^\top \mathbf{U}_{t-1}^{-1}\mathbf{G}_{t-1}\boldsymbol{\epsilon}_{t-1}/m \right| \\
&\le \sqrt{2}B\sigma_t(\mathbf{x}) + \sigma_t(\mathbf{x}) \frac{R}{\sqrt{\lambda}} \sqrt{2\log\left(\frac{1}{\alpha}\right)}
\end{aligned} \tag{A.1}
$$

where the first inequality uses triangle inequality and the fact that $\mathbf{r}_{t-1} = \mathbf{f}_{t-1} + \boldsymbol{\epsilon}_{t-1}$. The second inequality is from the reproducing property of function relying on RKHS, and the fourth equality is from the verification noted in Definition A.1.5. The last inequality directly uses the results from Lemma A.1.5.2 and Lemma A.1.5.4. We have a more compact form of the inequality A.1 as:

$$
|f(\mathbf{x}) - \langle \mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0); \mathbf{U}_{t-1}^{-1}\mathbf{G}_{t-1}\mathbf{r}_{t-1}/m\rangle| \le \nu_t \sigma_t(\mathbf{x}),
$$

where we set $\nu_t = \sqrt{2}B + \frac{R}{\sqrt{\lambda}}\sqrt{2\log(1/\alpha)}$. Then, by combining this bound with Lemma A.1.5.1, we have

$$
\begin{aligned}
|h(\mathbf{x};\boldsymbol{\theta}_{t-1}) - f(\mathbf{x})| \le \nu_t\sigma_t(\mathbf{x}) &+ \frac{b}{a} C_{\epsilon,1} t^{2/3} m^{-1/6} \lambda^{-2/3} L^3 \\
&+ \frac{b}{a} C_{\epsilon,2} (1 - \eta m \lambda)^J \sqrt{tL/\lambda} \\
&+ \left(\frac{b}{a}\right)^3 C_{\epsilon,3} m^{-1/6} \sqrt{\log m} L^4 t^{5/3} \lambda^{-5/3} \left(1 + \sqrt{t/\lambda}\right) \\
\le \nu_t\sigma_t(\mathbf{x}) &+ \epsilon(m)
\end{aligned}
$$

By setting α to $\alpha/3$ (required by the union bound discussed at the beginning of the proof) and taking $t = T$, we get the result presented in Lemma A.1.4. □

The next lemma gives a lower bound of the probability that the sampled value $\widetilde{f}_t(\mathbf{x})$ is larger than the true function value up to the approximation error $\epsilon(m)$.

Lemma A.1.6. *For any* $t \in [T], \mathbf{x} \in D$, *we have* $\mathbb{P}(\widetilde{f}_t(\mathbf{x}) + \epsilon(m) > f(\mathbf{x})) \ge (4e\pi)^{-1}$.

Proof of Lemma A.1.6. Following proof style of Lemma 8 in Zhou, Li, and Gu (2020), using Lemma A.1.4 and Gaussian anti-concentration property, we have

$$\begin{aligned}&\mathbb{P}(\widetilde{f}_t(\mathbf{x}) + \epsilon(m) > f(\mathbf{x})|\mathcal{F}_{t-1})\\ =&\mathbb{P}\left(\frac{\widetilde{f}_t(\mathbf{x}) - h(\mathbf{x};\boldsymbol{\theta}_{t-1})}{\nu_t\sigma_t(\mathbf{x})} > \frac{|f(\mathbf{x}) - h(\mathbf{x};\boldsymbol{\theta}_{t-1})| - \epsilon(m)}{\nu_t\sigma_t(\mathbf{x})}\middle| \mathcal{F}_{t-1}\right) \geq \frac{1}{4e\pi}\end{aligned}$$

□

For any step t, we consider how the standard deviation of the estimates for each point is, in comparison with the standard deviation for the optimal value. Following Zhang et al., 2021, we divide the discretization $\mathcal{D}_t$ into two sets: saturated and unsaturated sets. For more details, we define the set of saturated points as:

$$S_t = \{\forall \mathbf{x} \in \mathcal{D}_t, f([\mathbf{x}^*]_t) - f(\mathbf{x}) \geq (1 + c_t)\nu_t\sigma_t(\mathbf{x}) + 2\epsilon(m)\} \tag{A.2}$$

The following lemma shows that, the algorithm can pick unsaturated points $\mathbf{x}_t$ with high probability.

Lemma A.1.7 (Lemma 4.5, Zhou, Li, and Gu, 2020). *Let $\mathbf{x}_t$ be the chosen point at step $t \in [T]$. Then, $\mathbb{P}(\mathbf{x}_t \notin S_t|\mathcal{F}_{t-1}) \geq \frac{1}{4e\sqrt{\pi}} - \frac{1}{t^2}$*

The next lemma bounds the expectation of instantaneous regret at each round conditioned on history $\mathcal{F}_{t-1}$.

Lemma A.1.8. *Suppose the width of the neural network m satisfies Condition A.1.1. Set $\eta = C_1(m\lambda + mLT)^{-1}$. Then with probability at least $1 - \alpha$, we have for all $t \in [T]$ that*

$$\mathbb{E}[f(\mathbf{x}^*) - f(\mathbf{x}_t)|\mathcal{F}_{t-1}] \leq C_2(1 + c_t)\nu_t\sqrt{L}\mathbb{E}[\min(\sigma_t(\mathbf{x}_t), B)|\mathcal{F}_{t-1}] + 4\epsilon(m) + \frac{2B+1}{t^2}$$

where C_1, C_2 are absolute constants.

Proof of Lemma A.1.8. This proof inherits the proof of Lemma 4.6 Zhang et al., 2021, and by using the result of unsaturated point $\mathbf{x}_t$ in Lemma A.1.7, along with $|f(x)| = |\langle f, k(x,\cdot)\rangle| \leq B$ instead of $|f(x)| \leq 1$ as in Zhang et al., 2021, we have the following result:

$$\begin{aligned}&\mathbb{E}[f([\mathbf{x}^*]_t) - f(\mathbf{x}_t) \mid \mathcal{F}_{t-1}]\\ \leq& 44e\sqrt{\pi}(1 + c_t)\nu_t C_1\sqrt{L}\mathbb{E}[\min\{\sigma_t(\mathbf{x}_t), B\} \mid \mathcal{F}_{t-1}] + 4\epsilon(m) + \frac{2B}{t^2}\end{aligned}$$

Now using Eqn 3.1, we have the instantaneous regret at round t

$$r_t = f(\mathbf{x}^*) - f([\mathbf{x}^*]_t) + f([\mathbf{x}^*]_t) - f(\mathbf{x}_t) \leq \frac{1}{t^2} + f([\mathbf{x}^*]_t) - f(\mathbf{x}_t)$$

Taking conditional expectation, we have the result as stated in Lemma A.1.8.

□

The next lemma bounds the cumulative regret $\sum_{t=1}^{T} r_t$ after T iterations.

Lemma A.1.9. *Suppose the width of the neural network m satisfies Condition A.1.1. Then set $\eta = C_1(m\lambda + mLT))^{-1}$, we have, with probability at least $1-\alpha$, that*

$$\sum_{t=1}^{T} f(\mathbf{x}^*) - f(\mathbf{x}_t) \leq 4T\epsilon(m) + \frac{(2B+1)\pi^2}{6} + C_2(1+c_T)\nu_t \sum_{t=1}^{T} \min(\sigma_t(\mathbf{x}_t), B) + (4B + C_3(1+c_T)\nu_t L + 4\epsilon(m))\sqrt{2\log(1/\alpha)T}$$

where C_1, C_2, C_3 are absolute constants.

Proof of Lemma A.1.9. Similar to Lemma A.1.8, we utilize the proof of Lemma 4.7 in Zhang et al. (2021) or equivalent proof of Lemma 13 in Chowdhury and Gopalan (2017). Then with $f(\mathbf{x}) \leq B$, we have

$$\sum_{i=1}^{T} r_t \leq 4T\epsilon(m) + (2B+1)\sum_{i=1}^{T} t^{-2} + C_1(1+c_T)\nu_t \sum_{i=1}^{T} \min(\sigma_t(\mathbf{x}_t), B) + (4B + C_1C_2(1+c_T)\nu_t L + 4\epsilon(m))\sqrt{2\log(1/\alpha)T}$$

□

The next lemma gives a bound on the sum of variance $\sum_{i=1}^{T} \min(\sigma_t(\mathbf{x}_t), B)$ which appears in Lemma A.1.9.

Lemma A.1.10. *Suppose the width of the neural network m satisfies Condition A.1.1. Then set $\eta = C_1(m\lambda + mLT))^{-1}$, we have, with probability at least $1-\alpha$, that*

$$\sum_{i=1}^{T} \min(\sigma_t(\mathbf{x}_t), B) \leq \sqrt{\frac{\lambda BT}{\log(B+1)}(2\gamma_T + 1)}$$

To prove lemma A.1.10, we first need to utilize a technical lemma:

Lemma A.1.10.1. Let $\{\mathbf{v}_t\}_{t=1}^{\infty}$ be a sequence in $\mathbb{R}^p$, and define $\mathbf{V}_t = \lambda\mathbf{I} + \sum_{i=1}^{t} \mathbf{v}_i\mathbf{v}_i^\top$ and B is a positive constant. If $\lambda \geq 1$, then

$$\sum_{i=1}^{T} \min\{\mathbf{v}_t^\top \mathbf{V}_{t-1}^{-1}\mathbf{v}_{t-1}, B\} \leq \frac{B}{\log(B+1)} \log\det\left(\mathbf{I} + \lambda^{-1}\sum_{i=1}^{T}\mathbf{v}_i\mathbf{v}_i^\top\right)$$

A.2.4 provided the proof for Lemma A.1.10.1. Now, we start to prove Lemma A.1.10

Proof of Lemma A.1.10. From Cauchy-Schwartz inequality, we have

$$\sum_{i=1}^{T} \min\{\sigma_t(\mathbf{x}_t), B\} \leq \sqrt{T\sum_{i=1}^{T} \min\{\sigma_t^2(\mathbf{x}_t), B\}}$$

We also have,

$$
\begin{aligned}
\sum_{i=1}^{T}\min\{\sigma_t^2(\mathbf{x}_t), B\} &\leq \lambda \sum_{i=1}^{T}\min\{\mathbf{g}(\mathbf{x}_t;\boldsymbol{\theta}_0)^\top \mathbf{U}_{t-1}^{-1}\mathbf{g}(\mathbf{x}_t;\boldsymbol{\theta}_0)/m, B\} \\
&\leq \frac{\lambda B}{\log(B+1)} \log\det\left(\mathbf{I} + \lambda^{-1}\sum_{i=1}^{T}\mathbf{g}(\mathbf{x}_t;\boldsymbol{\theta}_0)\mathbf{g}(\mathbf{x}_t;\boldsymbol{\theta}_0)^\top/m\right) \\
&= \frac{\lambda B}{\log(B+1)} \log\det\left(\mathbf{I} + \lambda^{-1}\mathbf{G}_T\mathbf{G}_T^\top/m\right) \\
&= \frac{\lambda B}{\log(B+1)} \log\det\left(\mathbf{I} + \lambda^{-1}\mathbf{G}_T^\top\mathbf{G}_T/m\right) \\
&= \frac{\lambda B}{\log(B+1)} \log\det\left(\mathbf{I} + \lambda^{-1}\mathbf{K}_T\right) \\
&\leq \frac{\lambda B}{\log(B+1)}(2\gamma_T + 1)
\end{aligned}
$$

where the first inequality moves the positive parameter λ outside the min operator. Then the second inequality utilizes Lemma A.1.10.1, the first equality use the expression of $\mathbf{G}_t$ in 1, the second equality is from the fact that $\det(\mathbf{I} + \mathbf{A}\mathbf{A}^\top) = \det(\mathbf{I} + \mathbf{A}^\top\mathbf{A})$, and the last equality uses the definition of $\mathbf{K}_T$ in Definition A.1.5 and the last inequality uses Lemma A.1.5.3. Finally, we have,

$$
\sum_{i=1}^{T}\min\{\sigma_t(\mathbf{x}_t), B\} \leq \sqrt{\frac{\lambda BT}{\log(B+1)}(2\gamma_T + 1)}.
$$

□

Finally, we repeat to emphasize the proof of Theorem 4.2.2 given in Section 3.3.

Proof of Theorem 4.2.2. With probability at least $1-\alpha$, we have

$$
\begin{aligned}
R_T &= \sum_{t=1}^{T} f(\mathbf{x}^*) - f(\mathbf{x}_t) \\
&= \sum_{t=1}^{T} [f(\mathbf{x}^*) - f([\mathbf{x}^*]_t)] + [f([\mathbf{x}^*]_t) - f(\mathbf{x}_t)] \\
&\leq 4T\epsilon(m) + \frac{(2B+1)\pi^2}{6} + \bar{C}_1(1+c_T)\nu_T\sqrt{L}\sum_{i=1}^{T}\min(\sigma_t(\mathbf{x}_t), B) \\
&\quad + (4 + \bar{C}_2(1+c_T)\nu_T L + 4\epsilon(m))\sqrt{2\log(1/\alpha)T} \\
&\leq \bar{C}_1(1+c_T)\nu_T\sqrt{L}\sqrt{\frac{\lambda BT}{\log(B+1)}(2\gamma_T+1)} + \frac{(2B+1)\pi^2}{6} + 4T\epsilon(m) \\
&\quad + 4\epsilon(m)\sqrt{2\log(1/\alpha)T} + (4 + \bar{C}_2(1+c_T)\nu_T L)\sqrt{2\log(1/\alpha)T} \\
&= \bar{C}_1(1+c_T)\nu_t\sqrt{L}\sqrt{\frac{\lambda BT}{\log(B+1)}(2\gamma_T+1)} + \frac{(2B+1)\pi^2}{6} \\
&\quad + \epsilon(m)(4T + \sqrt{2\log(1/\alpha)T}) + (4 + \bar{C}_2(1+c_T)\nu_t L)\sqrt{2\log(1/\alpha)T}
\end{aligned}
$$

The first inequality is due to Lemma A.1.9, which provides the bound for cumulative regret R_T in terms of $\sum_{t=1}^{T}\min(\sigma_t(\mathbf{x}_t), B)$. The second inequality further provides the bound of term $\sum_{t=1}^{T}\min(\sigma_t(\mathbf{x}_t), B)$ due to Lemma A.1.10, while the last equality rearranges addition. Picking $\eta = (m\lambda + mLT)^{-1}$ and $J = (1 + LT/\lambda)\left(\log(C_{\epsilon,2}) + \log\left(T^3L\lambda^{-1}\log(1/\alpha)\right)\right)$, we have

$$\begin{aligned}
&\frac{b}{a}C_{\epsilon,2}(1-\eta m\lambda)^J\sqrt{TL/\lambda}\left(4T+\sqrt{2\log(1/\alpha)T}\right)\\
=&\frac{b}{a}C_{\epsilon,2}\left(1-\frac{1}{1+LT/\lambda}\right)^J\left(4T+\sqrt{2\log(1/\alpha)T}\right)\\
=&\frac{b}{a}C_{\epsilon,2}e^{-\left(\log(C_{\epsilon,2})+\log\left(T^3L\lambda^{-1}\log(1/\alpha)\right)\right)}\left(4T+\sqrt{2\log(1/\alpha)T}\right)\\
=&\frac{b}{a}\frac{1}{C_{\epsilon,2}}.T^{-3}L^{-1}\lambda\log^{-1}(1/\alpha)\left(4T+\sqrt{2\log(1/\alpha)T}\right)\le\frac{b}{a}
\end{aligned}$$

Then choosing m that satisfies:

$$\left(\frac{b}{a}C_{\epsilon,1}m^{-1/6}\lambda^{-2/3}L^3\sqrt{\log m}+\left(\frac{b}{a}\right)^3C_{\epsilon,3}m^{-1/6}\sqrt{\log m}L^4T^{5/3}\lambda^{-5/3}(1+\sqrt{T/\lambda})\right)\left(4T+\sqrt{2\log(1/\alpha)T}\right)\le\left(\frac{b}{a}\right)^3$$

We finally achieve the bound of R_T as:

$$\begin{aligned}
R_T \le\ &\bar{C}_1(1+c_T)\nu_T\sqrt{L}\sqrt{\frac{\lambda BT}{\log(B+1)}(2\gamma_T+1)}\\
&+(4+\bar{C}_2(1+c_T)\nu_TL)\sqrt{2\log(1/\alpha)T}+\frac{(2B+1)\pi^2}{6}+\frac{b(a^2+b^2)}{a^3}
\end{aligned}$$

□

A.2 Proof of Auxiliary Lemmas

A.2.1 Proof of Lemma A.1.5.1

We strictly inherit the technique used in Lemma 4.1 Cao and Gu, 2019, Lemma B.3 Cao and Gu, 2019 and Theorem 5 Allen-Zhu, Li, and Song, 2019, combine with Lemma A.1.2 to achieve following results:

Lemma A.2.0.1. There exists positive constants $\{C_i\}_{i=1}^{2}$ and $\bar{C}_{\epsilon,1}$ such that for any $\alpha \in (0, 1]$, if τ satisfies that:

$$C_1m^{-3/2}L^{-3/2}\left[\log\left(TL^2/\alpha\right)\right]^{3/2}\le\tau\le C_2L^{-6}[\log m]^{-3/2},$$

then with probability at least $1-\alpha$ over the randomness of $\boldsymbol{\theta}_0$ and $\mathbf{x} \in \{\mathbf{x}_1, \dots \mathbf{x}_T\}$ with $0 < a \le \|\mathbf{x}\|_2 \le b$:

1. For all $\hat{\boldsymbol{\theta}}, \widetilde{\boldsymbol{\theta}}$ satisfying $\|\boldsymbol{\theta}_0-\hat{\boldsymbol{\theta}}\|_2\le\tau$, $\left\|\boldsymbol{\theta}_0-\widetilde{\boldsymbol{\theta}}\right\|_2\le\tau$,

$$\left|h(\mathbf{x};\hat{\boldsymbol{\theta}})-h(\mathbf{x};\widetilde{\boldsymbol{\theta}})-\langle\mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0),\hat{\boldsymbol{\theta}}-\widetilde{\boldsymbol{\theta}}\rangle\right|\le\frac{b}{a}\bar{C}_{\epsilon,1}\tau^{4/3}L^3\sqrt{m\log m}$$

2. For all $\boldsymbol{\theta}$ satisfying $\|\boldsymbol{\theta}_0 - \boldsymbol{\theta}\|_2 \le \tau$,

$$\|\mathbf{g}(\mathbf{x};\boldsymbol{\theta}) - \mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0)\|_2 \le \frac{b}{a}\bar{C}_{\epsilon,1}\sqrt{\log m}\tau^{1/3}L^3\|\mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0)\|_2$$

3. For all $\boldsymbol{\theta}$ satisfying $\|\boldsymbol{\theta}_0 - \boldsymbol{\theta}\|_2 \le \tau$,

$$\|\mathbf{g}(\mathbf{x};\boldsymbol{\theta})\|_F \le \frac{b}{a}\bar{C}_{\epsilon,1}\sqrt{mL}$$

The next lemma controls the difference between the parameter of neural network learned by gradient descent and the theoretical optimal solution of the linearized network.

Lemma A.2.0.2. There exists constants $\{C_i\}_{i=1}^4$ and $\bar{C}_{\epsilon,2}$ such that for any $\alpha \in (0,1]$, if η, m satisfy that for all $t \in [T]$

$$\begin{aligned}
&2\sqrt{t/\lambda} \ge C_1 m^{-1}L^{-3/2}\left[\log(TL^2/\alpha)\right]^{3/2},\\
&2\sqrt{t/\lambda} \le C_2 \min\left\{m^{1/2}L^{-6}[\log m]^{-3/2}, m^{7/8}(\lambda^2\eta^2L^{-6}t^{-1}(\log m)^{-1}\right\},\\
&\eta \le C_3(m\lambda + tmL)^{-1},\\
&m^{1/6} \ge C_4\sqrt{\log m}L^{7/2}t^{7/6}\lambda^{-7/6}(1 + \sqrt{t/\lambda}),
\end{aligned}$$

then with probability at least $1-\alpha$ over the randomness of $\boldsymbol{\theta}_0$, $\mathbf{x} \in \{\mathbf{x}_1, \dots \mathbf{x}_T\}$ with $0 < a \le \|\mathbf{x}\|_2 \le b$, we have $\|\boldsymbol{\theta}_0 - \boldsymbol{\theta}\|_2 \le 2\sqrt{t/(m\lambda)}$ and

$$\begin{aligned}
&\left\|\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}_0 - \mathbf{U}_{t-1}^{-1}\mathbf{G}_{t-1}\mathbf{r}_{t-1}/m\right\|_2\\
&\le (1-\eta m\lambda)^J\sqrt{t/(m\lambda)} + \left(\frac{b}{a}\right)^2\bar{C}_{\epsilon,2}m^{-2/3}\sqrt{\log m}L^{7/2}t^{5/3}\lambda^{-5/3}(1+\sqrt{t/\lambda})
\end{aligned}$$

Proof of Lemma A.1.5.1. Set $\tau = 2\sqrt{t/m\lambda}$, it can be verified that τ satisfies condition in A.2.0.1. Therefore, by inequality 1 in Lemma A.2.0.1, and with the initialization of the network $f(\mathbf{x};\boldsymbol{\theta}_0) = 0$, there exists a constant $C_{\epsilon,1}$ such that

$$|h(\mathbf{x};\boldsymbol{\theta}_{t-1}) - \langle \mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0), \boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}_0\rangle| \le \frac{b}{a}C_{\epsilon,1}t^{2/3}m^{-1/6}\lambda^{-2/3}L^3\sqrt{m\log m}$$

Then, we have the difference between the linearized model and the theoretical optimal solution of kernelized regression:

$$\begin{aligned}
&|\langle \mathbf{g}(\mathbf{x}_t;\boldsymbol{\theta}_0), \boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}_0\rangle - \langle \mathbf{g}(\mathbf{x}_t;\boldsymbol{\theta}_0), \mathbf{U}_{t-1}^{-1}\mathbf{G}_{t-1}\mathbf{r}_{t-1}/m\rangle|\\
&\le \|\mathbf{g}(\mathbf{x}_t;\boldsymbol{\theta}_0)\|_2\left\|\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}_0 - \mathbf{U}_{t-1}^{-1}\mathbf{G}_{t-1}\mathbf{r}_{t-1}/m\right\|_2\\
&\le \frac{b}{a}\bar{C}_{\epsilon,1}\sqrt{mL}\Bigg((1-\eta m\lambda)^J\sqrt{t/(m\lambda)}\\
&\quad + \left(\frac{b}{a}\right)^2\bar{C}_{\epsilon,2}m^{-2/3}\sqrt{\log m}L^{7/2}t^{5/3}\lambda^{-5/3}(1+\sqrt{t/\lambda})\Bigg)\\
&\le \frac{b}{a}C_{\epsilon,2}(1-\eta m\lambda)^J\sqrt{tL/(\lambda)} + \left(\frac{b}{a}\right)^3 C_{\epsilon,3}m^{-1/6}\sqrt{\log m}L^4t^{5/3}\lambda^{-5/3}\left(1+\sqrt{t/\lambda}\right),
\end{aligned}$$

where the first inequality is from (3) and Lemma A.2.0.2, with $C_{\epsilon,2} = \bar{C}_{\epsilon,1}$ and $C_{\epsilon,3} = \bar{C}_{\epsilon,1}\bar{C}_{\epsilon,2}$. Finally, we have

$$\begin{aligned}
&\left|h(\mathbf{x};\boldsymbol{\theta}_{t-1}) - \langle \mathbf{g}(\mathbf{x}_t;\boldsymbol{\theta}_0), \mathbf{U}_{t-1}^{-1}\mathbf{G}_{t-1}\mathbf{r}_{t-1}/m\rangle\right| \\
&\leq |h(\mathbf{x};\boldsymbol{\theta}_{t-1}) - \langle \mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0), \boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}_0\rangle| \\
&+ |\langle \mathbf{g}(\mathbf{x}_t;\boldsymbol{\theta}_0), \boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}_0\rangle - \langle \mathbf{g}(\mathbf{x}_t;\boldsymbol{\theta}_0), \mathbf{U}_{t-1}^{-1}\mathbf{G}_{t-1}\mathbf{r}_{t-1}/m\rangle| \\
&\leq \frac{b}{a}C_{\epsilon,1}t^{2/3}m^{-1/6}\lambda^{-2/3}L^3\sqrt{m\log m} + \frac{b}{a}C_{\epsilon,2}(1-\eta m\lambda)^J\sqrt{tL/(\lambda)} \\
&+ \left(\frac{b}{a}\right)^3 C_{\epsilon,3}m^{-1/6}\sqrt{\log m}L^4t^{5/3}\lambda^{-5/3}\left(1+\sqrt{t/\lambda}\right)
\end{aligned}$$

as stated in Lemma A.1.5.1. □

A.2.2 Proof of Lemma A.1.5.2

Proof of Lemma A.1.5.2. We bound the noise-affected term using the sub-Gaussianity assumption. Let $\mathbf{Z}_{t-1}^\top(\mathbf{x}) = \mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0)^\top\mathbf{U}_{t-1}^{-1}\mathbf{G}_{t-1}/m$. We will show that $\mathbf{Z}_{t-1}^\top(\mathbf{x})\boldsymbol{\epsilon}_{t-1}$ is a sub-Gaussian random variable whose moment generating function is bounded by that of a Gaussian random variable with variance $\frac{R^2\sigma_t^2(\mathbf{x})}{\lambda}$.

Following the proof style in Vakili et al., 2021, we bound the noise-affected term $\mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0)^\top\mathbf{U}_{t-1}^{-1}\mathbf{G}_{t-1}\boldsymbol{\epsilon}_{t-1}/m = \mathbf{Z}_{t-1}^\top(\mathbf{x})\boldsymbol{\epsilon}_{t-1}$. Let $\zeta_i(\mathbf{x}) = [\mathbf{Z}_{t-1}(\mathbf{x})]_i$, then we have:

$$\begin{aligned}
\mathbb{E}\left[\exp\left(\mathbf{Z}_{t-1}^\top(\mathbf{x})\boldsymbol{\epsilon}_{t-1}\right)\right] &= \mathbb{E}\left[\exp\sum_{i=1}^{t-1}\zeta_i(\mathbf{x})\boldsymbol{\epsilon}_i\right] \\
&= \prod_{i=1}^{t-1}\mathbb{E}[\exp\zeta_i(\mathbf{x})\boldsymbol{\epsilon}_i] \\
&\leq \prod_{i=1}^{t-1}\exp\left(\frac{R^2\zeta_i^2(\mathbf{x})}{2}\right) \qquad\qquad \text{(A.3)} \\
&= \exp\left(\frac{R^2\sum_{i=1}^{t-1}\zeta_i^2(\mathbf{x})}{2}\right) \\
&= \exp\left(\frac{R^2\|\mathbf{Z}_{t-1}(\mathbf{x})\|_2^2}{2}\right),
\end{aligned}$$

where the second equation is the consequence of independence of $\zeta_i(\mathbf{x})\boldsymbol{\epsilon}_i$, which directly utilizes our i.i.d noise assumption which is mentioned in Assumption 3.3.2. The first inequality holds by the concentration property of the sub-Gaussian noise random variable with parameter R. Then we bound:

$$\begin{aligned}
\|\mathbf{Z}_{t-1}(\mathbf{x})\|_2^2 &= \mathbf{Z}_{t-1}(\mathbf{x})\mathbf{Z}_{t-1}^\top(\mathbf{x}) \\
&= \frac{1}{m^2}\mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0)^\top\mathbf{U}_{t-1}^{-1}\mathbf{G}_{t-1}\mathbf{G}_{t-1}^\top\mathbf{U}_{t-1}^{-1}\mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0) \\
&= \frac{1}{m}\mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0)^\top\mathbf{U}_{t-1}^{-1}(\mathbf{U}_{t-1}-\lambda\mathbf{I})\mathbf{U}_{t-1}^{-1}\mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0) \\
&= \frac{1}{m}\mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0)^\top(\mathbf{I}-\lambda\mathbf{U}_{t-1}^{-1})\mathbf{U}_{t-1}^{-1}\mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0) \\
&= \frac{1}{m}\mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0)^\top\left[\mathbf{U}_{t-1}^{-1}\mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0)-\lambda\mathbf{U}_{t-1}^{-2}\mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0)\right] \\
&= \frac{1}{m}\mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0)^\top\mathbf{U}_{t-1}^{-1}\mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0)-\frac{\lambda}{m}\mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0)^\top\mathbf{U}_{t-1}^{-2}\mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0) \\
&= \frac{1}{\lambda}\sigma_t^2(\mathbf{x})-\frac{\lambda}{m}\mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0)^\top\mathbf{U}_{t-1}^{-2}\mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0) \\
&= \frac{1}{\lambda}\sigma_t^2(\mathbf{x})-\frac{\lambda}{m}\|\mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0)\mathbf{U}_{t-1}^{-2}\|_2^2 \\
&\le \frac{1}{\lambda}\sigma_t^2(\mathbf{x}),
\end{aligned} \tag{A.4}$$

where the second equality is from definition of $\mathbf{Z}_{t-1}$ and the third inequality is from the note of Definition A.1.5. The inequality is from the fact that $\frac{\lambda}{m}\left\|\mathbf{g}(\mathbf{x};\boldsymbol{\theta}_0)\mathbf{U}_{t-1}^{-2}\right\|^2 \ge 0, \forall \mathbf{x}, t$. Replace inequality A.4 to equation A.3 then we have

$$\mathbb{E}\left[\exp\left(\mathbf{Z}_{t-1}^\top(\mathbf{x})\boldsymbol{\epsilon}_{t-1}\right)\right] \le \exp\left(\frac{R^2\sigma_t^2(\mathbf{x})}{2\lambda}\right)$$

Thus, using Chernoff-Hoeffding inequality Antonini, Kozachenko, and Volodin, 2008, we have with probabilty at least $1-\alpha$:

$$\left|\mathbf{Z}_{t-1}^\top(\mathbf{x})\boldsymbol{\epsilon}_{t-1}\right| \le \frac{R\sigma_t(\mathbf{x})}{\sqrt{\lambda}}\sqrt{2\log\left(\frac{1}{\alpha}\right)}$$

□

A.2.3 Proof of Lemma A.1.5.3

Proof of Lemma A.1.5.3. From the definition of $\mathbf{K}_t$ and Lemma B.7 Zhou, Li, and Gu, 2020, we have that

$$\begin{aligned}
\log\det\left(\mathbf{I}+\lambda^{-1}\mathbf{K}_t\right) &= \log\det\left(\mathbf{I}+\sum_{i=1}^{t}\mathbf{g}(\mathbf{x}_t;\boldsymbol{\theta}_0)\mathbf{g}(\mathbf{x}_t;\boldsymbol{\theta}_0)^\top/(m\lambda)\right)\\
&= \log\det\left(\mathbf{I}+\lambda^{-1}\mathbf{H}_t+\lambda^{-1}(\mathbf{K}_t-\mathbf{H}_t)\right)\\
&\leq \log\det\left(\mathbf{I}+\lambda^{-1}\mathbf{H}_t\right)+\langle(\mathbf{I}+\lambda^{-1}\mathbf{H}_t)^{-1},\lambda^{-1}(\mathbf{K}_t-\mathbf{H}_t)\rangle\\
&\leq \log\det\left(\mathbf{I}+\lambda^{-1}\mathbf{H}_t\right)+\left\|(\mathbf{I}+\lambda^{-1}\mathbf{H}_t)^{-1}\right\|_F\left\|\lambda^{-1}(\mathbf{K}_t-\mathbf{H}_t)\right\|_F\\
&\leq \log\det\left(\mathbf{I}+\lambda^{-1}\mathbf{H}_t\right)+t\sqrt{t}\epsilon\\
&\leq \log\det\left(\mathbf{I}+\lambda^{-1}\mathbf{H}_t\right)+1\\
&\leq 2\gamma_t+1,
\end{aligned}$$

where the first equality is from the definition of $\mathbf{K}_t$ in Definition A.1.5, the first inequality is from the convexity of $\log\det(\cdot)$ function, and the second inequality is from the fact that $\langle\mathbf{A},\mathbf{B}\rangle \leq \|\mathbf{A}\|_F\|\mathbf{B}\|_F$. The third inequality is from the fact that $\|\mathbf{A}\|_F \leq \sqrt{t}\|\mathbf{A}\|_2$ if $\mathbf{A}\in\mathbb{R}^{t\times t}$. The fourth inequality utilizes the choice of $\epsilon = T^{-3/2}$ and the last inequality inherits Lemma 3 in Chowdhury and Gopalan, 2017. □

A.2.4 Proof of Lemma A.1.10.1

Proof of Lemma A.1.10.1. By basic linear algebra, we have

$$\begin{aligned}
\det(\mathbf{V}_t) = \det\left(\mathbf{V}_{t-1}+\mathbf{v}_n\mathbf{v}_n^\top\right) &= \det(\mathbf{V}_t)\det\left(\mathbf{I}+\mathbf{V}_t^{-1/2}\mathbf{v}_n(\mathbf{V}_t^{-1/2}\mathbf{v}_n)^\top\right)\\
&= \det(\mathbf{V}_{t-1})(1+\|\mathbf{v}_{t-1}\|^2_{\mathbf{V}_{t-1}^{-1}})\\
&= \det(\lambda\mathbf{I})\prod_{t=1}^{T}(1+\|\mathbf{v}_{t-1}\|^2_{\mathbf{V}_{t-1}^{-1}})
\end{aligned}$$

where we used that all the eigenvalues of a matrix of the form $\mathbf{I}+\mathbf{x}\mathbf{x}^\top$ are 1, except one eigenvalue, which is $1+\|\mathbf{x}\|^2$ and which corresponds to the eigenvector $\mathbf{x}$. Using $\min\{u,B\} \leq \frac{B}{\log(B+1)}\log(1+u), \forall u\in[0,B]$, we get

$$\begin{aligned}
\sum_{t=1}^{T}\min\{B,\|\mathbf{v}_{t-1}\|^2_{\mathbf{V}_{t-1}^{-1}}\} &\leq \frac{B}{\log(B+1)}\sum_{t=1}^{T}\log\left(1+\|\mathbf{v}_{t-1}\|^2_{\mathbf{V}_{t-1}^{-1}}\right)\\
&\leq \frac{B}{\log(B+1)}\log\det\left(\mathbf{I}+\lambda^{-1}\sum_{i=1}^{T}\mathbf{v}_i\mathbf{v}_i^\top\right)
\end{aligned}$$

□

Appendix B

Supplementary Material of Chapter 4

Appendix C

Supplementary Material of Chapter 5

C.1 Proof of Theoretical Analysis in Chapter 5

C.1.1 Proof of Lemma 5.3.4

In this section, we present the detailed proof of Lemma 5.3.4 in the main paper. Before going to the proof, we repeat Lemma 5.3.4 here:

Lemma 5.3.4. *Conditioned on* $\mathcal{D}_t = \{\mathbf{x}_i, y_i\}_{i=1}^t, \mathcal{R} = \{\mathbf{z}_j, u_j\}_{j=1}^{N_r}$, *the acquisition function* $\widetilde{f}_t(\mathbf{x}) = h(\mathbf{x}; \boldsymbol{\theta}_{t-1})$ *can be viewed as a random draw from a* GP $\left(\mu_t^f(\mathbf{x}), \nu_t^2 \left(\sigma_t^f\right)^2(\mathbf{x})\right)$ *with the following mean and covariance functions:*

$$\mu_t^f(\mathbf{x}) = \phi(\mathbf{x})^\top \boldsymbol{\xi}_t^\top \widehat{\mathbf{K}}_{\text{PINN}}^{-1} \begin{bmatrix} \mathbf{Y}_t \\ \mathbf{U}_r \end{bmatrix}$$
$$\left(\sigma_t^f\right)^2(\mathbf{x}) = \langle \phi(\mathbf{x}), \phi(\mathbf{x}) \rangle - \phi(\mathbf{x})^\top \boldsymbol{\xi}_t^\top \widehat{\mathbf{K}}_{\text{PINN}}^{-1} \boldsymbol{\xi}_t \phi(\mathbf{x}),$$

where

$$\widehat{\mathbf{K}}_{\text{PINN}}^{-1} = \begin{bmatrix} \mathbf{K}_{uu} + \lambda_1 \mathbf{I} & \mathbf{K}_{ur} \\ \mathbf{K}_{ru} & \mathbf{K}_{rr} + \lambda_2 \mathbf{I} \end{bmatrix}^{-1} = \begin{bmatrix} \widetilde{\mathbf{A}} & \widetilde{\mathbf{B}} \\ \widetilde{\mathbf{C}} & \widetilde{\mathbf{D}} \end{bmatrix}$$
$$\boldsymbol{\Phi}_t = [\phi(\mathbf{x}_1)^\top, \dots, \phi(\mathbf{x}_t)^\top]^\top$$
$$\boldsymbol{\Omega}_r = [\omega(\mathbf{z}_1)^\top, \dots, \omega(\mathbf{z}_{N_r})^\top]^\top, \ \boldsymbol{\xi}_t = \begin{bmatrix} \boldsymbol{\Phi}_t^\top & \boldsymbol{\Omega}_r^\top \end{bmatrix}^\top$$
$$\mathbf{K}_{uu} = \boldsymbol{\Phi}_t \boldsymbol{\Phi}_t^\top, \ \mathbf{K}_{ur} = \boldsymbol{\Phi}_t \boldsymbol{\Omega}_r^\top, \mathbf{K}_{ru} = \mathbf{K}_{ur}^\top, \mathbf{K}_{rr} = \boldsymbol{\Omega}_r \boldsymbol{\Omega}_r^\top$$
$$\mathbf{Y}_t = [y_1, y_2, \dots, y_t]^\top, \mathbf{U}_r = [u_1, u_2, \dots, u_{N_r}]^\top$$

To prove Lemma 5.3.4, we need the following lemma:

Lemma C.1.1. *(Theorem 4.1 Wang, Yu, and Perdikaris (2022)) A sufficiently wide physics-informed neural network for modeling the problem defined in Section 4 induces a joint multivariate Gaussian distribution between the network outputs and its "derivatives" after applying the differential operator to this network*

$$\begin{bmatrix} \mathbf{h}_{\boldsymbol{\theta}} \\ \mathbf{g}_{\boldsymbol{\theta}} \end{bmatrix} \sim N(\mathbf{0}, \mathbf{K}_{\text{NTK}-\text{PINN}}), \tag{C.1}$$

where $\mathbf{K}_{\text{NTK}-\text{PINN}} = \begin{bmatrix} \mathbf{K}_{uu} & \mathbf{K}_{ur} \\ \mathbf{K}_{ru} & \mathbf{K}_{rr} \end{bmatrix}$ *is the NTK matrix of PINN, with*

$$(\mathbf{K}_{uu})_{ij} = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle \tag{C.2}$$

$$(\mathbf{K}_{ur})_{ij} = \langle \phi(\mathbf{x}_i), \omega(\mathbf{z}_j) \rangle \tag{C.3}$$

$$(\mathbf{K}_{uu})_{ij} = \langle \omega(\mathbf{z}_i), \omega(\mathbf{z}_j) \rangle \tag{C.4}$$

$$\mathbf{K}_{ru} = \mathbf{K}_{ur}^{\top}, \tag{C.5}$$

$$\mathbf{h}_{\boldsymbol{\theta}} = [h(\mathbf{x}_1, \boldsymbol{\theta}), \dots, h(\mathbf{x}_t, \boldsymbol{\theta})]^{\top} \tag{C.6}$$

$$\mathbf{g}_{\boldsymbol{\theta}} = [\mathcal{N}[h](\mathbf{z}_1, \boldsymbol{\theta}), \dots, \mathcal{N}[h](\mathbf{z}_{N_r}, \boldsymbol{\theta})]^{\top} \tag{C.7}$$

where $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{D}_t$ *and* $\mathbf{z}_i, \mathbf{z}_j \in \mathcal{R}$ *are two arbitrary points belonging to the set* $\mathcal{R}$ *defined in Algorithm 1 and* $\mathcal{N}[h]$ *denotes a differential operator of the neural network* $h(\cdot, \boldsymbol{\theta})$ *with respect to the input* $\mathbf{x}$.

Corollary C.1.1.1. The unknown reward function values $f_{1:t}$ and the PDE observations $g_{1:N_r}$ are jointly Gaussian with an initial prior distribution with zero mean and the covariance $\nu_t^2 \mathbf{K}_{\text{NTK}-\text{PINN}}$, where ν_t is the exploration coefficient introduced in Section 4.

$$\begin{bmatrix} f_{1:t} \\ g_{1:N_r} \end{bmatrix} \sim N(\mathbf{0}, \nu_t^2 \mathbf{K}_{\text{NTK}-\text{PINN}}), \tag{C.8}$$

Proof of Corollary C.1.1.1. The algorithm updates the neural network by minimizing the loss function:

$$\mathcal{L}(t) = \sum_{i=1}^{t-1} [y_i - \nu_t h(\mathbf{x}_i; \boldsymbol{\theta}_{t-1})]^2 + \sum_{j=1}^{N_r} [u_j - \nu_t \mathcal{N}[h](\mathbf{z}_j; \boldsymbol{\theta}_{t-1})]^2 \tag{C.9}$$

As the function prediction and its “derivatives" prediction of each observation $\mathbf{x}_i$ and $\mathbf{z}_j$ at iteration t is modeled by $\nu_t h(\mathbf{x}_i; \boldsymbol{\theta}_{t-1})$ and $\nu_t \mathcal{N}[h](\mathbf{z}_j; \boldsymbol{\theta}_{t-1})$, it is clear to see, from Lemma C.1.1, that the function values of the unknown function f can be assumed to follow a joint Gaussian prior with zero means and covariance matrix $\nu_t^2 \mathbf{K}_{\text{NTK}-\text{PINN}}$. A similar argument can be applied to the function g, where $g(\cdot) = \mathcal{N}[f](\cdot)$ with $\mathcal{N}[f]$ is a differential operator. We are now ready to prove 5.3.4. □

Proof of Lemma 5.3.4. From Corollary C.1.1.1, the priors for values of both f and g follow a joint Gaussian distribution with kernel $\mathbf{K}_{\text{NTK}-\text{PINN}}$. Let $\mathbf{K}_{\mathbf{x}}$ be the NTK matrix between point $\mathbf{x}$ and all training data: $\mathbf{K}_x = \begin{bmatrix} \Sigma_a & \Sigma_b \\ \Sigma_c & \Sigma_d \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Phi}_t \phi(\mathbf{x}) & \boldsymbol{\Phi}_t \omega(\mathbf{x}) \\ \boldsymbol{\Omega}_r \phi(\mathbf{x}) & \boldsymbol{\Omega}_r \omega(\mathbf{x}) \end{bmatrix}$. Then the the posterior of f and g evaluated at an input $\mathbf{x}$ will be a Gaussian distribution with mean and variance functions:

Mean function:

$$\begin{aligned}
\begin{bmatrix} \mu_t^f(\mathbf{x}) \\ \mu_t^g(\mathbf{x}) \end{bmatrix} &= \mathbf{K}_{\mathbf{x}}^\top \widehat{\mathbf{K}}_{\text{PINN}}^{-1} \begin{bmatrix} \mathbf{Y}_t \\ \mathbf{U}_r \end{bmatrix} \\
&= \begin{bmatrix} \Sigma_a^\top & \Sigma_c^\top \\ \Sigma_b^\top & \Sigma_d^\top \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{A}} & \widetilde{\mathbf{B}} \\ \widetilde{\mathbf{C}} & \widetilde{\mathbf{D}} \end{bmatrix} \begin{bmatrix} \mathbf{Y}_t \\ \mathbf{U}_r \end{bmatrix} \\
&= \begin{bmatrix} \phi(\mathbf{x})^\top \mathbf{\Phi}_t^\top & \phi(\mathbf{x})^\top \mathbf{\Omega}_r^\top \\ \omega(\mathbf{x})^\top \mathbf{\Phi}_t^\top & \omega(\mathbf{x})^\top \mathbf{\Omega}_r^\top \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{A}} & \widetilde{\mathbf{B}} \\ \widetilde{\mathbf{C}} & \widetilde{\mathbf{D}} \end{bmatrix} \begin{bmatrix} \mathbf{Y}_t \\ \mathbf{U}_r \end{bmatrix} \\
&= \begin{bmatrix} \phi(\mathbf{x})^\top \mathbf{\Phi}_t^\top \widetilde{\mathbf{A}} \mathbf{Y}_t + \phi(\mathbf{x})^\top \mathbf{\Omega}_r^\top \widetilde{\mathbf{C}} \mathbf{Y}_t + \phi(\mathbf{x})^\top \mathbf{\Phi}_t^\top \widetilde{\mathbf{B}} \mathbf{U}_r + \phi(\mathbf{x})^\top \mathbf{\Omega}_r^\top \widetilde{\mathbf{D}} \mathbf{U}_r \\ \omega(\mathbf{x})^\top \mathbf{\Phi}_t^\top \widetilde{\mathbf{A}} \mathbf{Y}_t + \omega(\mathbf{x})^\top \mathbf{\Omega}_r^\top \widetilde{\mathbf{C}} \mathbf{Y}_t + \omega(\mathbf{x})^\top \mathbf{\Phi}_t^\top \widetilde{\mathbf{B}} \mathbf{U}_r + \omega(\mathbf{x})^\top \mathbf{\Omega}_r^\top \widetilde{\mathbf{D}} \mathbf{U}_r \end{bmatrix} \\
&= \phi(\mathbf{x})^\top \boldsymbol{\xi}_t^\top \widehat{\mathbf{K}}_{\text{PINN}}^{-1} \begin{bmatrix} \mathbf{Y}_t \\ \mathbf{U}_r \end{bmatrix}
\end{aligned}$$

Variance function: Let $\mathbf{K}_{\mathbf{xx}} = \begin{bmatrix} \langle \phi(\mathbf{x}), \phi(\mathbf{x}) \rangle & \langle \phi(\mathbf{x}), \omega(\mathbf{x}) \rangle \\ \langle \omega(\mathbf{x}), \phi(\mathbf{x}) \rangle & \langle \omega(\mathbf{x}), \omega(\mathbf{x}) \rangle \end{bmatrix}$, then we have the posterior covariance matrix of f and g at input $\mathbf{x}$ is:

$$\begin{bmatrix} \mathrm{Cov}_f(\mathbf{x}) & \mathrm{Cov}_{fg}(\mathbf{x}) \\ \mathrm{Cov}_{gf}(\mathbf{x}) & \mathrm{Cov}_g(\mathbf{x}) \end{bmatrix} \tag{C.10}$$

$$= \nu_t^2 \left(\mathbf{K}_{\mathbf{xx}} - \mathbf{K}_{\mathbf{x}}{}^\top \widehat{\mathbf{K}}_{\text{PINN}}^{-1} \mathbf{K}_{\mathbf{x}} \right) \tag{C.11}$$

$$= \nu_t^2 \begin{bmatrix} \langle \phi(\mathbf{x}), \phi(\mathbf{x}) \rangle & \langle \phi(\mathbf{x}), \omega(\mathbf{x}) \rangle \\ \langle \omega(\mathbf{x}), \phi(\mathbf{x}) \rangle & \langle \omega(\mathbf{x}), \omega(\mathbf{x}) \rangle \end{bmatrix} - \nu_t^2 \begin{bmatrix} \Sigma_a^\top & \Sigma_c^\top \\ \Sigma_b^\top & \Sigma_d^\top \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{A}} & \widetilde{\mathbf{B}} \\ \widetilde{\mathbf{C}} & \widetilde{\mathbf{D}} \end{bmatrix} \begin{bmatrix} \Sigma_a & \Sigma_b \\ \Sigma_c & \Sigma_d \end{bmatrix} \tag{C.12}$$

Therefore,

$$\mathrm{Cov}_f(\mathbf{x}) = \nu_t^2 \langle \phi(\mathbf{x}), \phi(\mathbf{x}) \rangle - \nu_t^2 (\Sigma_a^\top \widetilde{\mathbf{A}} \Sigma_a + \Sigma_c^\top \widetilde{\mathbf{C}} \Sigma_c + \Sigma_a^\top \widetilde{\mathbf{B}} \Sigma_a + \Sigma_c^\top \widetilde{\mathbf{D}} \Sigma_c) \tag{C.13}$$

$$= \nu_t^2 \langle \phi(\mathbf{x}), \phi(\mathbf{x}) \rangle - \nu_t^2 \begin{bmatrix} \Sigma_a^\top & \Sigma_c^\top \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{A}} & \widetilde{\mathbf{B}} \\ \widetilde{\mathbf{C}} & \widetilde{\mathbf{D}} \end{bmatrix} \begin{bmatrix} \Sigma_a \\ \Sigma_c \end{bmatrix} \tag{C.14}$$

$$= \nu_t^2 \langle \phi(\mathbf{x}), \phi(\mathbf{x}) \rangle - \nu_t^2 \begin{bmatrix} \phi(\mathbf{x})^\top \mathbf{\Phi}_t^\top & \phi(\mathbf{x})^\top \mathbf{\Omega}_r^\top \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{A}} & \widetilde{\mathbf{B}} \\ \widetilde{\mathbf{C}} & \widetilde{\mathbf{D}} \end{bmatrix} \begin{bmatrix} \mathbf{\Phi}_t \phi(\mathbf{x}) \\ \mathbf{\Omega}_r \phi(\mathbf{x}) \end{bmatrix} \tag{C.15}$$

$$= \nu_t^2 \langle \phi(\mathbf{x}), \phi(\mathbf{x}) \rangle - \nu_t^2 \phi(\mathbf{x})^\top \boldsymbol{\xi}_t^\top \widehat{\mathbf{K}}_{\text{PINN}}^{-1} \xi_t \phi(\mathbf{x}) \tag{C.16}$$

$$= \nu_t^2 (\sigma_t^f)^2(\mathbf{x}) \tag{C.17}$$

$\square$

C.1.2 Proof of Lemma C.1.2

Lemma C.1.2. *The interaction information between f and observation $\mathbf{Y}_t$ and PDE data $\mathbf{U}_r$, for the points chosen from Algorithm 1 can be calculated as:*

$$I(f;\mathbf{Y}_t;\mathbf{U}_r) = \frac{1}{2}\log\left(\frac{\det\left(\frac{\mathbf{\Phi}_t^\top\mathbf{\Phi}_t}{\lambda_1}+\mathbf{I}\right)\det\left(\frac{\mathbf{\Omega}_r^\top\mathbf{\Omega}_r}{\lambda_2}+\mathbf{I}\right)}{\det\left(\frac{\mathbf{\Phi}_t^\top\mathbf{\Phi}_t}{\lambda_1}+\frac{\mathbf{\Omega}_r^\top\mathbf{\Omega}_r}{\lambda_2}+\mathbf{I}\right)}\right)$$

To prove Lemma C.1.2, we need to prove the following technical lemma:

Lemma C.1.2.1. Let $\mathbf{u}\in\mathrm{R}^{n\times p}$ and $\mathbf{K}\in\mathrm{R}^{p\times p}$ is a positive semi-definite matrix and $p\geq n$. Then

$$\frac{\det\left[\mathbf{u}\left(\mathbf{K}(\mathbf{u}^\top\mathbf{u}+\mathbf{I})^{-1}+\mathbf{I}\right)^{-1}\mathbf{u}^\top\right]}{\left[\mathbf{u}\left(\mathbf{K}+\mathbf{I}\right)^{-1}\mathbf{u}^\top\right]} = \frac{\det\left(\mathbf{K}(\mathbf{u}^\top\mathbf{u}+\mathbf{I})^{-1}+\mathbf{I}\right)^{-1}}{\det\left(\mathbf{K}+\mathbf{I}\right)^{-1}} \tag{C.18}$$

Proof of Lemma C.1.2.1. We start the proof by gradually calculating the denominator and numerator.

Denominator

$$\begin{aligned}
&\det\left[\mathbf{u}(\mathbf{I}+\mathbf{K})^{-1}\mathbf{u}^\top\right]\\
&=\det\left[\mathbf{u}\left[(\mathbf{I}-\mathbf{K}(\mathbf{I}+\mathbf{K})^{-1}\right]\mathbf{u}^\top\right]\\
&=\det\left[\mathbf{u}\mathbf{u}^\top-\mathbf{u}\mathbf{K}(\mathbf{I}+\mathbf{K})^{-1}\mathbf{u}^\top\right]\\
&=\det\left[(\mathbf{u}\mathbf{u}^\top)\left(\mathbf{K}(\mathbf{I}+\mathbf{K})^{-1}\right)\left((\mathbf{I}+\mathbf{K})\mathbf{K}^{-1}-\mathbf{u}^\top(\mathbf{u}\mathbf{u}^\top)^{-1}\mathbf{u}\right)\right]\\
&=\det\left(\mathbf{u}\mathbf{u}^\top\right)\det\left(\mathbf{K}(\mathbf{I}+\mathbf{K})^{-1}\right)\det\left((\mathbf{I}+\mathbf{K})\mathbf{K}^{-1}-\mathbf{u}^\top(\mathbf{u}\mathbf{u}^\top)^{-1}\mathbf{u}\right)\\
&=\det\left(\mathbf{u}\mathbf{u}^\top\right)\det\left((\mathbf{I}+\mathbf{K})^{-1}\right)\det\mathbf{K}\det\left((\mathbf{I}+\mathbf{K})\mathbf{K}^{-1}-\mathbf{u}^\top(\mathbf{u}\mathbf{u}^\top)^{-1}\mathbf{u}\right)\\
&=\det\left(\mathbf{u}\mathbf{u}^\top\right)\det\left((\mathbf{I}+\mathbf{K})^{-1}\right)\det\left(\mathbf{K}(\mathbf{I}+\mathbf{K})\mathbf{K}^{-1}-\mathbf{K}\mathbf{u}^\top(\mathbf{u}\mathbf{u}^\top)^{-1}\mathbf{u}\right)\\
&=\det\left(\mathbf{u}\mathbf{u}^\top\right)\det\left((\mathbf{I}+\mathbf{K})^{-1}\right)\det\left(\mathbf{I}+\mathbf{K}-\mathbf{K}\mathbf{u}^\top(\mathbf{u}\mathbf{u}^\top)^{-1}\mathbf{u}\right)
\end{aligned}$$

The first equation utilizes the Woodbury matrix inversion formula while the third equation uses generalized matrix determinant lemma [1].

Numerator Let $\widetilde{\mathbf{K}}=\mathbf{K}(\mathbf{u}^\top\mathbf{u}+\mathbf{I})^{-1}$, then we have

$$\det\left[\mathbf{u}\left(\mathbf{K}(\mathbf{u}^\top\mathbf{u}+\mathbf{I})^{-1}+\mathbf{I}\right)^{-1}\mathbf{u}^\top\right] \tag{C.19}$$

$$=\det\left[\mathbf{u}\left(\mathbf{I}+\widetilde{\mathbf{K}}\right)^{-1}\mathbf{u}^\top\right] \tag{C.20}$$

$$=\det\left(\mathbf{u}\mathbf{u}^\top\right)\det\left((\mathbf{I}+\widetilde{\mathbf{K}})^{-1}\right)\det\left(\mathbf{I}+\widetilde{\mathbf{K}}-\widetilde{\mathbf{K}}\mathbf{u}^\top(\mathbf{u}\mathbf{u}^\top)^{-1}\mathbf{u}\right), \tag{C.21}$$

[1]Suppose $\mathbf{A}$ is an invertible n-by-n matrix and $\mathbf{U},\mathbf{V}$ are n-by-m matrices, $m\leq n$. Then $\det\left(\mathbf{A}+\mathbf{U}\mathbf{W}\mathbf{V}^\top\right)=\det(\mathbf{A})\det(\mathbf{W})\det\left(\mathbf{W}^{-1}+\mathbf{V}^\top\mathbf{A}^{-1}\mathbf{U}\right)$.

where we use the result at line (C.19) and replace $\mathbf{K}$ by $\widetilde{\mathbf{K}}$. We also have:

$$\begin{aligned}
&\det\left((\mathbf{I}+\widetilde{\mathbf{K}}) - \widetilde{\mathbf{K}}\mathbf{u}^\top(\mathbf{u}\mathbf{u}^\top)^{-1}\mathbf{u}\right) && \text{(C.22)}\\
&= \det\left(\mathbf{I}+\mathbf{K}(\mathbf{u}^\top\mathbf{u}+\mathbf{I})^{-1} - \mathbf{K}(\mathbf{u}^\top\mathbf{u}+\mathbf{I})^{-1}\mathbf{u}^\top(\mathbf{u}\mathbf{u}^\top)^{-1}\mathbf{u}\right) && \text{(C.23)}\\
&= \det\left[\mathbf{I}+\mathbf{K}\left(\mathbf{I}-\mathbf{u}^\top\mathbf{u}(\mathbf{u}^\top\mathbf{u}+\mathbf{I})^{-1}\right) - \mathbf{K}(\mathbf{u}^\top\mathbf{u}+\mathbf{I})^{-1}\mathbf{u}^\top(\mathbf{u}\mathbf{u}^\top)^{-1}\mathbf{u}\right] && \text{(C.24)}\\
&= \det\left[\mathbf{I}+\mathbf{K}-\mathbf{K}\mathbf{u}^\top\mathbf{u}(\mathbf{u}^\top\mathbf{u}+\mathbf{I})^{-1} - \mathbf{K}(\mathbf{u}^\top\mathbf{u}+\mathbf{I})^{-1}\mathbf{u}^\top(\mathbf{u}\mathbf{u}^\top)^{-1}\mathbf{u}\right] && \text{(C.25)}\\
&= \det\left[\mathbf{I}+\mathbf{K}-\mathbf{K}\mathbf{u}^\top(\mathbf{u}\mathbf{u}^\top+\mathbf{I})^{-1}\mathbf{u} - \mathbf{K}\mathbf{u}^\top(\mathbf{u}\mathbf{u}^\top+\mathbf{I})^{-1}(\mathbf{u}\mathbf{u}^\top)^{-1}\mathbf{u}\right] && \text{(C.26)}\\
&= \det\left[\mathbf{I}+\mathbf{K}-\mathbf{K}\mathbf{u}^\top(\mathbf{u}\mathbf{u}^\top+\mathbf{I})^{-1}\left(\mathbf{I}+(\mathbf{u}\mathbf{u}^\top)^{-1}\right)\mathbf{u}\right] && \text{(C.27)}\\
&= \det\left(\mathbf{I}+\mathbf{K}-\mathbf{K}\mathbf{u}^\top(\mathbf{u}\mathbf{u}^\top)^{-1}\mathbf{u}\right) && \text{(C.28)}
\end{aligned}$$

Therefore, we have the final expression of the numerator

$$\begin{aligned}
&\det\left((\mathbf{I}+\widetilde{\mathbf{K}}) - \widetilde{\mathbf{K}}\mathbf{u}^\top(\mathbf{u}\mathbf{u}^\top)^{-1}\mathbf{u}\right) && \text{(C.29)}\\
&= \det\left(\mathbf{u}\mathbf{u}^\top\right)\det\left((\mathbf{I}+\widetilde{\mathbf{K}})^{-1}\right)\det\left(\mathbf{I}+\mathbf{K}-\mathbf{K}\mathbf{u}^\top(\mathbf{u}\mathbf{u}^\top)^{-1}\mathbf{u}\right) && \text{(C.30)}
\end{aligned}$$

Using the derived numerator and denominator, we have

$$\begin{aligned}
&\frac{\det\left[\mathbf{u}\left(\mathbf{K}(\mathbf{u}^\top\mathbf{u}+\mathbf{I})^{-1}+\mathbf{I}\right)^{-1}\mathbf{u}^\top\right]}{\left[\mathbf{u}\left(\mathbf{K}+\mathbf{I}\right)^{-1}\mathbf{u}^\top\right]} && \text{(C.31)}\\
&= \frac{\det(\mathbf{u}\mathbf{u}^\top)\det\left((\mathbf{I}+\widetilde{\mathbf{K}})^{-1}\right)\det\left(\mathbf{I}+\mathbf{K}-\mathbf{K}\mathbf{u}^\top(\mathbf{u}\mathbf{u}^\top)^{-1}\mathbf{u}\right)}{\det(\mathbf{u}\mathbf{u}^\top)\det\left((\mathbf{I}+\mathbf{K})^{-1}\right)\det\left(\mathbf{I}+\mathbf{K}-\mathbf{K}\mathbf{u}^\top(\mathbf{u}\mathbf{u}^\top)^{-1}\mathbf{u}\right)} && \text{(C.32)}\\
&= \frac{\det\left(\mathbf{I}+\widetilde{\mathbf{K}}\right)^{-1}}{\det(\mathbf{I}+\mathbf{K})^{-1}} && \text{(C.33)}\\
&= \frac{\det\left(\mathbf{K}(\mathbf{u}^\top\mathbf{u}+\mathbf{I})^{-1}+\mathbf{I}\right)^{-1}}{\det\left(\mathbf{K}+\mathbf{I}\right)^{-1}} && \text{(C.34)}
\end{aligned}$$

□

Corollary C.1.2.2. Let $\mathbf{u} \in \mathrm{R}^{n\times p}$ and $\mathbf{K} \in \mathrm{R}^{p\times p}$ is a positive semi-definite matrix and $p \geq n$. Then

$$\frac{\det\left[\mathbf{u}(\mathbf{K}+\mathbf{I})^{-1}\mathbf{u}^\top\right]}{\det[\mathbf{u}(\mathbf{K}+\mathbf{u}^\top\mathbf{u}+\mathbf{I})^{-1}\mathbf{u}^\top]} = \frac{\det(\mathbf{K}+\mathbf{I})^{-1}}{\det(\mathbf{K}+\mathbf{u}^\top\mathbf{u}+\mathbf{I})^{-1}} \tag{C.35}$$

Proof of Corollary C.1.2.2.

$$
\begin{aligned}
&\frac{\det\left[\mathbf{u}(\mathbf{K}+\mathbf{I})^{-1}\mathbf{u}^\top\right]}{\det[\mathbf{u}(\mathbf{K}+\mathbf{u}^\top\mathbf{u}+\mathbf{I})^{-1}\mathbf{u}^\top]} \\
&= \frac{\det\left[\mathbf{u}(\mathbf{K}+\mathbf{I})^{-1}\mathbf{u}^\top\right]}{\det[\mathbf{u}(\mathbf{u}^\top\mathbf{u}+\mathbf{I})^{-1}(\mathbf{K}(\mathbf{u}^\top\mathbf{u}+\mathbf{I})^{-1}+\mathbf{I})^{-1}\mathbf{u}^\top]} && \text{(C.36)} \\
&= \frac{\det\left[\mathbf{u}(\mathbf{K}+\mathbf{I})^{-1}\mathbf{u}^\top\right]}{\det[(\mathbf{u}\mathbf{u}^\top+\mathbf{I})^{-1}\mathbf{u}(\mathbf{K}(\mathbf{u}^\top\mathbf{u}+\mathbf{I})^{-1}+\mathbf{I})^{-1}\mathbf{u}^\top]} && \text{(C.37)} \\
&= \frac{1}{\det(\mathbf{u}\mathbf{u}^\top+\mathbf{I})^{-1}}\frac{\det\left[\mathbf{u}(\mathbf{K}+\mathbf{I})^{-1}\mathbf{u}^\top\right]}{\det[\mathbf{u}(\mathbf{K}(\mathbf{u}^\top\mathbf{u}+\mathbf{I})^{-1}+\mathbf{I})^{-1}\mathbf{u}^\top]} && \text{(C.38)} \\
&= \frac{1}{\det(\mathbf{u}^\top\mathbf{u}+\mathbf{I})^{-1}}\frac{\det\left[(\mathbf{K}+\mathbf{I})^{-1}\right]}{\det[(\mathbf{K}(\mathbf{u}^\top\mathbf{u}+\mathbf{I})^{-1}+\mathbf{I})^{-1}]} && \text{(C.39)} \\
&= \frac{\det(\mathbf{K}+\mathbf{I})^{-1}}{\det(\mathbf{K}+\mathbf{u}^\top\mathbf{u}+\mathbf{I})^{-1}}, && \text{(C.40)}
\end{aligned}
$$

The second equation uses the matrix inversion identity of two non-singular matrices $\mathbf{A}$ and $\mathbf{B}$, i.e., $(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$ while the fourth equation directly utilizes Lemma C.1.2.1. □

Proof of Lemma C.1.2. By definition, we have $I(f;\mathbf{Y}_t;\mathbf{U}_r) = I(f;\mathbf{Y}_t) - I(f;\mathbf{Y}_t|\mathbf{U}_r)$. We start by proof by calculating $I(f;\mathbf{Y}_t|\mathbf{U}_r)$.

By the properties of GPs, given a set of sampling points $\mathcal{D}_t \subset \mathcal{D}$, we have that $f, \mathbf{Y}_t, \mathbf{U}_r$ are jointly Gaussian:

$$
\begin{pmatrix} f \\ \mathbf{Y}_t \\ \mathbf{U}_r \end{pmatrix} \sim \mathcal{N}\left(\mathbf{0}, \nu_t^2\begin{bmatrix} \mathbf{K}_{uu} & \mathbf{K}_{uu} & \mathbf{K}_{ur} \\ \mathbf{K}_{uu} & \mathbf{K}_{uu}+\lambda_1\mathbf{I} & \mathbf{K}_{ur} \\ \mathbf{K}_{ru} & \mathbf{K}_{uu} & \mathbf{K}_{rr}+\lambda_2\mathbf{I} \end{bmatrix}\right) \tag{C.41}
$$

Then, we have

$$
\begin{aligned}
\operatorname{Cov}(f|\mathbf{U}_r) &= \nu_t^2\left[\mathbf{K}_{uu} - \mathbf{K}_{ur}(\mathbf{K}_{rr}+\lambda_2\mathbf{I})^{-1}\mathbf{K}_{ru}\right] && \text{(C.42)} \\
&= \nu_t^2\left[\mathbf{\Phi}_t\mathbf{\Phi}_t^\top - \mathbf{\Phi}_t\mathbf{\Omega}_r^\top(\mathbf{\Omega}_r\mathbf{\Omega}_r^\top+\lambda_2\mathbf{I})^{-1}\mathbf{\Omega}_r\mathbf{\Phi}_t^\top\right] && \text{(C.43)} \\
&= \nu_t^2\left[\mathbf{\Phi}_t\mathbf{\Phi}_t^\top - \mathbf{\Phi}_t\left[\mathbf{I} - \lambda_2(\mathbf{\Omega}_r\mathbf{\Omega}_r^\top+\lambda_2\mathbf{I})^{-1}\right]\mathbf{\Phi}_t^\top\right] && \text{(C.44)} \\
&= \nu_t^2\lambda_2\mathbf{\Phi}_t(\mathbf{\Omega}_r^\top\mathbf{\Omega}_r+\lambda_2\mathbf{I})^{-1}\mathbf{\Phi}_t^\top && \text{(C.45)} \\
&= \nu_t^2\mathbf{\Phi}_t\left(\frac{\mathbf{\Omega}_r^\top\mathbf{\Omega}_r}{\lambda_2}+\mathbf{I}\right)^{-1}\mathbf{\Phi}_t^\top, && \text{(C.46)}
\end{aligned}
$$

and

$$
\operatorname{Cov}(f|\mathbf{Y}_t;\mathbf{U}_r) = \nu_t^2\left(\mathbf{K}_{uu} - \begin{bmatrix}\mathbf{K}_{uu} & \mathbf{K}_{ur}\end{bmatrix}\begin{bmatrix}\mathbf{K}_{uu}+\lambda_1\mathbf{I} & \mathbf{K}_{ur} \\ \mathbf{K}_{ru} & \mathbf{K}_{rr}+\lambda_2\mathbf{I}\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{K}_{uu} \\ \mathbf{K}_{ru}\end{bmatrix}\right) \tag{C.47}
$$

$$
= \nu_t^2(\mathbf{\Phi}_t\mathbf{\Phi}_t^\top - \mathbf{\Phi}_t\xi_t^\top\widehat{\mathbf{K}}_{\mathrm{PINN}}^{-1}\xi_t\mathbf{\Phi}_t^\top) \tag{C.48}
$$

Let $\mathbf{V} = \boldsymbol{\xi}_t^\top \widehat{\mathbf{K}}_{\text{PINN}}^{-1} \boldsymbol{\xi}_t$, now we need to calculate $\mathbf{V}$. We have

$$
\begin{aligned}
\mathbf{V} &= \boldsymbol{\xi_t}^\top \widehat{\mathbf{K}}_{\text{PINN}}^{-1} \boldsymbol{\xi_t} && \text{(C.49)}\\
&= \boldsymbol{\Phi}_t^\top \widetilde{\mathbf{A}} \boldsymbol{\Phi}_t + \boldsymbol{\Omega}_r^\top \widetilde{\mathbf{C}} \boldsymbol{\Phi}_t + \boldsymbol{\Phi}_t^\top \widetilde{\mathbf{B}} \boldsymbol{\Omega}_r + \boldsymbol{\Omega}_r^\top \widetilde{\mathbf{D}} \boldsymbol{\Omega}_r && \text{(C.50)}\\
&= \boldsymbol{\Phi}_t^\top (\mathbf{P}^{-1} - \mathbf{P}^{-1}\mathbf{Q}\widetilde{\mathbf{C}}) \boldsymbol{\Phi}_t + \boldsymbol{\Omega}_r^\top \widetilde{\mathbf{C}} \boldsymbol{\Phi}_t - \boldsymbol{\Phi}_t^\top \mathbf{P}^{-1}\mathbf{Q} \boldsymbol{\Omega}_r + \boldsymbol{\Omega}_r^\top \widetilde{\mathbf{D}} \boldsymbol{\Omega}_r && \text{(C.51)}\\
&= \boldsymbol{\Phi}_t^\top \mathbf{P}^{-1} \boldsymbol{\Phi}_t - \boldsymbol{\Phi}_t^\top \mathbf{P}^{-1}\mathbf{Q}\widetilde{\mathbf{C}} \boldsymbol{\Phi}_t + \boldsymbol{\Omega}_r^\top \widetilde{\mathbf{C}} \boldsymbol{\Phi}_t - \boldsymbol{\Phi}_t^\top \mathbf{P}^{-1}\mathbf{Q} \boldsymbol{\Omega}_r + \boldsymbol{\Omega}_r^\top \widetilde{\mathbf{D}} \boldsymbol{\Omega}_r && \text{(C.52)}\\
&= \boldsymbol{\Phi}_t^\top \mathbf{P}^{-1} \boldsymbol{\Phi}_t + (\underbrace{\boldsymbol{\Omega}_r - \boldsymbol{\Phi}_t^\top \mathbf{P}^{-1}\mathbf{Q}}_{U_1})(\widetilde{\mathbf{C}} \boldsymbol{\Phi}_t + \widetilde{\mathbf{D}} \boldsymbol{\Omega}_r) && \text{(C.53)}\\
&= \boldsymbol{\Phi}_t^\top \mathbf{P}^{-1} \boldsymbol{\Phi}_t + (\underbrace{\boldsymbol{\Omega}_r - \boldsymbol{\Phi}_t^\top \mathbf{P}^{-1}\mathbf{Q}}_{V_1})(\underbrace{\widetilde{\mathbf{C}} \boldsymbol{\Phi}_t + \widetilde{\mathbf{D}} \boldsymbol{\Omega}_r}_{V_2}) && \text{(C.54)}\\
&= \boldsymbol{\Phi}_t^\top (\boldsymbol{\Phi}_t \boldsymbol{\Phi}_t^\top + \lambda_1 \mathbf{I})^{-1} \boldsymbol{\Phi}_t + V_1 V_2 && \text{(C.55)}\\
&= (\boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t + \lambda_1 \mathbf{I})^{-1} \boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t + V_1 V_2 && \text{(C.56)}
\end{aligned}
$$

where

$$
\begin{bmatrix} \mathbf{P} & \mathbf{Q} \\ \mathbf{R} & \mathbf{S} \end{bmatrix} = \begin{bmatrix} \mathbf{K}_{uu} + \lambda_1 \mathbf{I} & \mathbf{K}_{ur} \\ \mathbf{K}_{ru} & \mathbf{K}_{rr} + \lambda_2 \mathbf{I} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Phi}_t \boldsymbol{\Phi}_t^\top + \lambda_1 \mathbf{I} & \boldsymbol{\Phi}_t \boldsymbol{\Omega}_r^\top \\ \boldsymbol{\Omega}_r \boldsymbol{\Phi}_t^\top & \boldsymbol{\Omega}_r \boldsymbol{\Omega}_r^\top + \lambda_2 \mathbf{I} \end{bmatrix} \tag{C.57}
$$

$$
\text{and} \begin{bmatrix} \widetilde{\mathbf{A}} & \widetilde{\mathbf{B}} \\ \widetilde{\mathbf{C}} & \widetilde{\mathbf{D}} \end{bmatrix} = \begin{bmatrix} \mathbf{P} & \mathbf{Q} \\ \mathbf{R} & \mathbf{S} \end{bmatrix}^{-1} \tag{C.58}
$$

The second equality applied the formula of block matrix inversion [2], while the last equality used push-through identity. Next, we have

$$\mathbf{M} = (\mathbf{S} - \mathbf{R}\mathbf{P}^{-1}\mathbf{Q})^{-1} \tag{C.59}$$

$$= \left[\boldsymbol{\Omega}_r\boldsymbol{\Omega}_r^\top + \lambda_2\mathbf{I} - \boldsymbol{\Omega}_r\boldsymbol{\Phi}_t^\top(\boldsymbol{\Phi}_t\boldsymbol{\Phi}_t^\top + \lambda_1\mathbf{I})^{-1}\boldsymbol{\Phi}_t\boldsymbol{\Omega}_r^\top)\right]^{-1} \tag{C.60}$$

$$= \left[\boldsymbol{\Omega}_r\boldsymbol{\Omega}_r^\top + \lambda_2\mathbf{I} - \boldsymbol{\Omega}_r(\boldsymbol{\Phi}_t^\top\boldsymbol{\Phi}_t + \lambda_1\mathbf{I})^{-1}\boldsymbol{\Phi}_t^\top\boldsymbol{\Phi}_t\right]\boldsymbol{\Omega}_r^\top)^{-1} \tag{C.61}$$

$$= \left[\boldsymbol{\Omega}_r\boldsymbol{\Omega}_r^\top + \lambda_2\mathbf{I} - \boldsymbol{\Omega}_r\left[\mathbf{I} - \lambda_1(\boldsymbol{\Phi}_t^\top\boldsymbol{\Phi}_t + \lambda_1\mathbf{I})^{-1}\right]\boldsymbol{\Omega}_r^\top)\right]^{-1} \tag{C.62}$$

$$= \left[\lambda_2\mathbf{I} + \lambda_1\boldsymbol{\Omega}_r(\boldsymbol{\Phi}_t^\top\boldsymbol{\Phi}_t + \lambda_1\mathbf{I})^{-1}\boldsymbol{\Omega}_r^\top\right]^{-1} \tag{C.63}$$

$$= \lambda_1^{-1}\left[\boldsymbol{\Omega}_r(\boldsymbol{\Phi}_t^\top\boldsymbol{\Phi}_t + \lambda_1\mathbf{I})^{-1}\boldsymbol{\Omega}_r^\top + \frac{\lambda_2}{\lambda_1}\mathbf{I}\right]^{-1} \tag{C.64}$$

$$V_1 = \boldsymbol{\Omega}_r^\top - \boldsymbol{\Phi}_t^\top\mathbf{P}^{-1}\mathbf{Q} \tag{C.65}$$

$$= \boldsymbol{\Omega}_r^\top - \boldsymbol{\Phi}_t^\top(\boldsymbol{\Phi}_t\boldsymbol{\Phi}_t^\top + \lambda_1\mathbf{I})^{-1}\boldsymbol{\Phi}_t\boldsymbol{\Omega}_r^\top \tag{C.66}$$

$$= \left[\mathbf{I} - \boldsymbol{\Phi}_t^\top(\boldsymbol{\Phi}_t\boldsymbol{\Phi}_t^\top + \lambda_1\mathbf{I})^{-1}\boldsymbol{\Phi}_t\right]\boldsymbol{\Omega}_r^\top \tag{C.67}$$

$$= \left[\mathbf{I} - (\boldsymbol{\Phi}_t^\top\boldsymbol{\Phi}_t + \lambda_1\mathbf{I})^{-1}\boldsymbol{\Phi}_t^\top\boldsymbol{\Phi}_t\right]\boldsymbol{\Omega}_r^\top \tag{C.68}$$

$$= \left[\mathbf{I} - (\boldsymbol{\Phi}_t^\top\boldsymbol{\Phi}_t + \lambda_1\mathbf{I})^{-1}(\boldsymbol{\Phi}_t^\top\boldsymbol{\Phi}_t + \lambda_1\mathbf{I} - \lambda_1\mathbf{I})\right]\boldsymbol{\Omega}_r^\top \tag{C.69}$$

$$= \lambda_1(\boldsymbol{\Phi}_t^\top\boldsymbol{\Phi}_t + \lambda_1\mathbf{I})^{-1}\boldsymbol{\Omega}_r^\top \tag{C.70}$$

$$V_2 = \widetilde{\mathbf{C}}\boldsymbol{\Phi}_t + \widetilde{\mathbf{D}}\boldsymbol{\Omega}_r \tag{C.71}$$

$$= -\mathbf{M}\mathbf{R}\mathbf{P}^{-1} + \mathbf{M}\boldsymbol{\Omega}_r \tag{C.72}$$

$$= \mathbf{M}(\boldsymbol{\Omega}_r - \mathbf{R}\mathbf{P}^{-1}\boldsymbol{\Phi}_t) \tag{C.73}$$

$$= \mathbf{M}\left[\boldsymbol{\Omega}_r - \boldsymbol{\Omega}_r\boldsymbol{\Phi}_t^\top(\boldsymbol{\Phi}_t\boldsymbol{\Phi}_t^\top + \lambda_1\mathbf{I})^{-1}\boldsymbol{\Phi}_t\right] \tag{C.74}$$

$$= \mathbf{M}\boldsymbol{\Omega}_r\left[\mathbf{I} - \boldsymbol{\Phi}_t^\top(\boldsymbol{\Phi}_t\boldsymbol{\Phi}_t^\top + \lambda_1\mathbf{I})^{-1}\boldsymbol{\Phi}_t\right] \tag{C.75}$$

$$= \lambda_1\mathbf{M}\boldsymbol{\Omega}_r(\boldsymbol{\Phi}_t^\top\boldsymbol{\Phi}_t + \lambda_1\mathbf{I})^{-1} \tag{C.76}$$

$$= \left[\boldsymbol{\Omega}_r(\boldsymbol{\Phi}_t^\top\boldsymbol{\Phi}_t + \lambda_1\mathbf{I})^{-1}\boldsymbol{\Omega}_r^\top + \frac{\lambda_2}{\lambda_1}\mathbf{I}\right]^{-1}\boldsymbol{\Omega}_r(\boldsymbol{\Phi}_t^\top\boldsymbol{\Phi}_t + \lambda_1\mathbf{I})^{-1} \tag{C.77}$$

[2]The inversion of matrix $\mathbf{K} = \begin{bmatrix}\mathbf{P} & \mathbf{Q}\\ \mathbf{R} & \mathbf{S}\end{bmatrix}$ is given as $\mathbf{K}^{-1} = \begin{bmatrix}\mathbf{P}^{-1} + \mathbf{P}^{-1}\mathbf{Q}\mathbf{M}\mathbf{R}\mathbf{P}^{-1} & \mathbf{P}^{-1}\mathbf{Q}\mathbf{M}\\ -\mathbf{M}\mathbf{R}\mathbf{P}^{-1} & \mathbf{M}\end{bmatrix}$ with $\mathbf{M} = (\mathbf{S} - \mathbf{R}\mathbf{P}^{-1}\mathbf{Q})^{-1}$.

Then we have,

$$\begin{aligned}
\mathbf{V} &= \boldsymbol{\xi_t}^\top \widehat{\mathbf{K}}_{\text{PINN}}^{-1} \boldsymbol{\xi_t} && \text{(C.78)}\\
&= (\boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t + \lambda_1 \mathbf{I})^{-1} \boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t + V_1 V_2 && \text{(C.79)}\\
&= (\boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t + \lambda_1 \mathbf{I})^{-1} \boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t + \lambda_1 (\boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t + \lambda_1 \mathbf{I})^{-1} \boldsymbol{\Omega}_r^\top \\
&\quad \cdot \left[\boldsymbol{\Omega}_r (\boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t + \lambda_1 \mathbf{I})^{-1} \boldsymbol{\Omega}_r^\top + \frac{\lambda_2}{\lambda_1} \mathbf{I} \right]^{-1} \boldsymbol{\Omega}_r (\boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t + \lambda_1 \mathbf{I})^{-1} && \text{(C.80)}\\
&= \mathbf{I} - \lambda_1 (\boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t + \lambda_1 \mathbf{I})^{-1} + \lambda_1 (\boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t + \lambda_1 \mathbf{I})^{-1} \boldsymbol{\Omega}_r^\top \\
&\quad \cdot \left[\boldsymbol{\Omega}_r (\boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t + \lambda_1 \mathbf{I})^{-1} \boldsymbol{\Omega}_r^\top + \frac{\lambda_2}{\lambda_1} \mathbf{I} \right]^{-1} \boldsymbol{\Omega}_r (\boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t + \lambda_1 \mathbf{I})^{-1} && \text{(C.81)}\\
&= \mathbf{I} - \lambda_1 (\boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t + \lambda_1 \mathbf{I})^{-1} && \text{(C.82)}\\
&\quad \cdot \left[\mathbf{I} - \boldsymbol{\Omega}_r^\top \left(\boldsymbol{\Omega}_r (\boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t + \lambda_1 \mathbf{I})^{-1} \boldsymbol{\Omega}_r^\top + \frac{\lambda_2}{\lambda_1} \mathbf{I} \right)^{-1} \boldsymbol{\Omega}_r (\boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t + \lambda_1 \mathbf{I})^{-1} \right] && \text{(C.83)}\\
&= \mathbf{I} - \lambda_1 (\boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t + \lambda_1 \mathbf{I})^{-1} && \text{(C.84)}\\
&\quad \cdot \left[\mathbf{I} - \left(\boldsymbol{\Omega}_r^\top \boldsymbol{\Omega}_r (\boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t + \lambda_1 \mathbf{I})^{-1} + \frac{\lambda_2}{\lambda_1} \mathbf{I} \right)^{-1} \boldsymbol{\Omega}_r^\top \boldsymbol{\Omega}_r (\boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t + \lambda_1 \mathbf{I})^{-1} \right] && \text{(C.85)}\\
&= \mathbf{I} - \lambda_1 (\boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t + \lambda_1 \mathbf{I})^{-1} && \text{(C.86)}\\
&\quad \cdot \left[\mathbf{I} - \mathbf{I} + \frac{\lambda_2}{\lambda_1} \left(\boldsymbol{\Omega}_r^\top \boldsymbol{\Omega}_r (\boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t + \lambda_1 \mathbf{I})^{-1} + \frac{\lambda_2}{\lambda_1} \mathbf{I} \right)^{-1} \right] && \text{(C.87)}\\
&= \mathbf{I} - \lambda_2 (\boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t + \lambda_1 \mathbf{I})^{-1} \left(\boldsymbol{\Omega}_r^\top \boldsymbol{\Omega}_r (\boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t + \lambda_1 \mathbf{I})^{-1} + \frac{\lambda_2}{\lambda_1} \mathbf{I} \right)^{-1} && \text{(C.88)}\\
&= \mathbf{I} - \lambda_2 \left(\boldsymbol{\Omega}_r^\top \boldsymbol{\Omega}_r + \frac{\lambda_2}{\lambda_1} (\boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t + \lambda_1 \mathbf{I}) \right)^{-1} && \text{(C.89)}\\
&= \mathbf{I} - \lambda_2 \left(\boldsymbol{\Omega}_r^\top \boldsymbol{\Omega}_r + \frac{\lambda_2}{\lambda_1} \boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t + \lambda_2 \mathbf{I} \right)^{-1} && \text{(C.90)}
\end{aligned}$$

In conclusion, we have

$$\begin{aligned}
\boldsymbol{\xi}_t^\top \widehat{\mathbf{K}}_{\text{PINN}}^{-1} \boldsymbol{\xi}_t &= \mathbf{I} - \lambda_2 \left(\boldsymbol{\Omega}_r^\top \boldsymbol{\Omega}_r + \frac{\lambda_2}{\lambda_1} \boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t + \lambda_2 \mathbf{I} \right)^{-1} && \text{(C.91)}\\
&= \mathbf{I} - \left(\frac{\boldsymbol{\Omega}_r^\top \boldsymbol{\Omega}_r}{\lambda_2} + \frac{\boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t}{\lambda_1} + \mathbf{I} \right)^{-1} && \text{(C.92)}
\end{aligned}$$

Replace Eqn. C.91 to the expression of $\text{Cov}(f_t|\mathbf{Y}_t;\mathbf{U}_r)$ in Eqn. C.47, we have

$$\text{Cov}(f|\mathbf{Y}_t;\mathbf{U}_r) = \nu_t^2 \boldsymbol{\Phi}_t \left(\frac{\boldsymbol{\Omega}_r^\top \boldsymbol{\Omega}_r}{\lambda_2} + \frac{\boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t}{\lambda_1} + \mathbf{I} \right)^{-1} \boldsymbol{\Phi}_t^\top \qquad \text{(C.93)}$$

Combining the expression of $\mathrm{Cov}(f|\mathbf{U}_r)$ in Eqn. C.46 with Eqn. C.93, with H is the entropy function, we have

$$\begin{aligned}
& I(f;\mathbf{Y}_t|\mathbf{U}_r) \\
&= H(f|\mathbf{U}_r) - H(f|\mathbf{Y}_t;\mathbf{U}_r) && \text{(C.94)} \\
&= \frac{1}{2}\log\det(\mathrm{Cov}(f|\mathbf{U}_r)) - \frac{1}{2}\log\det(\mathrm{Cov}(f|\mathbf{Y}_t;\mathbf{U}_r)) && \text{(C.95)} \\
&= \frac{1}{2}\log\det\left(\nu_t^2\mathbf{\Phi}_t\left(\frac{\mathbf{\Omega}_r^\top\mathbf{\Omega}_r}{\lambda_2}+\mathbf{I}\right)^{-1}\mathbf{\Phi}_t^\top\right) \\
&- \frac{1}{2}\log\det\left(\nu_t^2\mathbf{\Phi}_t\left(\frac{\mathbf{\Omega}_r^\top\mathbf{\Omega}_r}{\lambda_2}+\frac{\mathbf{\Phi}_t^\top\mathbf{\Phi}_t}{\lambda_1}+\mathbf{I}\right)^{-1}\mathbf{\Phi}_t^\top\right) && \text{(C.96)} \\
&= \frac{1}{2}\log\frac{\det\left(\mathbf{\Phi}_t\left(\frac{\mathbf{\Omega}_r^\top\mathbf{\Omega}_r}{\lambda_2}+\mathbf{I}\right)^{-1}\mathbf{\Phi}_t^\top\right)}{\det\left(\mathbf{\Phi}_t\left(\frac{\mathbf{\Omega}_r^\top\mathbf{\Omega}_r}{\lambda_2}+\frac{\mathbf{\Phi}_t^\top\mathbf{\Phi}_t}{\lambda_1}+\mathbf{I}\right)^{-1}\mathbf{\Phi}_t^\top\right)} && \text{(C.97)}
\end{aligned}$$

Then we have

$$
\begin{aligned}
& I(f;\mathbf{Y}_t;\mathbf{U}_r) \\
&= I(f;\mathbf{Y}_t) - I(f;\mathbf{Y}_t|\mathbf{U}_r) && \text{(C.98)} \\
&= \frac{1}{2}\log\det\left(\frac{\mathbf{\Phi}_t\mathbf{\Phi}_t^\top}{\lambda_1}+\mathbf{I}\right) - \frac{1}{2}\log\frac{\det\left(\mathbf{\Phi}_t\left(\frac{\mathbf{\Omega}_r^\top\mathbf{\Omega}_r}{\lambda_2}+\mathbf{I}\right)^{-1}\mathbf{\Phi}_t^\top\right)}{\det\left(\mathbf{\Phi}_t\left(\frac{\mathbf{\Omega}_r^\top\mathbf{\Omega}_r}{\lambda_2}+\frac{\mathbf{\Phi}_t^\top\mathbf{\Phi}_t}{\lambda_1}+\mathbf{I}\right)^{-1}\mathbf{\Phi}_t^\top\right)} && \text{(C.99)} \\
&= \frac{1}{2}\log\frac{\det\left(\frac{\mathbf{\Phi}_t\mathbf{\Phi}_t^\top}{\lambda_1}+\mathbf{I}\right)\det\left(\mathbf{\Phi}_t\left(\frac{\mathbf{\Omega}_r^\top\mathbf{\Omega}_r}{\lambda_2}+\frac{\mathbf{\Phi}_t^\top\mathbf{\Phi}_t}{\lambda_1}+\mathbf{I}\right)^{-1}\mathbf{\Phi}_t^\top\right)}{\det\left(\mathbf{\Phi}_t\left(\frac{\mathbf{\Omega}_r^\top\mathbf{\Omega}_r}{\lambda_2}+\mathbf{I}\right)^{-1}\mathbf{\Phi}_t^\top\right)} && \text{(C.100)} \\
&= \frac{1}{2}\log\frac{\det\left[\left(\frac{\mathbf{\Phi}_t\mathbf{\Phi}_t^\top}{\lambda_1}+\mathbf{I}\right)\mathbf{\Phi}_t\left(\frac{\mathbf{\Omega}_r^\top\mathbf{\Omega}_r}{\lambda_2}+\frac{\mathbf{\Phi}_t^\top\mathbf{\Phi}_t}{\lambda_1}+\mathbf{I}\right)^{-1}\mathbf{\Phi}_t^\top\right]}{\det\left[\mathbf{\Phi}_t\left(\frac{\mathbf{\Omega}_r^\top\mathbf{\Omega}_r}{\lambda_2}+\mathbf{I}\right)^{-1}\mathbf{\Phi}_t^\top\right]} && \text{(C.101)} \\
&= \frac{1}{2}\log\frac{\det\left[\mathbf{\Phi}_t\left(\frac{\mathbf{\Phi}_t^\top\mathbf{\Phi}_t}{\lambda_1}+\mathbf{I}\right)\left(\frac{\mathbf{\Omega}_r^\top\mathbf{\Omega}_r}{\lambda_2}+\frac{\mathbf{\Phi}_t^\top\mathbf{\Phi}_t}{\lambda_1}+\mathbf{I}\right)^{-1}\mathbf{\Phi}_t^\top\right]}{\det\left[\mathbf{\Phi}_t\left(\frac{\mathbf{\Omega}_r^\top\mathbf{\Omega}_r}{\lambda_2}+\mathbf{I}\right)^{-1}\mathbf{\Phi}_t^\top\right]} && \text{(C.102)} \\
&= \frac{1}{2}\log\frac{\det\left[\mathbf{\Phi}_t\left(\frac{\mathbf{\Omega}_r^\top\mathbf{\Omega}_r}{\lambda_2}\left(\frac{\mathbf{\Phi}_t^\top\mathbf{\Phi}_t}{\lambda_1}+\mathbf{I}\right)^{-1}+\mathbf{I}\right)^{-1}\mathbf{\Phi}_t^\top\right]}{\det\left[\mathbf{\Phi}_t\left(\frac{\mathbf{\Omega}_r^\top\mathbf{\Omega}_r}{\lambda_2}+\mathbf{I}\right)^{-1}\mathbf{\Phi}_t^\top\right]} && \text{(C.103)} \\
&= \frac{1}{2}\log\frac{\det\left(\frac{\mathbf{\Omega}_r^\top\mathbf{\Omega}_r}{\lambda_2}\left(\frac{\mathbf{\Phi}_t^\top\mathbf{\Phi}_t}{\lambda_1}+\mathbf{I}\right)^{-1}+\mathbf{I}\right)^{-1}}{\det\left(\frac{\mathbf{\Omega}_r^\top\mathbf{\Omega}_r}{\lambda_2}+\mathbf{I}\right)^{-1}} && \text{(C.104)} \\
&= \frac{1}{2}\log\frac{\det\left(\frac{\mathbf{\Omega}_r^\top\mathbf{\Omega}_r}{\lambda_2}+\mathbf{I}\right)}{\det\left(\frac{\mathbf{\Omega}_r^\top\mathbf{\Omega}_r}{\lambda_2}\left(\frac{\mathbf{\Phi}_t^\top\mathbf{\Phi}_t}{\lambda_1}+\mathbf{I}\right)^{-1}+\mathbf{I}\right)} && \text{(C.105)} \\
&= \frac{1}{2}\log\left(\frac{\det\left(\frac{\mathbf{\Phi}_t^\top\mathbf{\Phi}_t}{\lambda_1}+\mathbf{I}\right)\det\left(\frac{\mathbf{\Omega}_r^\top\mathbf{\Omega}_r}{\lambda_2}+\mathbf{I}\right)}{\det\left(\frac{\mathbf{\Phi}_t^\top\mathbf{\Phi}_t}{\lambda_1}+\frac{\mathbf{\Omega}_r^\top\mathbf{\Omega}_r}{\lambda_2}+\mathbf{I}\right)}\right), && \text{(C.106)}
\end{aligned}
$$

where Eqn C.104 is resulted from technical Lemma C.1.2.1. □

Bibliography

Allen-Zhu, Zeyuan, Yuanzhi Li, and Zhao Song (2019). “A convergence theory for deep learning via over-parameterization”. In: *International Conference on Machine Learning*. PMLR, pp. 242–252.

Antonini, Rita Giuliano, Yuriy Kozachenko, and Andrei Volodin (2008). “Convergence of series of dependent φ-subGaussian random variables”. In: *Journal of mathematical analysis and applications* 338.2, pp. 1188–1203.

Ariafar, Setareh et al. (2019). “ADMMBO: Bayesian optimization with unknown constraints using ADMM”. In: *Journal of Machine Learning Research* 20.123, pp. 1–26.

Arora, Sanjeev et al. (2019). “Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks”. In: *International Conference on Machine Learning*. PMLR, pp. 322–332.

Bergstra, James and Yoshua Bengio (2012). “Random search for hyper-parameter optimization.” In: *Journal of machine learning research* 13.2.

Berkenkamp, Felix, Andreas Krause, and Angela P Schoellig (2023). “Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics”. In: *Machine Learning* 112.10, pp. 3713–3747.

Cao, Yuan and Quanquan Gu (2019). “Generalization bounds of stochastic gradient descent for wide and deep neural networks”. In: *Advances in neural information processing systems* 32.

— (2020). “Generalization error bounds of gradient descent for learning over-parameterized deep relu networks”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34, pp. 3349–3356.

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

Cover, Thomas M (1999). *Elements of information theory*. John Wiley & Sons.

Du, Simon S et al. (2018). “Gradient Descent Provably Optimizes Over-parameterized Neural Networks”. In: *International Conference on Learning Representations*.

Gardner, Jacob R et al. (2014). “Bayesian optimization with inequality constraints.” In: *ICML*. Vol. 2014, pp. 937–945.

Gelbart, Michael A, Jasper Snoek, and Ryan P Adams (2014). “Bayesian optimization with unknown constraints”. In: *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, pp. 250–259.

Gonzalez, Javier et al. (2015). “Bayesian optimization for synthetic gene design”. In: *arXiv preprint arXiv:1505.01627*.

Gramacy, Robert B et al. (2016). “Modeling an augmented Lagrangian for blackbox constrained optimization”. In: *Technometrics* 58.1, pp. 1–11.

Greenhill, Stewart et al. (2020). “Bayesian optimization for adaptive experimental design: A review”. In: *IEEE access* 8, pp. 13937–13948.

Hansen, Nikolaus et al. (2021). “COCO: A platform for comparing continuous optimizers in a black-box setting”. In: *Optimization Methods and Software* 36.1, pp. 114–144.

He, Kaiming et al. (2015). “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. In: *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034.

He, Zecheng, Tianwei Zhang, and Ruby B Lee (2018). “Verideep: Verifying integrity of deep neural networks through sensitive-sample fingerprinting”. In: *arXiv preprint arXiv:1808.03277*.

Hernández-Lobato, José Miguel et al. (2015). “Predictive entropy search for bayesian optimization with unknown constraints”. In: *International conference on machine learning*. PMLR, pp. 1699–1707.

Hoffman, Matthew W, Bobak Shahriari, and Nando de Freitas (2013). “Exploiting correlation and budget constraints in bayesian multi-armed bandit optimization”. In: *arXiv preprint arXiv:1303.6746*.

Hutter, Frank, Holger H Hoos, and Kevin Leyton-Brown (2011). “Sequential model-based optimization for general algorithm configuration”. In: *International conference on learning and intelligent optimization*. Springer, pp. 507–523.

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

Ju, Shenghong et al. (2017). “Designing nanostructures for phonon transport via Bayesian optimization”. In: *Physical Review X* 7.2, p. 021024.

Kassraie, Parnian and Andreas Krause (2022). “Neural contextual bandits without regret”. In: *Artificial Intelligence and Statistics*. PMLR, pp. 240–278.

Kikuchi, Shun et al. (2018). “Bayesian optimization for efficient determination of metal oxide grain boundary structures”. In: *Physica B: Condensed Matter* 532, pp. 24–28.

Kiritchenko, Svetlana, Stan Matwin, A Fazel Famili, et al. (2005). “Functional annotation of genes using hierarchical text categorization”. In: *Proc. of the ACL Workshop on Linking Biological Literature, Ontologies and Databases: Mining Biological Semantics*.

Kumar, Abhishek et al. (2020). “A test-suite of non-convex constrained optimization problems from the real-world and some baseline results”. In: *Swarm and Evolutionary Computation* 56, p. 100693.

LeCun, Yann and Corinna Cortes (2010). “MNIST handwritten digit database”. In: URL: `http://yann.lecun.com/exdb/mnist/`.

Letham, Benjamin et al. (2019). “Constrained Bayesian Optimization with Noisy Experiments”. In: *Bayesian Analysis* 14.2, pp. 495–519.

Li, g et al. (2017). “Rapid Bayesian optimisation for synthesis of short polymer fiber materials”. In: *Scientific reports* 7.1, pp. 1–10.

Lu, Congwen and Joel A Paulson (2022). “No-regret Bayesian optimization with unknown equality and inequality constraints using exact penalty functions”. In: *IFAC-PapersOnLine* 55.7, pp. 895–902.

Mockus, Jonas, Vytautas Tiesis, and Antanas Zilinskas (1978). “The application of Bayesian methods for seeking the extremum”. In: *Towards global optimization* 2.117-129, p. 2.

Nguyen, Quoc Phong et al. (2023). “Optimistic Bayesian Optimization with Unknown Constraints”. In: *The Twelfth International Conference on Learning Representations*.

Nour, Majid, Zafer Cömert, and Kemal Polat (2020). “A novel medical diagnosis model for COVID-19 infection detection based on deep features and Bayesian optimization”. In: *Applied Soft Computing* 97, p. 106580.

Osborne, Michael A, Roman Garnett, and Stephen J Roberts (2009). "Gaussian processes for global optimization". In: *3rd international conference on learning and intelligent optimization (LION3)*. Citeseer, pp. 1–15.

Paria, Biswajit et al. (2022). "Be Greedy–a Simple Algorithm for Blackbox Optimization using Neural Networks". In: *ICML2022 Workshop on Adaptive Experimental Design and Active Learning in the Real World*.

Phan-Trong, Dat, Hung Tran-The, and Sunil Gupta (2023). "NeuralBO: A black-box optimization algorithm using deep neural networks". In: *Neurocomputing* 559, p. 126776. ISSN: 0925-2312. DOI: `https://doi.org/10.1016/j.neucom.2023.126776`. URL: `https://www.sciencedirect.com/science/article/pii/S0925231223008998`.

Picheny, Victor et al. (2016). "Bayesian optimization under mixed constraints with a slack-variable augmented Lagrangian". In: *Advances in neural information processing systems* 29.

Rasmussen, Carl Edward, Christopher KI Williams, et al. (2006). *Gaussian processes for machine learning*. Vol. 1. Springer.

Schonlau, Matthias, William J Welch, and Donald R Jones (1998). "Global versus local search in constrained optimization of computer models". In: *Lecture notes-monograph series*, pp. 11–25.

Shahriari, Bobak et al. (2015). "Taking the human out of the loop: A review of Bayesian optimization". In: *Proceedings of the IEEE* 104.1, pp. 148–175.

Snelson, Edward and Zoubin Ghahramani (2007). "Local and global sparse Gaussian process approximations". In: *Artificial Intelligence and Statistics*. PMLR, pp. 524–531.

Snoek, Jasper, Hugo Larochelle, and Ryan P Adams (2012). "Practical bayesian optimization of machine learning algorithms". In: *Advances in neural information processing systems* 25.

Snoek, Jasper et al. (2015). "Scalable bayesian optimization using deep neural networks". In: *International conference on machine learning*. PMLR, pp. 2171–2180.

Springenberg, Jost Tobias et al. (2016). "Bayesian optimization with robust Bayesian neural networks". In: *NeurIPS* 29.

Srinivas, Niranjan et al. (2009). "Gaussian process optimization in the bandit setting: No regret and experimental design". In: *arXiv preprint arXiv:0912.3995*.

Streltsov, Simon and Pirooz Vakili (1999). "A non-myopic utility function for statistical global optimization algorithms". In: *Journal of Global Optimization* 14, pp. 283–298.

Swersky, Kevin, Jasper Snoek, and Ryan P Adams (2013). "Multi-task bayesian optimization". In: *Advances in neural information processing systems* 26.

Takeno, Shion et al. (2022). "Sequential and parallel constrained max-value entropy search via information lower bound". In: *International Conference on Machine Learning*. PMLR, pp. 20960–20986.

Tran, Dustin, Rajesh Ranganath, and David M Blei (2016). "The variational Gaussian process". In: *4th International Conference on Learning Representations, ICLR 2016*.

Tran-The, Hung et al. (2022). "Regret bounds for expected improvement algorithms in Gaussian process bandit optimization". In: *International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 8715–8737.

Turgeon, Martine, Cindy Lustig, and Warren H Meck (2016). "Cognitive aging and time perception: Roles of Bayesian optimization and degeneracy". In: *Frontiers in aging neuroscience* 8, p. 102.

Vakili, Sattar, Kia Khezeli, and Victor Picheny (2021). "On information gain and regret bounds in gaussian process bandits". In: *International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 82–90.

Vakili, Sattar et al. (2021). "Optimal order simple regret for Gaussian process bandits". In: *Advances in Neural Information Processing Systems* 34, pp. 21202–21215.

Wang, Sifan, Xinling Yu, and Paris Perdikaris (2022). "When and why PINNs fail to train: A neural tangent kernel perspective". In: *Journal of Computational Physics* 449, p. 110768.

Wang, Zi and Stefanie Jegelka (2017). "Max-value entropy search for efficient Bayesian optimization". In: *International Conference on Machine Learning*. PMLR, pp. 3627–3635.

Xu, Jiaming and Hanjing Zhu (2024). "Overparametrized Multi-layer Neural Networks: Uniform Concentration of Neural Tangent Kernel and Convergence of Stochastic Gradient Descent". In: *Journal of Machine Learning Research* 25.94, pp. 1–83.

Xu, Pan et al. (2020). "Neural contextual bandits with deep representation and shallow exploration". In: *preprint arXiv:2012.01780*.

Xu, Wenjie et al. (2023). "Constrained efficient global optimization of expensive black-box functions". In: *International Conference on Machine Learning*. PMLR, pp. 38485–38498.

Zhang, Weitong et al. (2021). "Neural Thompson Sampling". In: *International Conference on Learning Representation (ICLR)*.

Zhou, Dongruo, Lihong Li, and Quanquan Gu (2020). "Neural contextual bandits with ucb-based exploration". In: *International Conference on Machine Learning*. PMLR, pp. 11492–11502.

Zhou, Xingyu and Bo Ji (2022). "On kernelized multi-armed bandits with constraints". In: *Advances in neural information processing systems* 35, pp. 14–26.