On the Adversarial Robustness of Bayesian Machine Learning Models



Arno C. Blaas

Christ Church

University of Oxford

Submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

Trinity Term 2021

Abstract

Bayesian machine learning (ML) models have long been advocated as an important tool for safe artificial intelligence. Yet, little is known about their vulnerability against adversarial attacks. Such attacks aim to cause undesired model behaviour (e.g. misclassification) by crafting small perturbations to regular inputs which appear to be insignificant to humans (e.g. slight blurring of image data). This fairly recent phenomenon has undermined the suitability of many ML models for deployment in safety critical applications. In this thesis, we investigate how robust Bayesian ML models are against adversarial attacks, focussing on Gaussian process (GP) and Bayesian neural network (BNN) classification models. In particular, for GP classification models, we derive guarantees for the robustness against adversarial attacks that facilitate the evaluation of their suitability for a given safety critical application. Furthermore we investigate whether better posterior approximations benefit the adversarial robustness of BNNs, comparing the adversarial robustness resulting from high quality posterior approximations to that resulting from deterministic approximations in a range of experiments. We find that well approximated BNNs tend to be empirically more susceptible to adversarial attacks than deterministic neural network models for most popular priors. This calls for caution when deploying such BNNs in safety critical applications. Lastly, we show that using GPs in a Bayesian optimisation framework, it is possible to craft successful adversarial perturbations in black-box scenarios (where the attacked model is a priori unknown and can only be studied by querying it) with fewer model queries than previously needed. This facilitates a better evaluation of the adversarial robustness of a model in black-box scenarios.

Statement of Originality

I hereby declare that, except where made explicitly clear, the contents of this dissertation are my own original work, and to the best of my knowledge do not contain materials submitted in whole, or in part, for consideration for any other degree or qualification at the University of Oxford or any other academic or professional institution.

Arno Christian Blaas
May 13, 2021

Acknowledgements

First and foremost, I would like to thank my supervisor Stephen J. Roberts for enabling me to having pursued this unique experience and for helping me along the way with his guidance, patience, and generous support. Second, I would like to thank my parents, my sister, Gretel, and my friends for their unconditional trust and support. Third, I would like to thank my collaborators and in general the brilliant people in the MLRG and the OMI for plenty of interesting and fruitful discussions. A special thanks goes to Jen and Helen from the OMI Admin team. Also, I thank Wharf24 for making the year of Covid lockdowns a year to remember rather than a year to forget. Lastly, I thank the Konrad-Adenauer-Stiftung for supporting me with a PhD scholarship.

A note on originality

Chapters 3 and 5 of the thesis have been in very similar form published at AISTATS 2020 (Blaas et al., 2020) and ICLR 2020 (Ru et al., 2019) and have been joint efforts with my coauthors. While it is difficult to fully disentangle the joint efforts I will try to point out my specific contributions below. Apart from that, all work was done under general guidance and supervision of Stephen J. Roberts.

Chapter 3 In Chapter 3, I came up with the original idea for the bounds we obtained in the paper and they were mostly derived by me for the binary case as presented in Propositions 1 and 2, as well as Lemma 1. The improved binary case bounds using the Wasserstein distance were obtained after the collaboration and as such are fully my own work. The algorithm for bounding the latent mean and variance as well as the branch and bound algorithm were mostly Andrea Patane's contribution, while the bounds for the multiclass case were obtained mostly by Luca Laurenti. All experiments except for the interpretability experiment on MNIST358 were done by me.

Chapter 5 In Chapter 5, the original idea was generated by Binxin Ru (main author) and Adam Cobb. I contributed to the optimal dimensionality reduction techniques, the comparison to related work and more generally to the experiments, with a particular focus on kernel selection and hyperparameter tuning.

Contents

List of Figures viii

List of Tables ix

1 Introduction 1
- 1.1 The Rise of Machine Learning . . . 1
- 1.2 The Need for Safe Machine Learning . . . 2
- 1.3 Research Objectives and Contributions . . . 4
- 1.4 Structure of the Thesis . . . 5

2 Background and Preliminaries 7
- 2.1 Learning from Data with Bayes . . . 8
  - 2.1.1 Introduction to the Bayesian Paradigm . . . 8
  - 2.1.2 Bayesian Neural Networks . . . 17
  - 2.1.3 Gaussian Processes . . . 22
  - 2.1.4 Relationship between Bayesian Neural Networks and Gaussian Processes . . . 30
  - 2.1.5 Other Bayesian Machine Learning Models . . . 32
- 2.2 Adversarial Attacks and Adversarial Robustness . . . 33
  - 2.2.1 Archetypes of Adversarial Attacks . . . 34
  - 2.2.2 Implementations of Adversarial Attacks . . . 36
  - 2.2.3 Measuring Adversarial Robustness and Evaluating Adversarial Attacks . . . 39

3 Adversarial Robustness Guarantees for GP Classification 42
- 3.1 Introduction . . . 42
- 3.2 Related Work . . . 43
- 3.3 Preliminaries . . . 45
  - 3.3.1 Bayesian Classification with Gaussian Processes . . . 45
  - 3.3.2 Problem Formulation: Adversarial Robustness Guarantees . . . 46
- 3.4 Guarantees for Binary Classification . . . 48
  - 3.4.1 Outline of Approach . . . 48
  - 3.4.2 Computation of Bounds: General Case . . . 50

3.4.3 Computation of Bounds: Special Case of Probit Likelihood . 54
3.4.4 Obtaining Bounds on Latent Mean and Variance 55
3.5 Branch and Bound Algorithm 58
3.5.1 The Algorithm 58
3.5.2 Computational Complexity 61
3.6 Guarantees for Multiclass Classification 62
3.7 Experimental Results 66
3.7.1 Experimental Set Up 66
3.7.2 Adversarial Local Safety 71
3.7.3 Adversarial Local Robustness 72
3.7.4 Sensitivity Analysis 74
3.7.5 Runtime Analysis 77
3.8 Improved Bounds for Binary Classification 79
3.8.1 Tighter Outer Bounds via Wasserstein Distance Bounding . 79
3.8.2 Experiments Using the New Bound 81
3.9 Discussion 83

4 Does Being More Bayesian Increase the Adversarial Robustness of BNNs? 84
4.1 Introduction 84
4.2 Related Work 85
4.2.1 Works on Bayesian Neural Networks 85
4.2.2 Randomised Smoothing Techniques 87
4.2.3 Ensemble Methods 88
4.3 Problem Definition, Research Questions, and Outline 89
4.4 Experimental Set Up 91
4.4.1 Data 91
4.4.2 Models 92
4.4.3 Inference 93
4.4.4 AGA Estimation 95
4.5 Experimental Results 97
4.5.1 Multinomial Logistic Regression / Neural Network with Zero Hidden Layers 97
4.5.2 Neural Network with Three Hidden Layers 99
4.6 Discussion 101

5 BayesOpt Adversarial Black-box Attack **107**
5.1 Introduction . . . 107
5.2 Related Work . . . 108
5.3 Preliminaries . . . 111
5.3.1 Problem Definition . . . 111
5.3.2 Bayesian Optimisation . . . 112
5.4 BayesOpt Attack . . . 113
5.4.1 BayesOpt attack objective . . . 113
5.4.2 GP-based BayesOpt Attack . . . 114
5.4.3 Additive GP-based BayesOpt Attack . . . 115
5.4.4 Learning The Optimal d^r . . . 116
5.5 Experimental Results . . . 118
5.5.1 Experimental Set Up . . . 118
5.5.2 Effect of d^r . . . 121
5.5.3 Performance Under a Fixed Query Budget . . . 122
5.5.4 Query Efficiency Comparison . . . 126
5.5.5 Additional Experiments on MNIST and CIFAR10 . . . 127
5.6 Discussion . . . 130

6 Conclusions and Future Work **132**
6.1 Conclusions . . . 132
6.2 Future Work . . . 134
6.2.1 Extensions of our Work . . . 134
6.2.2 New Directions . . . 135

Bibliography **137**

List of Figures

1.1 Tesla Autopilot real-world adversarial example 2

2.1 Illustration of a fully connected feedforward neural network with $K-1$ hidden layers. 19
2.2 Example of the predictive posterior distribution of a Gaussian process. 24
2.3 Example of a an adversarial evasion attack. 35

3.1 Illustration of GPC adversarial robustness guarantee's branch and bound approach. 49
3.2 GPC safety analysis on Synthetic2D and SPAM. 67
3.3 GPC safety analysis on MNIST38. 68
3.4 GPC adversarial local robustness comparison between Laplace and EP approximation. 73
3.5 GPC interpretability analysis on MNIST358. 75
3.6 GPC interpretability analysis on Synthetic2D and SPAM. 77
3.7 Runtime analysis for GPC adversarial robustness guarantees. 78
3.8 Comparison of improved GPC bounds and original GPC bounds. . 82

4.1 Samples from MNIST and FashionMNIST data set. 91
4.2 The linear Mlogreg model used. 92
4.3 The 3 hidden layer neural network DNN3 model used. 93
4.4 Adversarial global accuracy for Mlogreg model on MNIST. 97
4.5 Adversarial global accuracy for Mlogreg model on FashionMNIST. . 98
4.6 Adversarial global accuracy for DNN3 model on MNIST. 101
4.7 Adversarial global accuracy for DNN3 model on FashionMNIST. . . 102

5.1 Sample images from CIFAR10 and ImageNet data sets. 119
5.2 Performance of BayesOpt attack with GP surrogate in different reduced dimension d^r. 121
5.3 CIFAR10 adversarial examples generated by BayesOpt attack. . . . 125
5.4 Query efficiency of BayesOpt Attack vs. other black-box attacks across data sets. 127
5.5 Attack objective value against BayesOpt iterations. 130

List of Tables

3.1 Comparison of improved GPC bounds and original GPC bounds on SPAM. 82

4.1 Hyperparameter settings for HMC algorithms per model and data set. 94
4.2 Further key metrics of predictive functions of Mlogreg model on MNIST and FashionMNIST data sets. 99
4.3 Further key metrics of predictive functions of DNN3 model on MNIST and FashionMNIST data sets. 103

5.1 Black-box attack results on 50 random MNIST images. 123
5.2 Black-box attack results on 50 random CIFAR10 images. 123
5.3 Black-box attack results on 50 random ImageNet images. 124
5.4 Summary of attack results on 7 MNIST images. 128
5.5 Summary of attack results on 27 CIFAR10 images. 129
5.6 Summary of attack results on CIFAR10 for different decomposition learning methods. 130

1 Introduction

1.1 The Rise of Machine Learning

Over the last decade, we have witnessed unprecedented advances in the field of machine learning (ML) and artificial intelligence (AI). Fuelled by the successes of deep learning models on tasks such as image recognition and speech recognition at the beginning of the decade (e.g. the milestone performance of AlexNet (Krizhevsky et al., 2012) on the ImageNet competition (Russakovsky et al., 2015) in 2012), interest in ML/AI[1] research has since been ever increasing, leading to even more impressive successes. These in turn attracted further research and funding. An illustration of this self-perpetuating dynamic can be found, for example, in the number of paper submissions to the biggest annual ML conference (NIPS/NeurIPS), which has increased almost tenfold between 2010 and 2020 (from around $1,000$ to $10,000$). Big software companies such as Google and Microsoft have adopted an AI-first strategy to their product development (Taulli, 2019) and contribute heavily to the research in the field, regularly publishing more papers at big ML conferences than most academic institutions.

The resulting advances in ML lead to the deployment of ML models in a multitude of real-world systems and applications (LeCun et al., 2015), ranging from identity

[1] While these two terms are often used interchangeably, we see ML strictly speaking as (the main) subfield of (modern) AI.

verification (Liu et al., 2018b), through financial services (Heaton et al., 2017), to translation (Wu et al., 2016). In general, this development creates improved and more affordable services and products for people around the world. However, the increased deployment of ML models does also come with new risks.



Figure 1.1: Screenshot of Figure 33 in Tencent Keen Security Lab Technical research paper on Tesla Autopilot (Tencent Keen Security Lab, 2019). **Left:** Front camera image of Tesla S. Highlighted in red squares are the white points which constitute the adversarial example. **Right:** Processed input for controller of lane keeping assistant.

1.2 The Need for Safe Machine Learning

More recently, the potential of ML has been increasingly uncovered also in safety critical applications, such as autonomous driving (Bojarski et al., 2016) or healthcare (Hu et al., 2018). While it can be questioned whether processes will ever be entirely autonomous in such applications, even in the present case of having humans in the loop in some way or another (e.g. doctors using outcomes of ML predictions in healthcare, humans still sitting in the driving seat and retaining the possibility to interfere in autonomous driving) there is a need for the used ML models and methods to be *safe* in order to avoid catastrophic outcomes in such safety critical applications.

Such a safe ML model should arguably satisfy a number of desiderata. For example, depending on the use case, one might require a model to be interpretable (e.g. in healthcare (Vellido, 2019)). Another important component of safety might be a reliable awareness of prediction uncertainty, i.e. for a model to know when it does not know. For example in autonomous driving, such a feature could help

prevent accidents by forcing the driver to take over in situations of high uncertainty (NHTSA, Jan. 2017; McAllister et al., 2017). The Bayesian machine learning framework naturally lends itself to this requirement, as it allows us to account for epistemic uncertainty in predictions by means of a posterior distribution over prediction functions. For example, incorporating a Bayesian machine learning model into a decision pipeline for medical diagnosis can lead to improved overall accuracy by means of involving a human expert when prediction uncertainty is high (Mobiny et al., 2019).

Another important desideratum for safe ML models is that they should not change their predictions drastically near data points, thus it should not be easy to fool them by small perturbations which are visually imperceptible to the human eye (Szegedy et al., 2013; Carlini et al., 2016). A real-life example where such an unwanted behaviour can cause catastrophic outcomes can be found in (Tencent Keen Security Lab, 2019), where the authors successfully fooled the lane control algorithm of a Tesla S by perturbing only a few pixels of the front camera input which led the lane control system to perceive a non-existant lane, potentially requiring a lane-switch, as illustrated in Figure 1.1. The danger of such minor perturbations fooling the model, often called adversarial examples, lies in their imperceptibility by the human eye, which prevents timely human intervention to avoid the catastrophic outcomes they were designed to cause (e.g. lane switching into oncoming traffic).

The robustness of a model against these adversarial examples has been increasingly the focus of ML research in recent years, with "TrustGeneralML / AdversarialExamples" being the fourth most common ML research sub-area among all submissions to the second largest annual ML conference, ICML 2020 (Lambda, 2020). However, most existing works focus on adversarial examples for models for which the posterior distribution over functions is approximated with Dirac delta distributions on a single function, omitting much of the Bayesian nature of inference (which we shall thus refer to as *deterministic* models, Szegedy et al. (2013); Madry et al. (2017); Carlini and Wagner (2017a)). Given the suitability of (true) Bayesian models to

safety critical applications, we aim to contribute to a better understanding of the adversarial robustness of Bayesian ML models in this thesis.

1.3 Research Objectives and Contributions

In order to contribute to a better understanding of the adversarial robustness of Bayesian ML models, we align our research endeavours to the following fundamental research questions as guiding principles:

- How susceptible are Bayesian machine learning models to adversarial examples in the white-box scenario, where attackers can access any desired information about the attacked model?
- Can we establish theoretical guarantees on the robustness of Bayesian machine learning against such adversarial examples?
- Are Bayesian machine learning models more/less robust against adversarial examples than their deterministic counterparts? Does being more Bayesian help?
- How do the answers to the previous questions differ in the more realistic black-box scenario, where attackers can only gain information about the attacked model by querying it?

The specific contributions we make as (partial) answers to these questions are as follows:

1. For Bayesian classification with Gaussian process classification (GPC) models, we propose a method for computing, for any compact set of input points, the class probability ranges of a GPC model across all points in that set, up to any precision $\epsilon > 0$. This allows us to analyse robustness and *guarantee safety against adversarial attacks* for GPC models, which we demonstrate on multiple data sets and approximate Bayesian inference techniques for GPC models.

2. In a case study, we empirically compare the adversarial robustness of Bayesian neural networks with well approximated posteriors (BNNs) against the robustness of identical neural network architectures for which the posterior has been approximated with a Dirac delta distribution using stochastic gradient descent. Contrary to previous findings and hypotheses, we demonstrate that being more Bayesian does *not* automatically lead to higher adversarial robustness, but can in fact even lead to worse adversarial robustness. We also investigate the effect of differences in prior distributions on the adversarial robustness of BNNs, and find that it can indeed be significant. In particular, we find uninformative, wide priors to generally lead to worse adversarial robustness in our experiments.

3. We introduce a novel black-box attack method, BayesOpt attack, that improves the quality of practical assessments of a model's black-box robustness by needing *2 to 5 times* fewer model queries than previous attacks. It uses Bayesian optimisation with Gaussian process surrogate models to find adversarial examples that allows the incorporation of Bayesian model selection techniques to learn the optimal degree of search dimensionality reduction. We demonstrate its query-efficiency on three benchmark data sets and models.

1.4 Structure of the Thesis

The thesis is structured as follows. After this introductory chapter, we first review the necessary concepts and prerequisites that form the basis of our research contributions in a background chapter (Chapter 2). We then start our contributions with the work on adversarial robustness and safety guarantees for GPC models in Chapter 3. This is followed by our work that compares the adversarial robustness of well approximated Bayesian neural networks to deterministic neural networks in Chapter 4. In Chapter 5, we present our work on using Bayesian optimisation to improve the assessment of model robustness against black-box-attacks. Finally, in

Chapter 6 we have a short overall conclusive discussion of our work and discuss interesting directions for future work.

2

Background and Preliminaries: Bayesian Inference and Adversarial Attacks in Supervised Learning Problems

This chapter offers an introduction to the concepts of Bayesian inference for the general problem of learning from data in a supervised setting, as well as introducing adversarial attacks as a possible threat to models used in such problems. We consider the basic concepts, with the aim of covering background material for the remainder of the thesis. The chapter is structured as follows: in Section 2.1, we first provide an overview to the core elements of Bayesian inference for supervised learning problems and how inference and prediction can be performed. Then, we exhibit how problems can be solved using parametric models, working from simple linear models to deep neural networks. From there we move to Bayesian non-parametric models and introduce regression and classification with Gaussian processes. After covering the range of models and inference methods we use in subsequent chapters, we use Section 2.2 to describe the problem of adversarial attacks. We provide an overview of different adversarial attack archetypes as well as covering their implementation and evaluation.

2.1 Learning from Data with Bayes

2.1.1 Introduction to the Bayesian Paradigm

We first introduce the core principles and elements of Bayesian learning. We consider the problem of approximative inference, which we subsequently describe, before ending Section 2.1.1 with a discussion of Bayesian prediction.

The Core Elements of Bayesian Learning

There are three key components to the Bayesian approach to learning from data: the data itself, assumptions regarding the data generating process, and the inference.

Data. Throughout this thesis, we consider problems in which the data consists of input and output pairs of the form $\mathcal{D}_n = \{(x^{(i)}, y^{(i)})\}_{1 \leq i \leq n}$, i.e. so called supervised learning problems. The inputs $x^{(i)}$ can be represented as vectors in the problems we consider, i.e. $x^{(i)} \in \mathbb{R}^d$. The outputs $y^{(i)}$ can be represented either as continuous vectors, i.e. $y^{(i)} \in \mathbb{R}^{d'}$, or as one of a finite set of integers, i.e. $y^{(i)} \in \{1, \ldots, C\}$ for some integer C, depending on the type of supervised learning problem. The first case is referred to as a regression problem, while the latter is called a classification problem. In classification problems, an alternative notation for the outputs is given by one-hot encoding $y^{(i)}$ into a vector $Y^{(i)} \in \{0, 1\}^C$ as a vector of 0s with the exception of a 1 in dimension $y^{(i)}$.

Assumptions. The assumptions about the nature of the data generating process are made *a-priori*, that is before the data $\mathcal{D}_n$ is observed. Since they are made a-priori, they have to take some uncertainty about the true underlying data generating process into account. This is expressed in the language of uncertainty - probability distributions.

On a first level, uncertainty about the observed output (aleatoric, or irreducible uncertainty) is incorporated by modelling the outputs as a combination of a mapping from inputs to $\mathbb{R}^{d'}/\{0, 1\}^C$ and a probability distribution $p(\xi)$. For regression

problems, this probability distribution explains the difference between observed outputs and input mappings, that is

$$\begin{aligned} \xi &= y - f(x) \\ \Leftrightarrow \quad y &= f(x) + \xi \end{aligned} \tag{2.1}$$

A common choice for the distribution of ξ is $\boldsymbol{\Xi} \sim \mathcal{N}(\boldsymbol{\Xi}|0, \sigma^2 I)$ for any finite collection $\boldsymbol{\Xi}$ of realisations of ξ (also called white noise). For classification problems, the deviation of observed Y and input mapping $f(x)$ is modelled through a multinomial distribution on ξ with $\xi_c = 1$ with probability $\pi_c = f(x)_c$ and $\xi_c = 0$ with probability $1 - \pi_c$ for all $c \in \{1, \ldots, C\}$, resulting in

$$\begin{aligned} Y &= \xi \\ \Leftrightarrow \quad y &= \prod_{c=1}^{C} c^{\xi_c} \end{aligned} \tag{2.2}$$

Again, we assume any two draws of ξ to be independent.

On a second level, uncertainty about the function f (epistemic, or reducible uncertainty) is encoded through further probability distributions. It is also called reducible, as observing data will reduce this source of uncertainty, until in the limiting case of having observed an infinite amount of data points generated by the data generating process, no uncertainty about the function f is left[1]. In the following, we show how this is achieved with Bayesian inference for parametric models, in which an explicitly parametrised form of f is assumed. In this case, epistemic uncertainty can be expressed in probability distributions over the function / model parameters θ.

Inference. The observed data $\mathcal{D}_n$, once observed, can be used to reduce the epistemic uncertainty about the data generating process. This inference is done by applying Bayes' rule to the probability distributions involved:

$$p(\theta|\mathcal{D}_n) = \frac{p(\mathcal{D}_n|\theta)p(\theta)}{p(\mathcal{D}_n)} \tag{2.3}$$

This way, we obtain the posterior probability distribution $p(\theta|\mathcal{D}_n)$ as the product of the *prior* probability distribution $p(\theta)$ and the data *likelihood* $p(\mathcal{D}_n|\theta)$, divided

[1] At least under mild conditions, see Doob (1949); Barron et al. (1999)

by the *evidence* $p(\mathcal{D}_n)$. If the parametric structure of f has been correctly specified, the results by Doob (1949) guarantee that $p(\theta|\mathcal{D}_n)$ obtained this way converges to a Dirac delta function centred on the true parameters of the data generating process as $n \to \infty$, i.e. all epistemic uncertainty will have been reduced. However, while Eq. (2.3) looks simple at first glance, exact inference according to it is seldom possible. This is because the evidence in the denominator

$$p(\mathcal{D}_n) = \int p(\mathcal{D}_n|\theta)p(\theta)d\theta \tag{2.4}$$

involves integration that is only tractable for cases when prior and likelihood are *conjugate*, as for example with two Gaussian distributions. In the non-conjugate cases, we need to resort to approximate inference.

Approximate Bayesian Inference

There are three main approaches to approximate $p(\theta|\mathcal{D}_n)$: with Dirac delta functions (maximum a-posteriori or maximum likelihood estimation), with more general functions (e.g. Laplace, variational inference) or through the usage of sampling based methods (e.g. Monte Carlo Markov Chain, Hamiltonian Monte Carlo). We present the ones most relevant to this thesis below.

Dirac Delta Function Approximations. The simplest way to approximate the posterior distribution in Eq. (2.3) is by using a Dirac delta function, which is only assigning positive probability mass to a single set of parameters θ. A natural choice for this specific θ is the maximum of $p(\theta|\mathcal{D}_n)$, i.e. approximating the posterior only by its mode. This approach is referred to as *maximum a-posteriori* (MAP), and it seeks to find

$$\begin{aligned} \theta^{MAP}(\mathcal{D}_n) &= \arg\max_\theta p(\mathcal{D}_n|\theta)p(\theta) \\ &= \arg\max_\theta \prod_{i=1}^{n} p(y^{(i)}|x^{(i)},\theta)p(\theta) \\ &= \arg\max_\theta \left[\sum_{i=1}^{n} \log p(y^{(i)}|x^{(i)},\theta) + \log p(\theta)\right] \end{aligned} \tag{2.5}$$

where the first equality follows from the conditional independence assumptions in Eqns. (2.1) and (2.2) and the second simply uses the monotonicity of the logarithmic transform. This optimisation problem can be solved using gradient methods (facilitated by the logarithmic transformation), where depending on the context methods such as stochastic gradient descent (SGD) algorithm (Bottou, 2012) can be used. Most of the models that we referred to as 'deterministic' in Chapter 1 use this approximation. Crucially, they do not account for uncertainty about θ in the posterior. At least, the a-priori uncertainty about θ has an effect on the solution found to Eq. (4.5) via $p(\theta)$.

This is however not the case if an uninformative, flat prior $p(\theta)$ (also called improper prior), which puts equal probability mass on all θ, is used. In this case MAP collapses into *maximum likelihood estimation* (MLE), that is into finding

$$\begin{aligned} \theta^{MLE}(\mathcal{D}_n) &= \arg\max_{\theta} p(\mathcal{D}_n|\theta) \\ &= \arg\max_{\theta} \sum_{i=1}^{n} \log p(y^{(i)}|x^{(i)}, \theta) \end{aligned} \tag{2.6}$$

Note however, that in the case of parametric models, this procedure neither incorporates prior assumptions about the distribution of θ, nor does it encode uncertainties in the posterior.

General Function Approximations. In order to account for uncertainty about θ in the posterior approximation, one needs to go beyond Dirac delta functions and move to more general function approximations, i.e. approximate $p(\theta|\mathcal{D}_n)$ with some tractable probability distribution $q(\theta)$. There are several approaches to do so, such as the Laplace approximation which is based on Laplace's method (de Laplace, 1886), Expectation Propagation (EP, Minka (2001)), or Mean-Field Variational Inference (MFVI, Saul et al. (1996)). We briefly outline each of these below.

Laplace Approximation. The Laplace approximation goes one step beyond the MAP approximation by approximating $p(\theta|\mathcal{D}_n)$ with a Gaussian distribution centered around $\theta^{MAP}(\mathcal{D}_n)$. The question that thus arises is how to determine the covariance

matrix accordingly. The Laplace approach to this is to use a second order Taylor expansion of the log-posterior around $\theta^{MAP}(\mathcal{D}_n)$, that is

$$\begin{aligned}\log p(\theta|\mathcal{D}_n) &\approx \log p(\theta^{MAP}|\mathcal{D}_n) + \frac{(\theta-\theta^{MAP})^T \nabla p(\theta^{MAP}|\mathcal{D}_n)}{p(\theta^{MAP}|\mathcal{D}_n)} \\ &\quad + \frac{1}{2}(\theta-\theta^{MAP})^T[\nabla\nabla \log p(\theta^{MAP}|\mathcal{D}_n)](\theta-\theta^{MAP})\end{aligned} \tag{2.7}$$

where $\nabla p(\theta^{MAP}|\mathcal{D}_n)$ is the gradient of $p(\theta|\mathcal{D}_n)$ at θ^{MAP} and $\nabla\nabla \log p(\theta^{MAP}|\mathcal{D}_n)$ the Hessian of the log posterior at θ^{MAP}. Since θ^{MAP} maximises $p(\theta|\mathcal{D}_n)$, the gradient is zero and drops out of the equation, leaving us with

$$\begin{aligned}\log p(\theta|\mathcal{D}_n) &\approx \log p(\theta^{MAP}|\mathcal{D}_n) \\ &\quad + \frac{1}{2}(\theta-\theta^{MAP})^T[\nabla\nabla \log p(\theta^{MAP}|\mathcal{D}_n)](\theta-\theta^{MAP}).\end{aligned} \tag{2.8}$$

Taking the exponential on both sides and writing $\boldsymbol{m} := \theta^{MAP}$ as well as $\boldsymbol{M} := -\nabla\nabla \log p(\theta^{MAP}|\mathcal{D}_n)$, we get

$$p(\theta|\mathcal{D}_n) \propto \exp\left(-\frac{1}{2}(\theta-\boldsymbol{m})^T \boldsymbol{M}(\theta-\boldsymbol{m})\right), \tag{2.9}$$

that is

$$p(\theta|\mathcal{D}_n) \approx \mathcal{N}(\theta|\boldsymbol{m}, \boldsymbol{M}^{-1}) =: q(\theta). \tag{2.10}$$

Mean-Field Variational Inference (MFVI). Just like the Laplace approximation, MFVI approximates the true posterior with a multivariate Gaussian distribution. However, it follows a different approach. It approximates $p(\theta|\mathcal{D}_n)$ with a simple multivariate Gaussian distribution $q_{\mathbf{m},\mathbf{s}}(\theta) = \mathcal{N}(\theta|\mathbf{m}, \text{diag}(\mathbf{s})I)$ by maximising the evidence lower bound

$$\text{ELBO}(\mathbf{m},\mathbf{s}) = \int \prod_{i=1}^{n} p(y^{(i)}|x^{(i)},\theta) q_{\mathbf{m},\mathbf{s}}(\theta) d\theta - \text{KL}(q_{\mathbf{m},\mathbf{s}}(\theta)||p(\theta)), \tag{2.11}$$

where $\text{KL}(q_{\mathbf{m},\mathbf{s}}(\theta)||p(\theta))$ is the KL-divergence between $q_{\mathbf{m},\mathbf{s}}(\theta)$ and the prior $p(\theta)$, defined as

$$\text{KL}(q(x)||p(x)) = \int q(x) \log\left(\frac{q(x)}{p(x)}\right) dx. \tag{2.12}$$

for any two probability density functions q and p. Maximising this ELBO is equivalent to maximising $\log p(\mathcal{D}_n) - \mathrm{KL}(q_{\mathbf{m},\mathbf{s}}(\theta)||p(\theta|\mathcal{D}_n))$, i.e. simultaneously maximising log-evidence and minimising the KL-divergence between $q_{\mathbf{m},\mathbf{s}}(\theta)$ and the true posterior $p(\theta|\mathcal{D}_n)$, which as an objective could not be minimised in tractable form directly. MFVI thus also turns the original integration problem in Eq. (2.3) into an optimisation problem, which can be tackled using the same optimisation tools as for the MAP optimisation.

Expectation Propagation (EP). Similar to MFVI, EP aims to approximate $p(\theta|\mathcal{D}_n)$ with a parametrised Gaussian $q(\theta)$ by minimising a KL divergence. Here, the original aim is to minimise

$$\mathrm{KL}(p(\theta|\mathcal{D}_n)||q(\theta)), \tag{2.13}$$

which however, cannot be done directly as the true posterior $p(\theta|\mathcal{D}_n)$ is unknown. Instead, EP suggests the following approach. First, it assumes $q(\theta)$ to be of the form

$$q(\theta) = \frac{1}{Z}\prod_{i=1}^{n} q_i(\theta)p(\theta) \tag{2.14}$$

with a normalising constant Z and $q_i(\theta) = \mathcal{N}(\theta|\mathbf{m}^{(i)}, s^{(i)}I)$. The minimisation is then tackled in an iterative algorithm, where, after initialisation of each $\mathbf{m}^{(i)}$ and $s^{(i)}$, at each step of the algorithm one $i \in \{1, \ldots, n\}$ is picked for refinement of $q_i(\theta)$. The refinement is done by first calculating

$$q_{-i}(\theta) \propto \frac{q(\theta)}{q_i(\theta)} \tag{2.15}$$

and then instead combining $q_{-i}(\theta)$ with $p(y^{(i)}|x^{(i)}, \theta)$ to get

$$\hat{p}(\theta) \propto p(y^{(i)}|x^{(i)}, \theta)q_{-i}(\theta). \tag{2.16}$$

$\hat{p}(\theta)$ is then used as the target distribution for the update of $q_i(\theta)$, which is achieved by updating $\mathbf{m}^{(i)}$ and $s^{(i)}$ so that their values minimise the following KL divergence:

$$q_i^{new}(\theta) = \arg\min_{q_i} \mathrm{KL}\left(\hat{p}(\theta)||q_{-i}(\theta)q_i(\theta)\right). \tag{2.17}$$

Since the KL divergence is taken between two Gaussians, a closed form solution can be derived using moment matching.
This process is done iteratively until convergence is reached (which is not always guaranteed, but can be reached as the existence of fix points is guaranteed (Minka, 2001)). Based on a Gaussian prior $p(\theta)$, the resulting approximation $q(\theta)$ is again Gaussian.

Unlike the Dirac delta function approximations, all three presented general function approximations account for the uncertainty over θ also in the posterior in form of a Gaussian distribution. However, if the true posterior $p(\theta|\mathcal{D}_n)$ is multimodal, a unimodal approximation, such as the Gaussian approximations presented here, might not reflect the true posterior well. In order to get better approximations in such a case, one might need to resort to sampling methods.

Sampling Methods. Sampling based methods typically employ algorithms of the Markov Chain Monte Carlo (MCMC) family to build chains of samples that converge towards an equilibrium distribution which matches the posterior distribution $p(\theta|\mathcal{D}_n)$ and subsequently draw samples from that distribution. The method we resort to in this thesis is Hamiltonian Monte Carlo (HMC, Neal et al. (2011)), which is guaranteed to converge towards the posterior distribution $p(\theta|\mathcal{D}_n)$ as an equilibrium distribution and referred to as the 'gold standard' of approximate Bayesian inference (Nemeth and Fearnhead, 2020). It builds on the Metropolis-Hastings algorithm (Hastings, 1970) which at each step t generates the next sample $\theta^{(t+1)}$ based on a proposal distribution $q(\theta^{(t+1)}|\theta^{(t)})$ and then accepts this sample as part of the chain with probability

$$p^{accept}(\theta^{(t+1)}|\theta^{(t)}) = \min\left(1, \frac{p(\theta^{(t+1)}|\mathcal{D}_n)q(\theta^{(t)}|\theta^{(t+1)})}{p(\theta^{(t)}|\mathcal{D}_n)q(\theta^{(t+1)}|\theta^{(t)})}\right), \tag{2.18}$$

which is tractable as the intractable denominator in Eq. (2.3) cancels out between nominator and denominator. If $\theta^{(t+1)}$ gets rejected, then $\theta^{(t+1)} = \theta^{(t)}$.
The key challenge is to choose a proposal distribution q such that the convergence

towards the target distribution $p(\theta|\mathcal{D}_n)$ is guaranteed and happens fast, which requires to balance the trade-off between low acceptance rates and over-cautious proposal samples (i.e. proposals that are nearly identical to the previous sample). HMC is an approach to achieve this by leveraging Hamiltonian dynamics. To this end, it augments θ to (θ, ν) and works on a probability distribution on the augmented space $p(\theta, \nu|\mathcal{D}_n) = p(\nu|\theta, \mathcal{D}_n)p(\theta|\mathcal{D}_n)$, which gets rewritten in terms of its Hamiltonian function H as (MacKay, 2003)

$$p(\theta, \nu|\mathcal{D}_n) = \mathrm{e}^{-H(\theta,\nu)} \tag{2.19}$$

with

$$\begin{aligned} H(\theta, \nu) = -\log p(\nu|\theta, \mathcal{D}_n) &- \log p(\theta|\mathcal{D}_n) \\ =: K(\nu, \theta) &+ V(\theta), \end{aligned} \tag{2.20}$$

where K is thought of as kinetic energy and V as the potential energy. The idea is to derive q by applying the dynamics of Hamilton's equations on this Hamiltonian. We focus on the case where the kinetic energy is assumed to be independent of θ, e.g. under the Gaussian specification $p(\nu|\theta, \mathcal{D}_n) = \mathcal{N}(\nu|0, M)$ with mass matrix M, which ultimately leads to the following iterative sampling algorithm (also called the leapfrog integrator (MacKay, 2003; Betancourt, 2017)):

$$\begin{aligned} \nu^{(i+0.5)} &= \nu^{(i)} - \frac{\eta}{2}\frac{\partial V}{\partial \theta}(\theta^{(i)}) \\ \theta^{(i+1)} &= \theta^{(i)} + \eta M^{-1}\nu^{(i+0.5)} \\ \nu^{(i+1)} &= \nu^{(i+0.5)} - \frac{\eta}{2}\frac{\partial V}{\partial \theta}(\theta^{(i+1)}) \end{aligned} \tag{2.21}$$

where η is the step size. Combining this with a sign-flip operation on $\nu^{(i+1)}$ yields a reversible proposal chain, that is starting from $(\theta^{(i=0)}, \nu^{(i=0)}) = (\theta^{(t)}, \nu^{(t)})$ and ending after a trajectory length of L steps with $(\theta^{(t+1)}, \nu^{(t+1)}) := (\theta^{(i=L)}, \nu^{(i=L)})$, we have $q(\theta^{(t)}|\theta^{(t+1)}) = q(\theta^{(t+1)}|\theta^{(t)})$ in the acceptance/rejection step of the original Metropolis-Hastings algorithm (Eq. (2.18)), which guarantees its convergence. Imposing the Hamiltonian dynamics on the proposal distribution q has the advantage that it captures the geometry of $p(\theta|\mathcal{D}_n)$, which is particularly useful in high

dimensional spaces (Betancourt, 2017).

The drawback of HMC (and other MCMC methods) is two-fold. Firstly, the speed of convergence of the resulting chain of samples towards samples from the posterior $p(\theta|\mathcal{D}_n)$ can be very sensitive to the choices of step size η, trajectory length L, and mass matrix M. Secondly, even for well-chosen η, L, and M, it can be computationally prohibitively expensive to obtain samples from the posterior distribution, as the convergence is only guaranteed in the infinite limit but finite time results are not available in the general case to the best of our knowledge. In practice, for mildly deep NNs with many parameters, it still seems impossible to approximate the true posterior using HMC in anything that we perceive as finite time.

Prediction with Bayes

Once an exact or approximate posterior distribution $p(\theta|\mathcal{D}_n)$ has been obtained in one of the ways outlined above, it can be used for prediction on new, unseen test inputs x^*. For practical reasons, point predictions are often desirable (e.g. in a classification problem, ultimately a single class has to be picked as predicted class). These can be obtained by taking a single sample from the posterior, or, more commonly, if canonical loss functions (mean squared error for regression and cross-entropy for classification) are assumed, by marginalising over θ to get

$$\hat{f}^{(n)}(x^*) = \int f^{\theta}(x^*)p(\theta|\mathcal{D}_n)d\theta. \tag{2.22}$$

If the maximum likelihood or maximum a-posteriori approximations have been used for $p(\theta|\mathcal{D}_n)$, $\hat{f}^{(n)}(x^*)$ simplifies to $\hat{f}^{\hat{\theta}}(x^*)$ where $\hat{\theta}$ is either θ^{MLE} or θ^{MAP}. Then, for regression problems we use as point prediction $\hat{f}^{(n)}(x^*)$, whereas for classification problems the class with highest values of $\hat{f}^{(n)}(x^*)$, i.e. $\arg\max_c \hat{f}^{(n)}(x^*)_c$ is selected. Lastly, a simple measure for evaluating prediction uncertainty can be obtained by analysing for example the predictive variance score

$$\mathbb{V}(f(x^*)) = \int \|f^{\theta}(x^*) - \hat{f}^{(n)}(x^*)\|_2^2 p(\theta|\mathcal{D}_n)d\theta, \tag{2.23}$$

where $\|\cdot\|_2$ denotes the Euclidean norm. In the univariate case this exactly corresponds to the predictive variance. Such a measure can be helpful in order

to know when to trust the predictions of a model, which can be crucial in safety critical applications and principled decision making in general.

It is also worth mentioning that other measures for evaluating prediction uncertainty, especially for the multi-class classification case, have been proposed, such as the predictive entropy or the mutual information (MI) between θ and y^* at x^*. In contrast to the predictive entropy, MI allows to disentangle epistemic from aleatoric uncertainty (Smith and Gal, 2018). Interestingly, Smith and Gal (2018) show that the predictive variance score in Eq. (2.23) corresponds to the terms of order < 2 of a Taylor-series expansion of the logarithmic terms of the MI between θ and y^* at x^*, which makes their performance as prediction uncertainty evaluation measure closely tied.

Now that we have introduced the key elements and principles of Bayesian learning, we turn to two popular Bayesian models: Bayesian neural networks (BNNs) and Gaussian processes (GPs). First, we introduce BNNs.

2.1.2 Bayesian Neural Networks

Bayesian neural networks (BNNs) fall into the class of parametric models, which assume a parametric function f^θ to map inputs to outputs. Below, we introduce them, starting from simple linear models and showing how deep neural networks can be obtained as a combination of non-linear extensions of such simple linear models.

From Linear Models to Neural Networks

Linear Models. The simplest parametric model is the linear model, which assumes the relationship between inputs and outputs to be defined by

$$f^\theta(x) = \phi(Wx + b) \tag{2.24}$$

where $\theta = (W, b)$ is a tuple of weight matrix $W \in \mathbb{R}^{d'/C \times d}$ and bias vector $b \in \mathbb{R}^{d'/C}$ (output dimension being d' for regression and C for classification problems). The form of ϕ depends on the nature of the problem. For regression problems, it is simply the identity function (i.e. $\phi(x) = x$ for some argument x) and the resulting

model is called the linear regression model. For classification problems, ϕ is a monotonic function that maps from $\mathbb{R}^C$ to $[0,1]^C$ such that $\sum_{c=1}^{C} f^\theta(\cdot)_c = 1$, so that its outputs can be interpreted as probabilities of a multinomial distribution. One example of ϕ that is commonly used for such models is the softmax function

$$\mathrm{sm}(h)_c = \frac{\exp(h_c)}{\sum_{j=1}^{C} \exp(h_j)}, \tag{2.25}$$

for which the resulting model is called (multinomial) logistic regression model. $\mathrm{sm}(h)_c$ corresponds to the posterior class probability $p(c|x)$ of class c if h_c is modelled to be $\log p(x|c)p(c)$, i.e. the joint log probability of input x and class c. In the linear model this is the case if $(Wx+b)_c \mathrel{\widehat{=}} \log p(x|c)p(c)$.

Neural Networks. In a neural network (NN) model, f is modelled through a composition of simple parametric functions, i.e.

$$f^\theta(x) = f^{\theta_K}(f^{\theta_{K-1}}(\dots f^{\theta_1}(x))) \tag{2.26}$$

The output of the simple functions f^{θ_k} for $k < K$ are called hidden layers, and their components neurons. We focus on fully connected feedforward neural networks (for other types of NNs, see for example Goodfellow et al. (2016)), where each of the simple functions is reminiscent of f^θ in Eq. (2.24), with the difference that ϕ is not constrained to be of a specific form in any but the very last function f^{θ_K}. Instead, they can be freely chosen to introduce adequate non-linearity to the NN as so-called activation functions, for example as the sigmoid function $\phi_{sig}(h)_j = 1/(1+\exp(-h_j))$, or the so called ReLu function $\phi_{ReLu}(h)_j = \max\{h_j, 0\}$. This makes a fully connected feedforward NN a parametric model of the form

$$f^\theta(x) = \phi^K(W^K\phi^{K-1}(W^{K-1}\phi^{K-2}(\dots\phi^1(W^1x+b^1)\dots)+b^{K-1})+b^K) \tag{2.27}$$

with $\theta = (W^K, W^{K-1}, \dots W^1, b^K, b^{K-1}\dots, b^1)$. If ϕ^K is chosen to be the softmax function, its inputs are referred to as logits of the NN outputs (Croce and Hein, 2020). Such a model is visualised in Figure 2.1. The number of hidden layers, together with their dimensionalities and their activation functions ϕ^k, fully specify the form of the

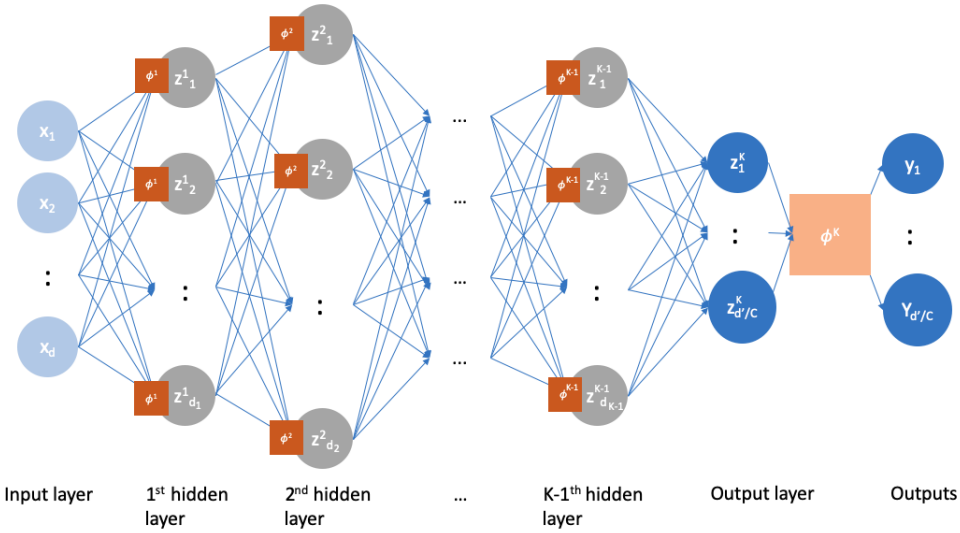


Figure 2.1: Illustration of a fully connected feedforward neural network with $K-1$ hidden layers. The ϕ^K are in colour ordered to highlight that, unlike activation functions of previous layers, the nature of the problem determines its form (e.g. ϕ^K is the identity function for regression). ϕ^k for $k < K$ shown at each individual neuron since they are typically componentwise transformations (Goodfellow et al., 2016).

fully connected feedforward NN model, which in general is called the NN architecture. Which NN architecture is optimal for a given problem is not straightforward to infer and has created a separate research area referred to as *Neural Architecture Search* (Zoph and Le, 2016). Existing approaches to overcome simple trial and error methods for example involve embedding the problem into a reinforcement learning (RL) problem (Zoph and Le, 2016; Pham et al., 2018). We do not go into more detail here, but note that a single-hidden layer NN itself has universal approximation properties if the dimensionality of the hidden layer is large enough (under some mild assumptions on the activation functions ϕ, Hornik et al. (1989)).

Remark 1. *For $K = 1$, a neural network corresponds to a linear (or logistic) model. Therefore, when using the term neural network (NN) we implicitly also refer to linear parametric models. In contrast, we use the term deep neural network (DNN) to explicitly refer to NNs with at least one hidden layer, i.e. $K > 1$.*

Now that we have established the nature of DNNs, let's see how (potentially approximate) Bayesian inference looks for them.

Bayesian Inference for Neural Networks

As before, we start with a simple case of a linear model and move to the more general case of a fully connected feedforward NN afterwards.

Bayesian Linear Regression. Let $\boldsymbol{X} \in \mathbb{R}^{d \times n}$ denote the matrix of observed input points $\{x^{(i)}\}_{1 \leq i \leq n}$ assorted in the columns of $\boldsymbol{X}$, and $\boldsymbol{y} \in \mathbb{R}^n$ the vector of observed scalar outputs of a regression problem (i.e. $d' = 1$ for ease of exposition). Then, combining Eqns. (2.1) and (2.24), we get (under the assumptions that ξ is i.i.d. zero mean Gaussian with variance σ^2)

$$\begin{aligned} p(\mathcal{D}_n|\theta) &= \prod_{i=1}^{n} \mathcal{N}(y_i|f^\theta(x^{(i)}), \sigma^2) \\ &= \prod_{i=1}^{n} \mathcal{N}(y_i|W^T x^{(i)} + b, \sigma^2) \\ &= \mathcal{N}(\boldsymbol{y}|\boldsymbol{X}^T W^+ b, \sigma^2 I) \\ &= \mathcal{N}(\boldsymbol{y}|\bar{\boldsymbol{X}}^T \theta, \sigma^2 I) \end{aligned} \tag{2.28}$$

where we define $\bar{\boldsymbol{X}} = [\boldsymbol{X}^T, \mathbf{1}]^T$ as the augmented version of $\boldsymbol{X}$ to simplify notation for the inference in this linear regression model by directly writing $\theta = [W^T, b]^T$. In order to do Bayesian inference on θ, we need to specify a prior distribution on it. Without further domain knowledge about the specific regression problem at hand, a canonical and practical assumption might be to assume $p(\theta) = \mathcal{N}(\theta|0, \text{diag}(\boldsymbol{a})I)$. Plugging $p(\theta)$ and $p(\mathcal{D}_n|\theta)$ from Eq. (2.28) into Eq. (2.3), we get (Rasmussen and Williams (2006), p. 9)

$$p(\theta|\mathcal{D}_n) = \mathcal{N}\left(\theta \mid \frac{1}{\sigma^2} S^{-1} \bar{\boldsymbol{X}} \boldsymbol{y}, S^{-1}\right) \tag{2.29}$$

with $S = \sigma^{-2} \bar{\boldsymbol{X}} \bar{\boldsymbol{X}}^T + \text{diag}(\boldsymbol{a})^{-1} I$. We thus get a closed form posterior, which is again Gaussian. This is in fact the rare occurrence of conjugate prior and likelihood (both Gaussian), for which the posterior is known in closed form no approximate inference is needed. Other such conjugate pairs are for example Bernoulli likelihood and Beta prior, or Pareto prior and Uniform likelihood. Naturally, we can use

Eq. (2.22), to obtain predictors, such as the mean predictor (Rasmussen and Williams (2006), p. 11)

$$\hat{f}^{(n)}(x^*) = \frac{1}{\sigma^2}\boldsymbol{y}^T\bar{\boldsymbol{X}}^T S^{-1}\bar{x^*} \tag{2.30}$$

and for the predictive variance

$$\mathbb{V}(f^\theta(x^*)) = \bar{x^*}^T S^{-1}\bar{x^*} \tag{2.31}$$

So let's see how this extends to the more general case of multilayer NNs and classification problems.

Bayesian (Deep) Neural Networks. Similarly to the linear case, for Bayesian neural networks (BNNs) (Denker and Lecun, 1991; MacKay, 1992) we start by specifying a prior distribution $p(\theta)$ over the parameters of a neural network, and use it in combination with the likelihood to infer the posterior using Bayes' rule (Eq. (2.3)). Let's consider a classification problem this time, i.e. categorical distributions with class probabilities given by the NN output such that

$$p(\mathcal{D}_n|\theta) = \prod_{i=1}^{n} f^\theta(x^{(i)})_{y^{(i)}}, \tag{2.32}$$

with $f(\cdot)_c$ being the NN output in dimension c. Applying Bayes' formula allows us to derive the posterior distribution over the parameters θ:

$$p(\theta|\mathcal{D}_n) = \frac{\prod_{i=1}^{n} f^\theta(x^{(i)})_{y^{(i)}} p(\theta)}{\int \prod_{i=1}^{n} f^\theta(x^{(i)})_{y^{(i)}} p(\theta) d\theta}. \tag{2.33}$$

Unfortunately, unlike in the Bayesian linear regression case, the posterior distribution $p(\theta|\mathcal{D}_n)$ in Eq. (2.33) is in general analytically intractable for DNNs, meaning that it cannot be written in closed form and as a result sampling and/or predicting from it is not immediately possible. In order to still work with it and use it for example to build the mean prediction function in Eq. (2.22), $p(\theta|\mathcal{D}_n)$ thus needs to be approximated as described in Section 2.1.1. For neural network models, this is often done using maximum likelihood approximations or even more commonly using (implicitly) the MAP approximation - namely whenever weight decay in

the form of an L_2-regularisation term added to the cross-entropy loss is applied. Such a regularisation corresponds exactly to MAP approximation for a zero-mean Gaussian i.i.d. prior (Krogh and Hertz, 1992; MacKay, 1992). In this case, finding the maximum of $p(\theta|\mathcal{D}_n)$ in Eq. (2.33) is equivalent to finding the maximum of its nominator (since the denominator is constant w.r.t. θ) or indeed of the logarithmic transform of its nominator:

$$\log p(\theta|\mathcal{D}_n) \propto \sum_{i=1}^{n} \log f^{\theta}(x^{(i)})_{y^{(i)}} - \frac{1}{2\alpha^2}\|\theta\|_2^2, \tag{2.34}$$

with α^2 being the variance of the Gaussian prior. Maximising Eq. (2.34) is in turn equivalent to minimising

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} - \log f^{\theta}(x^{(i)})_{y^{(i)}} + \frac{1}{2\alpha^2}\|\theta\|_2^2, \tag{2.35}$$

which corresponds to the commonly used cross-entropy loss minimisation with L_2-regularisation used in the deep learning literature (Goodfellow et al., 2016). However, also general function approximations and HMC have been successfully used for approximate inference for BNNs (Blundell et al., 2015; Cobb and Jalaian, 2020). In Chapter 4, where we compare the adversarial robustness of differently approximated BNNs, we will do so both using this MAP approximation and the sampling based approximation using HMC. But first, we move on to introduce Gaussian process models, another important class of Bayesian machine learning models.

2.1.3 Gaussian Processes

In Gaussian process (GP) models, rather than assuming a specific parametric form of the function linking inputs and outputs, this function is assumed to be drawn from a Gaussian process as a prior assumption.

Definition 1 (Gaussian Process)**.** *A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.*

As such, an element of the collection of random variables of a GP can be written as $f(x)$, where x is an element of a (typically infinite) index set $\mathcal{X}$. In the supervised

learning setting we consider, this indexing set is $\mathcal{X} = \mathbb{R}^d$ making the collection of random variables the function values over $\mathbb{R}^d$. A Gaussian process is completely specified by its mean function $m(x)$ and covariance function $k(x, x')$,

$$m(x) = E[f(x)] \tag{2.36}$$

$$k(x, x') = E[(f(x) - m(x))(f(x') - m(x'))], \tag{2.37}$$

as these fully specify the joint Gaussian distribution over any finite set $\{x^{(i)}\}_{i\in\mathcal{I}}$. We thus write $f(x) \sim \mathcal{GP}(m(x), k(x, x'))$ to indicate that f follows a Gaussian process with mean function m and covariance function k. Note that in the literature, the covariance function of a GP is often referred to as its kernel and we shall call it like this as well from now on. Also, if not modelled explicitly using some domain knowledge, $m(x)$ is assumed to be 0. However this prior assumption does not inflict drastic restrictions on the posterior as it does not confine the posterior mean to be 0 as well (Rasmussen and Williams, 2006).

Based on the observations of input output pairs $\mathcal{D}_n$, a posterior stochastic process can be derived and used for prediction using Bayes' rule.

Regression with Gaussian Processes

Based on the assumption on ξ in Eq. (2.1) and the assumption of $f \sim \mathcal{GP}(0, k(x, x'))$, one can derive the posterior distributions of f at the inputs $\{x^{(i)}\}_{1\leq i\in n} \subset \mathcal{D}_n$ using Bayes' rule as

$$\begin{aligned} p(\boldsymbol{f}|\boldsymbol{X}, \boldsymbol{y}) &= \frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{X})}{p(\boldsymbol{y}|\boldsymbol{X})} \\ &= \frac{\mathcal{N}(\boldsymbol{y}|\boldsymbol{f}, \sigma^2 I)\mathcal{N}(\boldsymbol{f}|0, K(\boldsymbol{X}, \boldsymbol{X}))}{\int \mathcal{N}(\boldsymbol{y}|\boldsymbol{f}, \sigma^2 I)\mathcal{N}(\boldsymbol{f}|0, K(\boldsymbol{X}, \boldsymbol{X}))d\boldsymbol{f}} \\ &= \mathcal{N}(\boldsymbol{f}|K(\boldsymbol{X}, \boldsymbol{X})[K(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 I]^{-1}\boldsymbol{y}, [K(\boldsymbol{X}, \boldsymbol{X})^{-1} + \sigma^{-2} I]^{-1}) \end{aligned} \tag{2.38}$$

where we have used the $\boldsymbol{X}$ and $\boldsymbol{y}$ for all inputs and outputs arranged as in Eq. (2.28) and write $K(\boldsymbol{X}, \boldsymbol{X})$ for the matrix with entries $k(x^{(i)}, x^{(j)})$ in entry (i, j). The last equality can be derived by completing the square in the nominator [2].

[2] Alternatively, one can directly use the results derived in Bishop (2006), p.93, where $\boldsymbol{f}$ in Eq. (2.38) corresponds to x in Bishop (2006) (while y stays y)

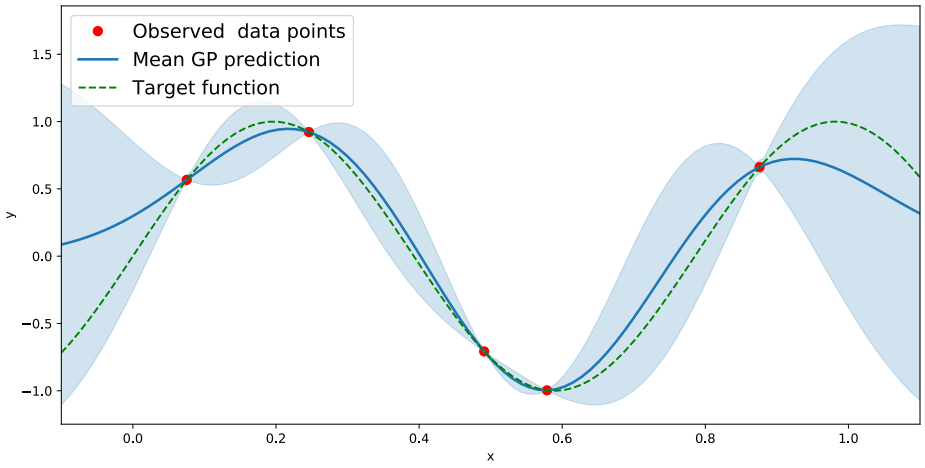


Figure 2.2: Example of the predictive posterior distribution of a Gaussian process, with the light blue area representing the 95% pointwise confidence intervals and the blue line the mean prediction for target function f (green) with $n = 5$ observed data points (red).

From this, for a finite collection of unseen inputs $\boldsymbol{X}^*$, the predictive distribution can be derived again building on the Gaussianity of all distributions involved:

$$\begin{aligned} p(\boldsymbol{f}^*|\boldsymbol{X}^*, \boldsymbol{X}, \boldsymbol{y}) &= \int p(\boldsymbol{f}^*|\boldsymbol{f}) p(\boldsymbol{f}|\boldsymbol{X}, \boldsymbol{y}) d\boldsymbol{f} \\ &= \mathcal{N}(\boldsymbol{f}^*|K(\boldsymbol{X}^*, \boldsymbol{X})[K(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 I]^{-1}\boldsymbol{y}, \\ &\quad K(\boldsymbol{X}^*, \boldsymbol{X}^*) - K(\boldsymbol{X}^*, \boldsymbol{X})[K(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 I]^{-1} K(\boldsymbol{X}, \boldsymbol{X}^*)), \end{aligned} \tag{2.39}$$

A canonical point predictor for a single test point x^* that minimises the mean squared error can thus be obtained by the following GP prediction function:

$$\hat{f}^{(n)}(x^*) = \mathbb{E}[f^*|x^*, \boldsymbol{X}, \boldsymbol{y}] = K(x^*, \boldsymbol{X})[K(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 I]^{-1}\boldsymbol{y} \tag{2.40}$$

However, note that the full closed form predictive distribution in Eq. (2.39) can be used for more purposes than just point predictions, as for example just like BNNs they offer uncertainty quantifications, this time even in the form of computable quantiles and confidence intervals thanks to the known nature of the predictive distribution (Gaussian). Figure 2.2 illustrates that predictive distribution of a GP in the one-dimensional case.

Classification with Gaussian Processes

In the Gaussian process classification (GPC) case, the fundamental assumptions remain the same as in Eq. (2.2), but the class probabilities π_c are not modelled as a parametric function directly, but instead as a composition of a monotonic likelihood function $\sigma : \mathbb{R}^C \to [0,1]^C$ and a *latent* function $f : \mathbb{R}^d \to \mathbb{R}^C$ which is assumed to be a realisation of a Gaussian process with 0 mean. Unfortunately, this results in an absence of conjugacy between likelihood and prior which complicates inference, as

$$\begin{aligned} p(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{y}) &= \frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{X})}{p(\boldsymbol{y}|\boldsymbol{X})} \\ &= \frac{\prod_{i=1}^n \sigma(\boldsymbol{f})_{y^{(i)}}\mathcal{N}(\boldsymbol{f}|0,K(\boldsymbol{X},\boldsymbol{X}))}{\int \prod_{i=1}^n \sigma(\boldsymbol{f})_{y^{(i)}}\mathcal{N}(\boldsymbol{f}|0,K(\boldsymbol{X},\boldsymbol{X}))d\boldsymbol{f}} \end{aligned} \tag{2.41}$$

becomes intractable. We thus need to resort to approximations. Similar to the BNN methods in Section 2.1.2, several approximation methods have been developed to perform GPC inference. The results we obtain in Chapter 3 focus on Gaussian general function approximations, that is, methods that perform approximate inference for $p(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{y})$ by finding a Gaussian distribution $q(\boldsymbol{f};\boldsymbol{m},\boldsymbol{M})$ that resembles $p(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{y})$. The two methods that we use in Chapter 3 to derive q are the *Laplace approximation* and *Expectation Propagation* (see Section 2.1.1). Other methods to obtain a Gaussian q, such as MFVI, have also been used successfully for GPC (Hensman et al., 2015), but our results in Chapter 3 might not hold for these. The key point for our results is that with a Gaussian approximation to $p(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{y}) \approx q(\boldsymbol{f};\boldsymbol{m},\boldsymbol{M})$ the predictive latent function becomes Gaussian of the following form (using the results for Gaussian marginals presented in Bishop (2006), p. 93):

$$\begin{aligned} q(f^*|x^*,\boldsymbol{X},\boldsymbol{y}) &= \int p(f^*|x^*,\boldsymbol{f},\boldsymbol{X})q(\boldsymbol{f};\boldsymbol{m},\boldsymbol{M})d\boldsymbol{f} \\ &= \int \mathcal{N}(f^*|\boldsymbol{k}_*^T\boldsymbol{K}^{-1}\boldsymbol{f}, k_{**}-\boldsymbol{k}_*^T\boldsymbol{K}^{-1}\boldsymbol{k}_*)\mathcal{N}(\boldsymbol{f}|\boldsymbol{m},\boldsymbol{M})d\boldsymbol{f} \\ &= \mathcal{N}(f^*|\boldsymbol{k}_*^T\boldsymbol{K}^{-1}\boldsymbol{m}, k_{**}-\boldsymbol{k}_*^T[\boldsymbol{K}^{-1}-\boldsymbol{K}^{-1}\boldsymbol{M}\boldsymbol{K}^{-1}]\boldsymbol{k}_*), \end{aligned} \tag{2.42}$$

where we follow the shortened notation of Rasmussen and Williams (2006), using $k_{**} = k(x^*, x^*)$, $\boldsymbol{k}_* = K(\boldsymbol{X}, x^*)$ and $\boldsymbol{K} = K(\boldsymbol{X}, \boldsymbol{X})$ [3]. Predictions of the class probabilities π_c are then obtained by marginalising π_c over $q(f^*)$, giving the following GPC prediction function:

$$\hat{f}^{(n)}(x^*)_c = \pi(x^*)_c = \int \sigma(f^*)_c q(f^*|x^*, \boldsymbol{X}, \boldsymbol{y}) df^*. \tag{2.43}$$

For most likelihood functions, this integral becomes intractable even for Gaussian approximations q, so one has to resort to Monte Carlo estimates to obtain $\hat{f}^{(n)}(x^*)_c$ in practice. To this end, L samples $f^{*(l)}$ are drawn from q and $\hat{f}^{(n)}(x^*)_c = \frac{1}{L}\sum_{l=1}^{L} \sigma(f^{*(l)})_c$ is used as approximation to the intractable integral in Eq. (2.43). Before we move on to establish the well-known connections between GP models and BNNs (including linear models), let's briefly discuss one crucial element of GP models: the kernel.

Choosing Kernel Functions

As we have seen in Eqns. (2.40) and (2.42), the predictions of a GP regression model depend directly on the chosen kernel function k. Let us therefore briefly discuss which functions can be used as kernel function, as not any function can be used as such. In particular, in order to make the resulting prior over inputs $\mathcal{N}(\boldsymbol{f}|0, K(\boldsymbol{X}, \boldsymbol{X}))$ a well defined probability distribution, $K(\boldsymbol{X}, \boldsymbol{X})$ needs to be symmetric positive semi-definite. This can be guaranteed by imposing the following two conditions on k:

Condition 1: First, a kernel needs to be symmetric, that is $k(x, x') = k(x', x)$ because obviously $\text{cov}(f(x), f(x')) = \text{cov}(f(x'), f(x))$.

Condition 2: Second, a kernel needs to be positive semidefinite, that is it needs to fulfil

$$\int k(x, x') g(x) g(x') d\mu(x) d\mu(x') \geq 0, \tag{2.44}$$

[3] While many of our results hold for any such Gaussian approximation, our proposed bounding of the predictive variance requires $\boldsymbol{K}^{-1} - \boldsymbol{K}^{-1}\boldsymbol{M}\boldsymbol{K}^{-1}$ to be positive semidefinite, which is the case for Laplace and EP approximations Rasmussen and Williams (2006)

for all $g \in L_2(\mathcal{X},\ \mu)$ where μ is a measure on $\mathcal{X}$. This ensures that for any finite collection of inputs, the covariance matrix of the joint distribution of their outputs $K(\boldsymbol{X}, \boldsymbol{X})$ is positive semidefinite.

There are obviously infinitely many functions that fulfil these conditions (as for example scaling a kernel by any scalar $\alpha > 0$ gives a distinct, yet also valid kernel). Choosing the right kernel is thus a non-trivial task.

For regression problems, one handy implication of the tractability of the integrals in Eq. (2.38) is that Bayesian model selection can be used, as we can compute the model likelihood, also called marginal likelihood (Rasmussen and Williams, 2006), as

$$\begin{aligned} p(\mathcal{D}_n|\mathcal{M}) &= p(\boldsymbol{y}|\boldsymbol{X}, \mathcal{M}) = \int \mathcal{N}(\boldsymbol{y}|\boldsymbol{f}, \sigma^2 I)\mathcal{N}(\boldsymbol{f}|0, K_{\mathcal{M}}(\boldsymbol{X}, \boldsymbol{X}))d\boldsymbol{f} \quad (2.45) \\ &= \mathcal{N}(\boldsymbol{y}|0, K_{\mathcal{M}}(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 I) \\ &= (2\pi)^{-\frac{n}{2}}|K_{\mathcal{M}}(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 I|^{-\frac{1}{2}}\mathrm{e}^{-\frac{1}{2}\boldsymbol{y}^T[K_{\mathcal{M}}(\boldsymbol{X},\boldsymbol{X})+\sigma^2 I]^{-1}\boldsymbol{y}} \end{aligned}$$

where the model $\mathcal{M}$ in this case is defined by its kernel $k_{\mathcal{M}}$. While full Bayesian inference would strictly require using the model posterior $p(\mathcal{M}|\mathcal{D}_n)$ of all models under consideration and averaging over them for prediction, a more practical approach is often to use the MAP approximation of the model posterior for prediction, similarly to what we have seen for BNN parameters. In lack of domain knowledge that allows to specify a better prior, the prior over different models is assumed to be uniform, which turns model MAP into model MLE, i.e. maximisation of Eq. (2.45) with respect to $k_{\mathcal{M}}$. Note however that, unlike MLE and MAP in the parametric case, uncertainty regarding f prevails in the posterior and its predictions (see Eq. (2.39)) even when such Dirac delta function approximations have been used for model selection.

For classification tasks, choosing the right kernel function can be done in a similar way, but the resulting approximation for $p(\mathcal{D}_n|\mathcal{M})$ has to be used instead:

$$p(\mathcal{D}_n|\mathcal{M}) \approx q(\boldsymbol{y}|\boldsymbol{X}, k_{\mathcal{M}}) \quad (2.46)$$

The form of $q(\boldsymbol{y}|\boldsymbol{X}, k_{\mathcal{M}})$ as a function of $k_{\mathcal{M}}$ depends on the specific type of approximation used. For example, for *Laplace* and *EP*, they can be found in

Rasmussen and Williams (2006).

In the following, we present those classes of kernels which we consider in Chapters 3 and 5 (for a detailed presentation see Rasmussen and Williams (2006)). Choosing the so-called hyperparameters of these kernel classes is part of the type-II-maximum likelihood optimisation and is typically done using gradient methods. Note however that since the matrix inversion in Eq. (2.45) requires $\mathcal{O}(n^3)$ elementary operations, choosing the right kernel incl. its hyperparameters ('training the GP') is very costly for large data sets. There have however been made advances in the field of *sparse Gaussian processes* that achieve a significant reduction of this computational burden. For example, Titsias (2009) and Hensman et al. (2015) achieve to reduce the complexity to $\mathcal{O}(nm^2)$ and $\mathcal{O}(m^3)$ by the use of $m < n$ so called inducing points on which the posterior is conditioned on (rather than on the n actually observed training points).

Squared Exponential Kernels. Squared exponential (SE) kernels have the form

$$k_{SE}(x, x') = \exp\left(-0.5(x - x')^T \sum_l (x - x')\right), \tag{2.47}$$

where $\sum_l = \text{diag}(l_1^{-2}, l_2^{-2}, \ldots, l_d^{-2})$ and $l_i > 0$ for $1 \leq i \leq d$. A function that is a realisation of a GP with such a kernel is infinitely differentiable (Rasmussen and Williams, 2006). Its variation can be controlled in the standard case $l := l_1 = l_2 = \ldots = l_d$ through the so called length-scale l. Large values for l will be good at accounting for highly fluctuating data, while small values for l are convenient for data where closely located inputs do have similar outputs. In the extended case where the l_i's can differ from each other a characteristic length-scale is assigned to each dimension, making it possible to adjust the influence of each input factor: with a very large characteristic length-scale for the i-th dimension, the kernel will almost become independent of the i-th input factor. As this is the only way through which the inputs are related to the output in the GP, such an input factor is basically removed from inference. This feature is called automatic relevance determination

(ARD, Neal (1995)).

Matérn Kernels. If we denote $D := (x - x')^T \sum_l (x - x')$ with $\sum_l$ defined as for the SE kernels, the Matérn class of kernels is defined by

$$k_{\text{Matérn}}(x, x') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu D}\right)^{\nu} K_{\nu} \left(\sqrt{2\nu D}\right), \tag{2.48}$$

where $\nu > 0$ and K_{ν} is a modified Bessel function. For $\nu \to \infty$ the Matérn kernel asymptotically approaches the SE kernel. Since this convergence happens rather quick, we focus on only the two cases of $\nu = 5/2$, $\nu = 3/2$. For these explicit kernel representations can be found as:

$$\begin{aligned} k_{\nu=5/2}(x, x') &= \left(1 + \sqrt{5D} + \frac{5}{3}D\right) \exp\left(-\sqrt{5D}\right) \\ k_{\nu=3/2}(x, x') &= \left(1 + \sqrt{3D}\right) \exp\left(-\sqrt{3D}\right) \end{aligned} \tag{2.49}$$

Using a Matérn kernel yields a GP which is p-times mean square differentiable[4] if and only if $\nu > p$, thereby making the above Matérn kernels less smooth than SE kernels. Again, forcing the diagonal elements of $\sum_l$ to be identical yields in a shared length-scale across dimensions which allows to control the variation of realised functions whereas allowing them to differ enables ARD and automatic feature selection.

Note that both the SE and the Matérn kernels are functions of $|x - x'|$, and as such could be written as $k(\tau)$ with $\tau = x - x'$. Such kernels are called stationary kernels and have (among other interesting properties) the property that $k(x, x) = k(x', x') = k(0, 0)$, i.e. that the pointwise prior variance they induce is input independent. There are however also non-stationary kernels, as for example the dot product kernels presented below.

[4]For a definition and criterion of mean square differentiability see Rasmussen and Williams (2006)

Dot Product Kernels. The class of dot product kernels is characterised by kernels of the form

$$k_{dot}(x, x') = x^T \sum_l x' + \sigma_0^2, \tag{2.50}$$

where $\sum_l$ is an arbitrary positive semidefinite matrix. In the case where $\sum_l$ is diagonal the kernel is called linear kernel because, as we will see in the next paragraph, it yields exactly the same covariance and prediction function as obtained by Bayesian linear regression. Linear kernels can again exhibit ARD if the elements of the diagonal are not forced to be identical.

Now that we have introduced the kernel function, we can proceed to highlight established connections between GPs and the Bayesian parametric models introduced in Section 2.1.2.

2.1.4 Relationship between Bayesian Neural Networks and Gaussian Processes

Interestingly, while at first glance, the non-parametric nature of GPs seems to make them entirely different from the parametric models introduced in Section 2.1.2, various connections and equivalences between the two have been established. We present some fundamental ones below.

Bayesian Linear Regression as Gaussian Process

From the posterior over the weights in a Bayesian linear regression model as given in Eq. (2.29) we can derive the following predictive distribution for a new test input x^* for a Bayesian linear regression model without intercept (i.e. $b = 0$) and a prior variance of 1 for the weights (i.e. $\boldsymbol{a} = \mathbf{1}$):

$$p(f^*|x^*, \mathcal{D}_n) = \mathcal{N}\left(f^*|x^{*T} S^{-1} \boldsymbol{X}\boldsymbol{y}, x^{*T} S^{-1} x^*\right) \tag{2.51}$$

with $S = \boldsymbol{X}\boldsymbol{X}^T + \sigma^2 I$. Compare this to the predictive distribution of a GP regression model with linear kernel with $\sum_p = I$ and $\sigma_0^2 = 0$. Combining Eqns.

(2.39) and (2.50) we get:

$$p(f^*|x^*, \mathcal{D}_n) = \mathcal{N}(f^*|x^{*T}\boldsymbol{X}[\boldsymbol{X}^T\boldsymbol{X} + \sigma^2 I]^{-1}\boldsymbol{y}, \quad (2.52)$$
$$x^{*T}x^* - x^{*T}\boldsymbol{X}[\boldsymbol{X}^T\boldsymbol{X} + \sigma^2 I]^{-1}\boldsymbol{X}^T x^*)$$

With a bit of algebra, it can be seen that $\boldsymbol{X}[\boldsymbol{X}^T\boldsymbol{X} + \sigma^2 I]^{-1} = [\boldsymbol{X}\boldsymbol{X}^T + \sigma^2 I]^{-1}\boldsymbol{X}$, as well as $x^{*T}x^* - x^{*T}\boldsymbol{X}[\boldsymbol{X}^T\boldsymbol{X} + \sigma^2 I]^{-1}\boldsymbol{X}^T x^* = \sigma^2 x^{*T}[\boldsymbol{X}\boldsymbol{X}^T + \sigma^2 I]^{-1}x^*$, i.e. the predictive means are identical and the predictive variances are identical up to the scaling factor σ^2. This means that for $\sigma^2 = 1$ we get the exact same posterior predictive distribution from a GP regression model with linear kernel as we get from a Bayesian linear regression model without intercept and unit variance i.i.d. priors on the weights. As we see next, similar parallels can be established between GPs and BNNs.

Bayesian Neural Networks as Gaussian Processes

A similar connection can be established for deep Bayesian neural networks and GPs. Most notably, as is shown in Neal (1995), a BNN of the form of a one hidden-layer converges to a GP as the number of units in the hidden layer goes to infinity, if some mild assumptions about activation functions ϕ and prior distributions $p(\theta)$ are met, namely

1. ϕ^1, the element-wise activation function for the hidden layer, is bounded, and ϕ^2, the activation function for the output unit f^θ, is simply the identity function.

2. b^2 and the elements of W^2 have independent zero-mean prior distributions with variances σ_b^2 and σ_w^2 respectively.

3. The weight vectors W_j^1 have independent and identical prior distributions.

If these are fulfilled, we have for

$$f^\theta(x) = b^2 + \sum_{j=1}^{d_1} W_j^2 \phi^1(W_j^1 x + b_j^1), \quad (2.53)$$

with d_1 being the number of hidden units,

$$\begin{aligned} \mathbb{E}[f^\theta(x)] &= 0 \\ \mathbb{E}[f^\theta(x)f^\theta(x')] &= \sigma_b^2 + \sum_{j=1}^{d_1} \sigma_w^2 \mathbb{E}[\phi^1(W_j^1 x + b_j^1)\phi^1(W_j^1 x' + b_j^1)] \\ &= \sigma_b^2 + d_1 \sigma_w^2 \mathbb{E}[\phi^1(W_0^1 x + b_0^1)\phi^1(W_0^1 x' + b_0^1)]. \end{aligned} \tag{2.54}$$

Letting σ_w^2 scale as $\frac{c}{d_1}$ for some constant c keeps the covariance finite even for $d_1 \to \infty$. As ϕ^1 is assumed to be bounded, all moments of $p(\phi^1(W_j^1 x + b_j^1))$ will be bounded and thus the central limit theorem can be applied to the sum of i.i.d. random variables in Eq. (2.53). Letting $d_1 \to \infty$ thus lets the prior distribution over $f^\theta(x)$ converge to a GP with mean and covariance functions given as in Eqns. (2.54). For example, if the activation function is chosen to be the error function, i.e. $\phi^1(v) = 2/\sqrt{(\pi)} \int_0^v \exp(-t^2)dt$, and $[W_j^1, b_j^1] \overset{i.i.d}{\sim} \mathcal{N}(0, \Sigma)$ for all j, then in Neal (1995) it is derived that from Eq. (2.54) we get as covariance function

$$k_{NN}(x, x') = \frac{2}{\pi} \sin^{-1}\left(\frac{2\overline{x}^T \Sigma \overline{x}'}{\sqrt{(1 + 2\overline{x}^T \Sigma \overline{x})(1 + 2\overline{x}'^T \Sigma \overline{x}')}} \right), \tag{2.55}$$

with $\overline{x} = (1, x_1, \ldots, x_d)^T$ being the augmented input vector as before. More recently, several works have shown similar correspondences also for deeper neural networks and different activation functions, always in the limit of infinitely wide layers. We point to for example Lee et al. (2017), Novak et al. (2018), and Garriga-Alonso et al. (2018) for further reading. Naturally, the existing connections between GPs and BNNs bring up the question whether their adversarial robustness is similar, a question which we will address (among others) through Chapters 3 and 4.

2.1.5 Other Bayesian Machine Learning Models

The work in this thesis focusses on the two types of Bayesian models introduced in the previous subsections of this chapter, BNNs and GPs. We do note however that there are several other parametric and non-parametric Bayesian models for supervised learning problems, such as for example Bayesian tree based models (Chipman et al., 2010), the Naïve Bayes classifier (Rish et al., 2001), or hierarchical

Bayesian models and Dirichlet processes (Teh, 2010). A more detailed overview can be found for example in Gelman et al. (2013).

2.2 Adversarial Attacks and Adversarial Robustness

Now that we have introduced all relevant concepts of Bayesian learning in supervised problems, we can proceed to the second part of this background chapter and introduce adversarial attacks. The term adversarial attack is used to refer to an algorithm whose goal is to create an adversarial example. Such adversarial examples can broadly be described as *"inputs to machine learning models that an attacker intentionally designed to cause the model to make mistakes"* (Yang et al., 2019). A successful attack then refers to a successful run of an adversarial attack algorithm, i.e. to the case in which a given run of the algorithm results in an input on which the attacked model does indeed make a mistake. While in some cases this can be interesting for regression problems (Nguyen and Raff, 2018), most commonly, the model mistake caused by an attack is understood to be a misclassification in classification problems (Croce and Hein, 2020; Madry et al., 2017; Szegedy et al., 2013). Depending on the application and the attacked model, there exists a wide range of types of adversarial attacks. We here focus on the ones which are relevant for the subsequent chapters. Adversarial attacks are most problematic in safety critical applications such as autonomous driving, where they have been successfully used to cause erroneous lane switching (Tencent Keen Security Lab, 2019), or medical applications, where they can cause wrong diagnoses (Finlayson et al., 2019). The types of attacks relevant in these scenarios are typically evasion attacks, that is attacks that operate on a fixed trained model, rather than poisoning attacks, which aim to cause model mistakes by manipulating its training data (e.g. in applications where web data is used, Xu et al. (2020)). Furthermore, in such applications, the created adversarial example should be imperceptible from a regular input to a human at first glance, since for safety critical applications one would assume at least some high-level interaction with humans in otherwise automated decision making

processes (e.g. a human in the driving seat of an autonomous vehicle being able to manually break in case he/she spots something weird or a doctor glancing over inputs to ML based diagnosis pipelines). That means we restrict the adversarial attacks we consider in this thesis to attacks near regular data inputs, which is usually done by constraining its difference in norm with respect to a reference data point. This yields the following generic restricted evasion attack formulation:

Definition 2 (Restricted Evasion Attack)**.** *Consider a fixed classification model with prediction function* $\hat{y}(\cdot) = \arg\max_c \hat{f}^{(n)}(\cdot)_c$*. For a given test point* x^* *with true class label* y^**, the objective of a restricted evasion attack with perturbation budget* γ *is defined as*

$$\begin{aligned} &\textit{Find} \quad \textbf{adversarial example} \quad x \\ &\textit{s.t.} \quad \|x - x^*\| < \gamma, \end{aligned} \tag{2.56}$$

where the specific type of **adversarial example** can be defined depending on the context (see for example Eqns. (2.57) and (2.58) below). Typical metrics are L_0-, L_2- or $L_\infty-$norm induced metrics on $\mathbb{R}^d$. For example, Figure 2.3 shows an adversarial example with respect to the $L_\infty-$norm, found to a panda image which as a result is classified as a gibbon. While our focus is therefore narrowed down significantly, there still exists a range of subtypes of restricted evasion attacks, which we exhibit in the following. Note that unless stated otherwise, we will use the term adversarial attack and restricted evasion attack interchangeably henceforth.

2.2.1 Archetypes of Adversarial Attacks

Among evasion attacks, we can classify different subtypes along two dimensions: the goal of the attack (targeted vs. untargeted attack) and the knowledge of the attack (white-box vs. black-box attack).

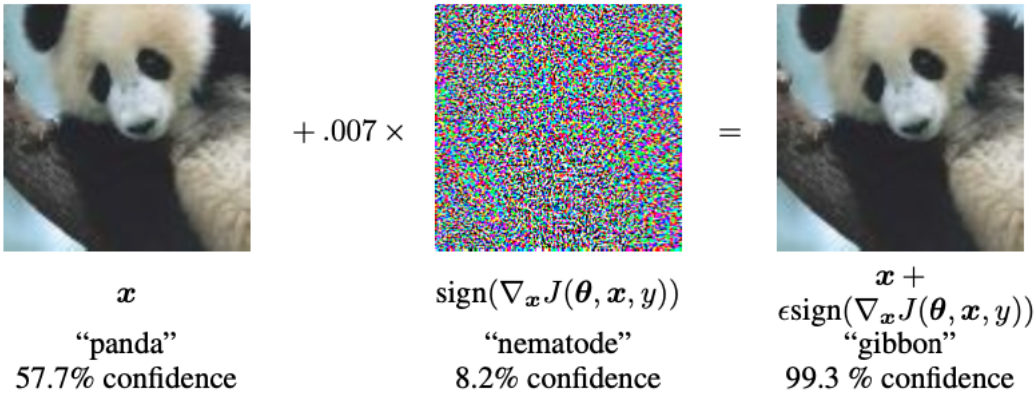


Figure 2.3: Example of a Restricted Evasion Attack with respect to the $L_\infty-$norm taken from Goodfellow et al. (2014). The originally correctly classified image of a panda (left) gets perturbed by a low magnitude (middle) to give an adversarial example whose difference to the original image is imperceptible, yet causes the attacked model to classify it as a gibbon (right).

Targeted vs. Untargeted Adversarial Attacks

While in an untargeted attack, the goal is to cause any type of misclassification, the goal of targeted attacks is to cause the attacked model to output a specific, predetermined class. By definition, this means that any successful targeted attack is also a successful untargeted attack, making targeted attacks a harder problem. Translating this into concrete objectives for Definition 2 in place of Eq. (2.56), we get the following objective for an untargeted attack

$$\begin{aligned} \text{Find} \quad x \quad \text{with} \quad & \arg\max_c \hat{f}^{(n)}(x)_c \neq y^* \\ & \text{s.t.} \quad \|x - x^*\| < \gamma, \end{aligned} \tag{2.57}$$

and for a targeted attack with target class $\tilde{c} \neq y^*$, we get

$$\begin{aligned} \text{Find} \quad x \quad \text{with} \quad & \arg\max_c \hat{f}^{(n)}(x)_c = \tilde{c} \\ & \text{s.t.} \quad \|x - x^*\| < \gamma. \end{aligned} \tag{2.58}$$

White-Box vs. Black-Box Adversarial Attacks

In the white-box setting, the adversarial attack has access to all information about the attacked model. In particular, this includes gradient information, which can be used to find an adversarial example using gradient descent based algorithms (see

Section 2.2.2). On the other end of the spectrum lies the black-box setting, which describes the case in which initially the attacking algorithm has no information about the attacked model at all. The only way that such information can be generated is by querying the attacked model and observing its outputs. In the hard black-box setting this means only observing the class labels, while in the soft black-box setting this means observing the class probabilities that the model assigns (i.e. $\hat{f}^{(n)}(x)$). This means the attack difficulty level is increasing from white-box to soft to hard black-box setting. While there exist grey-box types of settings in between the settings described here, they are not a focus of the work in this thesis.

2.2.2 Implementations of Adversarial Attacks

The goal of an adversarial attack, depending on its type, has now been established. Starting from these common goals, there have been a multitude of works proposing different adversarial attacks in order to achieve these goals. We will not go into all of them and into every detail but again highlight some of the main types of approaches in terms of optimisation procedures.

Optimisation Objectives

In particular for untargeted attacks, it is not clear what the best way to translate Eq. (2.57) into a concrete optimisation objective is. The most common one (Croce and Hein, 2020) is to find an adversarial example δ^* by maximising the cross entropy loss, turning Eq. (2.57) into the following optimisation objective:

$$\begin{aligned} \delta^* &= \arg\max_{\delta} \quad \mathcal{L}^{CE}_{\hat{f}^{(n)}}(x^* + \delta, y^*) \quad \text{s.t.} \quad \|\delta\| \leq \gamma \\ &= \arg\max_{\delta} \; -\log \hat{f}^{(n)}(x^* + \delta)_{y^*} \quad \text{s.t.} \quad \|\delta\| \leq \gamma \end{aligned} \tag{2.59}$$

For NN models specifically, another proposed objective is based on maximising the Carlini and Wagner (CW) loss (Carlini and Wagner, 2017b), which is

$$\begin{aligned} \delta^* &= \arg\max_{\delta} \quad \mathcal{L}^{CW}_{\hat{f}^{(n)}}(x^* + \delta, y^*) \quad \text{s.t.} \quad \|\delta\| \leq \gamma \\ &= \arg\max_{\delta} \; -\phi^{K-1}(\hat{f}^{(n)}(x^* + \delta))_{y^*} + \max_{i \neq y^*} \phi^{K-1}(\hat{f}^{(n)}(x^* + \delta))_i \quad \text{s.t.} \quad \|\delta\| \leq \gamma \end{aligned} \tag{2.60}$$

It is sometimes referred to as the margin loss (Xu et al., 2020) since it directly minimises the margin. Lastly Croce and Hein (2020) propose to maximise a scaling invariant version of the CW loss called the Difference in Logits Ratio (DLR) which aims to find an adversarial example for NN models as

$$\begin{aligned}\delta^* &= \arg\max_{\delta} \quad \mathcal{L}^{DLR}_{\hat{f}^{(n)}}(x^* + \delta, y^*) \qquad \text{s.t.} \quad \|\delta\| \leq \gamma \\ &= \arg\max_{\delta} \quad \frac{-\phi^{K-1}(\hat{f}^{(n)}(x^* + \delta))_{y^*} + \max_{i \neq y^*} \phi^{K-1}(\hat{f}^{(n)}(x^* + \delta))_i}{-\phi^{K-1}(\hat{f}^{(n)}(x^* + \delta))_{r(1)} + \phi^{K-1}(\hat{f}^{(n)}(x^* + \delta))_{r(3)}} \quad \text{s.t.} \quad \|\delta\| \leq \gamma\end{aligned} \tag{2.61}$$

where $r(n)$ denotes the $n-$th largest entry of $\hat{f}^{(n)}(\cdot)$.

For targeted attacks with target class $\tilde{c}$, a straightforward adjustment of Eq. (2.60) consists in replacing $\max_{i \neq y^*} \phi^{K-1}(\hat{f}^{(n)}(x^* + \delta))_i$ by $\phi^{K-1}(\hat{f}^{(n)}(x^* + \delta))_{\tilde{c}}$. The same is done for the targeted DLR loss in the nominator in Eq. (2.61), but additionally $\phi^{K-1}(\hat{f}^{(n)}(x^* + \delta))_{r(3)}$ is replaced by $[\phi^{K-1}(\hat{f}^{(n)}(x^* + \delta))_{r(3)} + \phi^{K-1}(\hat{f}^{(n)}(x^* + \delta))_{r(4)}]/2$ in the denominator (to avoid a constant loss when $\tilde{c} = r(3)$). For the CE loss, the loss gets multiplied by -1 (turning maximisation into minimisation) with $\tilde{c}$ replacing y^* in Eq. (2.59).

Optimisation Approaches

There are several approaches to tackle the optimisation problems in Eqns. (2.59 - 2.61) (or similar variants of adversarial attack objectives). Naturally, they will differ for white-box and black-box settings, based on the differences in available information.

White-Box Approaches. Most white-box approaches use the gradient information that is available from the model to find the adversarial perturbation δ^* that maximises Eqns. (2.59 - 2.61). We below review the two approaches which are relevant for Chapters 3 and 4.

Definition 3 (Fast Gradient Sign Methods Attack (FGSM))**.** *Given the prediction function $\hat{f}^{(n)}$ of a classification model and a test point x^* with label y^*, a fast gradient sign method (FGSM) attack with perturbation level γ is defined as*

$$\delta^{adv} = \gamma \, \text{sign}(\nabla_x \mathcal{L}^{(\cdot)}_{\hat{f}^{(n)}}(x^*, y^*)) \tag{2.62}$$

resulting in

$$x^{adv}_{FGSM} = x^* + \gamma \operatorname{sign}(\nabla_x \mathcal{L}^{(\cdot)}_{\hat{f}^{(n)}}(x^*, y^*)) \tag{2.63}$$

FGSM was proposed by Goodfellow et al. (2014) as a computationally cheap alternative to earlier attacks for the case where the perturbation restricting norm is the $L_\infty-$norm. It only requires one gradient evaluation, one (element-wise) multiplication, one (element-wise) addition, and one (element-wise) evaluation of the sign function. Clearly, FGSM is bound to work well if the attacked model behaves linearly around x^*, as taking a maximal step in the gradient direction (onto the perturbation boundary) will cause the highest attainable loss for a linear model. Figure 2.3 actually displays an adversarial example that was found by an FGSM attack in Goodfellow et al. (2014).

Yet, in the case where the model is not fully linear and/or gradients are obfuscated (e.g. due to randomisation or scaling of logits), a stronger attack might be necessary to find an adversarial example. The most popular white-box attack for this purpose is the following projected gradient descent attack introduced by Madry et al. (2017):

Definition 4 (Projected Gradient Descent Attack (PGD))**.** *Given a perturbation level γ, an L_p-norm and the prediction function $\hat{f}^{(n)}$ of a classification model and a test point x^* with label y^*, a projected gradient descent attack on $\hat{f}^{(n)}$ with K iterations (PGD-K) for p at γ is defined via*

$$\delta^0 = 0 \tag{2.64}$$

$$\delta^{k+1} = \prod\nolimits_{\mathcal{B}^p_\gamma(x^*)}[x^* + \delta^k + \alpha \operatorname{sign}(\nabla_x \mathcal{L}^{(\cdot)}_{\hat{f}^{(n)}}(x^* + \delta^k, y^*))] - x^* \tag{2.65}$$

with step-size hyperparameter $\alpha > 0$, resulting in

$$x^{adv}_{PGD} = x^* + \delta^{(K)} \tag{2.66}$$

Here, $\prod_{\mathcal{B}^p_\gamma(x^*)}[\cdot] : \mathbb{R}^d \to \mathcal{B}^p_\gamma(x^*)$ is a projection function that projects inputs back to the γ-ball around x^*. For example, for $p = \infty$ this is typically simply a component-wise clipping function $\prod_{\mathcal{B}^p_\gamma(x^*)}[x]_i = \max(\min(x_i, x^*_i + \gamma), x^*_i - \gamma)$.

Typically, a larger maximum iteration value K yields a stronger attack, however at the price of increased computational cost. PGD is considered to be one of the strongest white-box attacks. Even the most recent state-of-the-art adversarial white box attack, AutoAttack (Croce and Hein, 2020), relies on PGD at its core, combining it with further attacks and automating the selection of step-size α in an dynamically adjusting step-size schedule.

Black-Box Approaches. In the blackbox-setting, gradient information is not readily available. Some approaches aim to approximate gradients using finite difference methods, with the obvious drawback that doing so requires a lot of queries (in high-dimensional settings in particular). Other approaches employ surrogate models and yet other works employ gradient-free optimisation. We give a more detailed overview over competing black-box approaches in Chapter 5, where we propose a query efficient surrogate model method based on Bayesian optimisation.

2.2.3 Measuring Adversarial Robustness and Evaluating Adversarial Attacks

Measuring Adversarial Robustness

The robustness of a model with given prediction function $\hat{f}^{(n)}$ against adversarial attacks can be analysed both locally and globally. On a local level, the (local) maximal safety radius can be used (Croce et al., 2019):

Definition 5 (Local maximal safety radius (LMSR))**.** *Let $\hat{f}^{(n)}$ be the prediction function of a classification model and x^* be a data point with true label y^*. The maximal safety radius for (x^*, y^*) w.r.t. an L_p-norm is defined as*

$$LMSR_p(x^*, y^*) := \min_{\delta \in \mathbb{R}^d} \|\delta\|_p \quad s.t. \quad \max_{c \neq y^*} \hat{f}^{(n)}(x^* + \delta)_c \geq \hat{f}^{(n)}(x^* + \delta)_{y^*} \tag{2.67}$$

If the local maximal safety radius for a given point x^* is $\gamma > 0$, then all points inside the $\gamma-$ball around x^* (excluding the boundary) are classified as y^* if the canonical $0-1$ loss is used, and as a result no adversarial attack could find adversarial examples inside that ball. The larger the local maximal safety radius is, the more

locally robust against adversarial attacks is the model.

Extensions to global robustness are conceptually and computationally not straightforward. Taking the minimum of local maximal safety radii across test points always leads to a value of 0 if the model has less than 100% test accuracy. Taking the average could give a better global picture, but can be influenced by outliers. In either way, computing the local maximal safety radius exactly is NP-hard already for single test points for many non-linear models such as deeper NNs or GPC models (Ruan et al., 2018; Blaas et al., 2020). Getting to a global robustness measure based on maximal safety radii is thus generally prohibitively expensive and is as such often approximated, e.g. by using an empirical analysis of the robustness against adversarial attacks. We give more details on such quantifications in Chapters 3 and 4.

Evaluating Adversarial Attacks

How we judge the quality of an adversarial attack depends slightly on the setting. In the white-box setting, the main criterion is whether it manages to generate adversarial examples successfully. Strictly speaking, this should be assessed against the possibility of finding an adversarial example, i.e. only evaluated if the LMSR in Eq. (2.67) is smaller than the allowed perturbation budget γ. However, since computing the LMSR is typically prohibitively expensive, the attack success is instead usually evaluated plainly, even if no adversarial example exists for the given perturbation budget. As a result, on a global scale, the quality of an adversarial white-box attack is typically measured by the attack success rate (ASR), which is the fraction of test points that the attack found an adversarial example for. Additional secondary criteria for the white-box setting can be the runtime of the attack as well as the proximity to the attacked test point(s), i.e. the magnitude of the perturbation δ (again in an idealised world this would be compared against the LMSR).

For the black-box setting, the primary criterion is still the attack success rate. However, since in a real-life application, each query of the attacked model has an associated cost (computational, monetary, risk of detection, etc.), another

criterion becomes important as well: the number of queries necessary to generate the adversarial example. These two criteria can be globally combined by analysing the ASR as a function of the number of model queries (see Chapter 5). Again, further secondary criteria can be the runtime of the attack as well as the the magnitude of the perturbation δ.

3

Adversarial Robustness Guarantees for Gaussian Process Classification Models

3.1 Introduction

As our first contribution, we analyse the adversarial robustness of GP classification (GPC) models in this chapter. In particular, given a trained GPC model and a compact subset of the input space $T \subseteq \mathbb{R}^d$, we pose the problem of computing the maximum and minimum of the GPC class probabilities over all $x \in T$. We show that such values naturally allow us to compute robustness properties such as the LMSR introduced in Definition 5 as well as others previously employed for analysis of deep learning models (Ruan et al., 2018). Unfortunately, exact direct computation of the maximum and minimum class probabilities over compact sets is not possible, as these would require providing an *exact solution* of a global non-linear optimisation problem, for which no general method exists (Neumaier, 2004). We show how upper and lower bounds for the maximum and minimum classification probabilities of GPCs can be computed on any given compact set T, and then iteratively refine these bounds in a branch and bound algorithmic scheme until convergence to the minimum and maximum is obtained.

Specifically, through discretisation of the GPC latent space, we derive an upper and lower bound on the GPC class confidence output by analytically optimising

a set of Gaussian integrals, whose parameters depend upon extrema of the GPC latent posterior mean and variance in T. We show how the latter can be bounded by solving a set of convex quadratic and linear programming problems, for which solvers are readily available (Boyd and Vandenberghe, 2004). Finally, for any given error tolerance $\epsilon > 0$, we prove that there exists a discretisation of the latent space that ensures convergence of the branch and bound to values ϵ-close to the actual maximum and minimum class probabilities in finitely many steps. The method we propose is anytime (the bounds provided are at every step an over-estimation of the actual classification ranges over T, and can hence be used to provide guarantees) and ϵ-exact (the actual values are retrieved in finitely many steps up to an error ϵ selected a-priori).

We apply our approach to conduct adversarial robustness analyses for GPC models on a synthetic two-dimensional data set, the SPAM data set (Dua and Graff, 2017), and a feature-based analysis of a binary and a 3-class subset of the MNIST data set (LeCun, 1998). In particular, we compare the guarantees computed by our method with the robustness estimation approximated by adversarial attack methods for GPCs (Grosse et al., 2018), discussing in which settings the latter fail. Then, we analyse the effect of approximate Bayesian inference techniques and hyperparameter optimisation procedures on the GPC adversarial robustness. Interestingly, across the three data sets analysed here, we observe that approximations based on Expectation Propagation (Minka, 2001) give more robust classification models than Laplace approximations (Rasmussen and Williams, 2006), and that GPC robustness increases with the number of epochs during hyperparameter optimisation. Finally, we show how robustness can be used to perform interpretability analysis in the form of sensitivity analyses of GPC predictions and compare our methodology with LIME (Ribeiro et al., 2016).

3.2 Related Work

Different notions of robustness have been studied for GPs. For instance, Kim and Ghahramani (2008) consider robustness against outliers, while Hernández-

Lobato et al. (2011) study robustness against labelling errors. Instead, here we consider robustness against adversarial examples, whose quantification for Bayesian models is a problem addressed in several works. Heuristic approaches based on studying adversarial examples using adversarial attacks are developed by Grosse et al. (2018); Feinman et al. (2017). Formal guarantees are derived by Cardelli et al. (2019b); Bogunovic et al. (2018); Smith et al. (2019) for GPs and by Cardelli et al. (2019a); Wicker et al. (2020); Carbone et al. (2020) for Bayesian neural networks. In particular, Cardelli et al. (2019b) derive an upper bound on the probability that there exists a point in the neighbourhood of a given test point of a GP such that the prediction of the GP on the latter differs from the initial test input point by at least a specified threshold, whereas Bogunovic et al. (2018) consider a GP optimisation algorithm in which the returned solution is guaranteed to be robust to adversarial perturbations with a certain probability. The problem and the techniques developed in this chapter are substantially different from both of these. First, we consider a classification problem, for which the bounds in the referenced papers cannot be applied due to the non-Gaussian nature of GPC prediction functions (see Section 2.1.3). Furthermore, the approach in this work gives stronger (i.e., non-probabilistic) guarantees, is guaranteed to converge to any given error $\epsilon > 0$ in finite time, and is anytime (i.e., at any time it gives sound upper and lower bounds of the classification probabilities). This also differs from Cardelli et al. (2019a), where the authors consider statistical guarantees that require the solution of many non-linear optimisation problems (one for each sample from the posterior distribution). Our approach also differs from that by Smith et al. (2019), where the authors give guarantees for GPC in a binary classification setting under the L_0-norm and only consider the mean of the distribution in the latent space without taking into account the uncertainty intrinsic in the GPC framework. In contrast, our approach also considers multi-class classification, takes into account the full posterior distribution and allows for exact (up to $\epsilon > 0$) computation under any L_p-norm.

3.3 Preliminaries

3.3.1 Bayesian Classification with Gaussian Processes

The basics of Bayesian classification with GPs have been introduced in Section 2.1.3. We revisit crucial concepts here, in particular when we modify notation for ease of exposition.

Given a test point $x^* \in \mathbb{R}^d$, the probability assigned by a GPC model to x^* belonging to class c can be written as:

$$\pi(x^*|\mathcal{D})_c = \int \sigma(\bar{f})_c p(f(x^*) = \bar{f}|\mathcal{D}) d\bar{f}, \tag{3.1}$$

where $f(x^*) = [f(x^*)_1, \ldots, f(x^*)_C]$ is the latent function vector, $\sigma(\cdot)_c : \mathbb{R}^C \to [0,1]$ is the likelihood function for class c, $p(f(x^*) = \bar{f}|\mathcal{D})$ is the predictive posterior distribution of the GP after observing n data points $\mathcal{D} := \mathcal{D}_n$, and the integral is computed over the C-dimensional latent space (Rasmussen and Williams, 2006). The vector of class probabilities, $\Pi(x^*) = [\pi(x^*|\mathcal{D})_1, ..., \pi(x^*|\mathcal{D})_C]$, can be computed by iterating Eq. (3.1) for each class $c = 1, \ldots, C$. Of particular interest in applications is the binary classification case (i.e., when $C = 2$), which leads to a significant simplification of the inference equations and techniques, while still encompassing important practical applications (Kuss et al., 2005; Nickisch and Rasmussen, 2008) [1]. More specifically, in this case it suffices to compute $\pi(x^*|\mathcal{D}) = \int \sigma(\bar{f}) p(f(x^*) = \bar{f}|\mathcal{D}) d\bar{f} := \pi(x^*|\mathcal{D})_1$, with f being a univariate latent function and setting $\pi(x^*|\mathcal{D})_2 := 1 - \pi(x^*|\mathcal{D})$. In other words, when $C = 2$ the latent space of the GPC model is one-dimensional.

Unfortunately, as pointed out, even under the GP prior assumption, the posterior distribution in classification settings, $p(f(x^*) = \bar{f}|\mathcal{D})$, is non-Gaussian and intractable (Rasmussen and Williams, 2006). We here focus on Gaussian analytical approximations, that is, we employ GPC methods that perform approximate inference of $p(f(x^*) = \bar{f}|\mathcal{D})$ by estimating a Gaussian distribution $q(f(x^*) = \bar{f}|\mathcal{D}) = \mathcal{N}(\bar{f} \,|\, \mu(x^*), \Sigma(x^*))$ (compare Eq. (2.42) in Section 2.1.3). As

[1] Notice that the binary case can be used for multi-class classification as well by means of, e.g., one-vs-all classifiers (Rasmussen and Williams, 2006; Hsu and Lin, 2002)

shown in Eq. (2.43), the latter is then used at prediction time in Eq. (3.1) in place of the exact posterior $p(f(x^*) = \bar{f}|\mathcal{D})$. With an abuse and overloading of notation, in the rest of this work we will thus consider

$$\pi(x^*|\mathcal{D})_c = \int \sigma(\bar{f})_c q(f(x^*) = \bar{f}|\mathcal{D}) d\bar{f}, \tag{3.2}$$

i.e. the approximation to the true posterior class probability. The methods presented in this work do hold for Gaussian approximations using either *Laplace approximations* (Williams and Barber, 1998) or *Expectation Propagation* (EP, Minka (2001)).

3.3.2 Problem Formulation: Adversarial Robustness Guarantees

Given a GPC model trained on data set $\mathcal{D}$ and a test point x^*, we are interested in quantifying the adversarial robustness of the GPC in a neighborhood of x^*. To do so, for a compact set T and a class $c \in \{1, \ldots, C\}$, we pose the problem of computing the minimum and the maximum that the GPC assigns to the probability of class c in T, that is:

$$\pi_{\min}(T)_c := \min_{x \in T} \pi(x|\mathcal{D})_c \quad \pi_{\max}(T)_c := \max_{x \in T} \pi(x|\mathcal{D})_c \tag{3.3}$$

The computation of the classification extrema in T allows us to determine the reachable interval of class probabilities over T, which can be used to derive adversarial robustness guarantees. Unfortunately, exact computation of Eq. (3.3) involves the solution of two non-linear optimisation problems for which no general solution method exists. Nevertheless, in Sections 3.4 – 3.6 we derive a branch and bound scheme for the anytime computation of the classification ranges of Eq. (3.3) that is guaranteed to converge in finitely many iterations up to any arbitrary error tolerance $\epsilon > 0$.

To see how the computation of the classification range over T facilitates adversarial robustness analyses, consider the following two notions of adversarial robustness:

Definition 6. *(Adversarial Local Safety) Let $T \subseteq \mathbb{R}^d$ and $x^* \in T$ belong to class $y^* \in \{1, \ldots, C\}$. Then, we say that a correct classification of x^* is safe in T iff* $\forall x \in T$, $\arg\max_{c \in \{1,\ldots,C\}} \pi(x|\mathcal{D})_c = \arg\max_{c \in \{1,\ldots,C\}} \pi(x^*|\mathcal{D})_c = y^*$.

Adversarial local safety establishes whether adversarial examples exist in T (for the canonical 0 - 1 loss), yielding formal guarantees against adversarial attacks for GPCs. If we again consider T to be a $\gamma-$ball around x^*, the satisfaction of Definition 6 guarantees that it is not possible to cause a misclassification by perturbing x^* by a magnitude of up to γ. Thus, testing adversarial local safety for an iteratively increasing value of γ can give a good understanding of LMSR (Definition 5). Based on Eq. (3.3), adversarial local safety can be established in a straightforward way: for the binary case, if $\pi(x^*|\mathcal{D}) > 0.5$ (< 0.5), we need to check if $\pi_{\min} > 0.5$ ($\pi_{\max} < 0.5$). In the multiclass case in order to check if x^* is safe in T, for $\bar{c} = \arg\max_{c\in\{1,\dots,C\}} \pi(x^*|\mathcal{D})_c$, we need to check that $\min_{x\in T} (\pi(x|\mathcal{D})_{\bar{c}} - \max_{c\neq\bar{c}} \pi(x|\mathcal{D})_c) > 0$, which can be computed with a trivial extension of the results presented in this work.

Definition 7. *(Adversarial Local Robustness) Let $T \subseteq \mathbb{R}^d$ and $x^* \in T$. Then, for $\delta > 0$ we say that the classification of x^* is $\delta-$robust in T iff $\forall x \in T$, $|\Pi(x^*) - \Pi(x)| \leq \delta$, where $|\cdot|$ is a given norm.*

This notion of adversarial local robustness has been for example used by Ruan et al. (2018) to analyse adversarial robustness of deep learning models. Given the quantities in Eq. (3.3), it is straightforward to calculate a conservative estimate of the $\delta-$robustness. For T being a $\gamma-$ball around a test point x^*, then robustness defined in Definition 7 allows one to quantify how much, in the worst case, the prediction in x^* can be affected by input perturbations of radius no greater than γ. Therefore, $\delta-$robustness naturally leads to adversarial local safety according to Definition 6 if $\pi(x^*)_{y^*} - \max_{c\neq y^*} \pi(x^*)_c > \delta$.

Remark 2. *In this work, we compute the quantities in Eq. (3.3) and respectively all guarantees and measures of robustness (Definitions 6 and 7) for the approximation $\pi(x^*|\mathcal{D})_c$ in Eq. (3.2). While this might seem counterintuitive at first, given that it is unclear how big the approximation error w.r.t. the true posterior is for any given GPC model, they are in practice the only quantities of relevance. This is because*

any prediction will be based on this approximation due to the intractability of the true posterior. Lastly, we note that we ignore numerical approximation errors (e.g. when calculating Gaussian cdfs), as their magnitude is irrelevant when we evaluate safety in terms of classification probabilities, which have 4-5 post-comma digits of relevance at most.

In the following, we show how the desired quantities in Eq. (3.3) can be obtained not just for $\gamma-$balls, but for any compact set $T \subset \mathbb{R}^d$. Note that both in Eq. (3.3) and in Definitions 6 and 7 we do not assume any distribution in the input space. This is because we want to derive formal and worst-case scenario guarantees, which must hold for any possible local perturbation of the test point.

3.4 Guarantees for Binary Classification

In this section we show how the classification ranges of a two-class GPC model in any given compact set $T \subseteq \mathbb{R}^d$ can be computed up to any arbitrary precision $\epsilon > 0$. As explained in Section 3.3.1, the latent space of the GPC model is one-dimensional in this case, and we thus omit the class superscript c in this section. The extension to the multi-class scenario is then described in Section 3.6.

3.4.1 Outline of Approach

An outline of our approach is depicted in Figure 3.1 for the computation of $\pi_{\min}(T)$ over a one-dimensional set T plotted along the x-axis (the method for the computation of $\pi_{\max}(T)$ is analogous). For any given region T we aim to compute lower and upper bounds on both $\pi_{\min}(T)$ and $\pi_{\max}(T)$, that is, we compute real values $\pi^L_{\min}(T)$, $\pi^U_{\min}(T)$, $\pi^L_{\max}(T)$ and $\pi^U_{\max}(T)$ such that:

$$0 \leq \pi^L_{\min}(T) \leq \pi_{\min}(T) \leq \pi^U_{\min}(T) \leq 1 \tag{3.4}$$

$$0 \leq \pi^L_{\max}(T) \leq \pi_{\max}(T) \leq \pi^U_{\max}(T) \leq 1. \tag{3.5}$$

In order to do so, we compute a lower and an upper bound function (the lower bound function is depicted with a dashed red curve in Figure 3.1) to the GPC

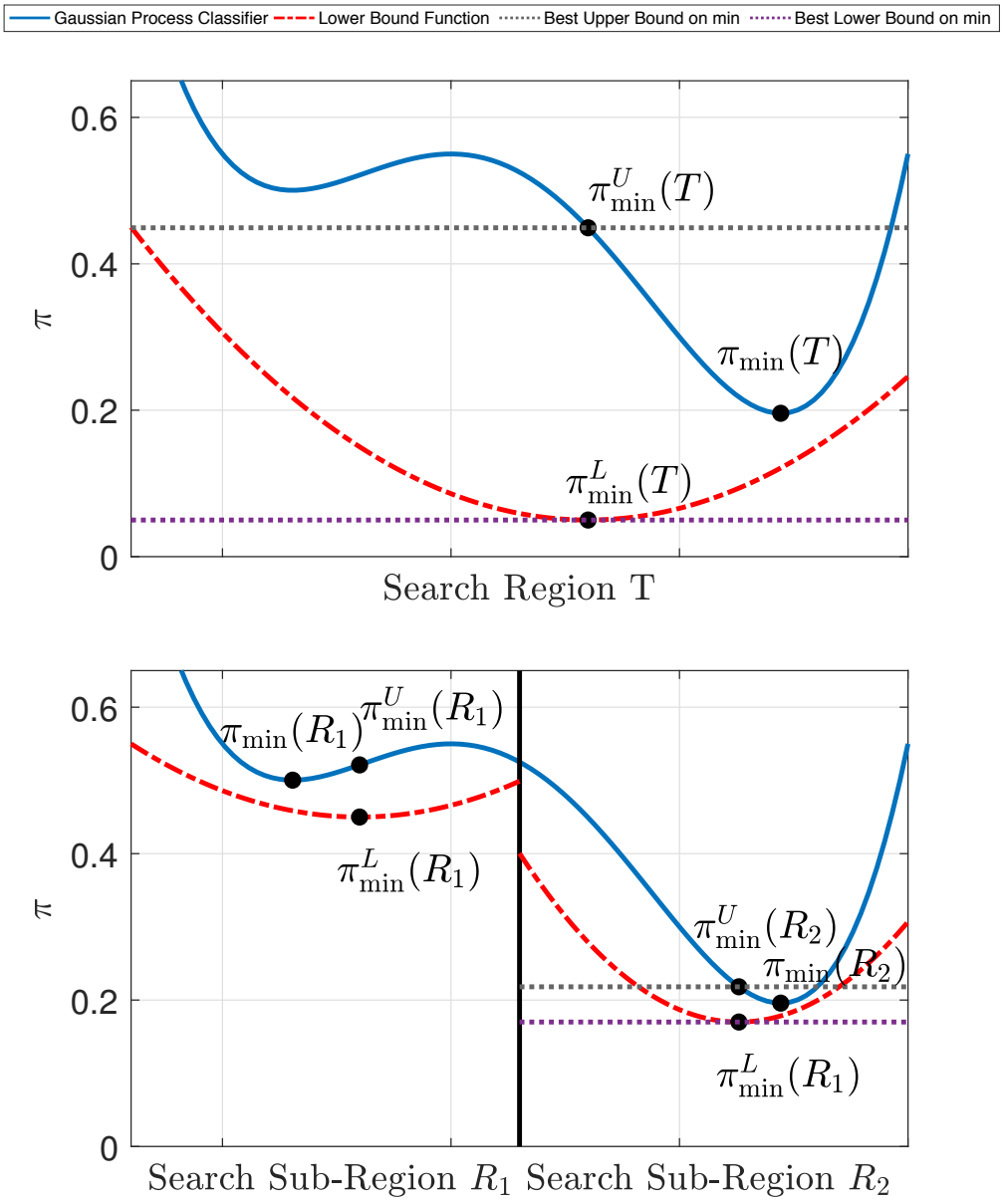


Figure 3.1: Top: Computation of upper and lower bounds on $\pi_{\min}(T)$, i.e. the minimum of the classification range on the search region T, using the methods we propose in Section 3.4.2. **Bottom:** The search region is repeatedly partitioned into sub-regions according to Algorithm 1 (only first partitioning visualised), reducing the gap between best lower and upper bounds until convergence (up to ϵ) is reached.

output (solid blue curve) in the region T. We then find the minimum of the lower bound function, $\pi^L_{\min}(T)$ (shown in the plot), and the maximum of the upper bound function, $\pi^U_{\max}(T)$ (not shown). Then, valid values for $\pi^U_{\min}(T)$ and $\pi^L_{\max}(T)$ can be computed by evaluating the GPC on any point in T (a specific $\pi^U_{\min}(T)$ is depicted in Figure 3.1). Finally, we iteratively refine the lower and upper bounds computed in T with a branch and bound algorithm. Namely, the region T is recursively subdivided into sub-regions, for which we compute new (tighter) bounds, until these converge up to a desired tolerance $\epsilon > 0$.

3.4.2 Computation of Bounds: General Case

Computation of Outer Bounds

In this paragraph we show how to compute $\pi^U_{\max}(T)$, an upper bound on the maximum, and $\pi^L_{\min}(T)$, a lower bound on the minimum of the GPC outputs. We work on the assumption that the likelihood function $\sigma(\cdot)$ is a monotonic, non-decreasing, and continuous function of the latent variable [2]. In the following proposition we show how the GPC output can be upper- and lower-bounded in T by a summation of Gaussian integrals.

Proposition 1. *Let $\mathcal{S} = \{S_i \mid i \in \{1, ...N\}\}$ be a partition of $\mathbb{R}$ (the latent space) in a finite set of intervals. Call $a_i = \inf_{\bar{f} \in S_i} \bar{f}$ and $b_i = \sup_{\bar{f} \in S_i} \bar{f}$. Then, it holds that:*

$$\pi_{\min}(T) \geq \sum_{i=1}^{N} \sigma(a_i) \min_{x \in T} \int_{a_i}^{b_i} \mathcal{N}(\bar{f} | \mu(x), \Sigma(x)) d\bar{f} \tag{3.6}$$

$$\pi_{\max}(T) \leq \sum_{i=1}^{N} \sigma(b_i) \max_{x \in T} \int_{a_i}^{b_i} \mathcal{N}(\bar{f} | \mu(x), \Sigma(x)) d\bar{f}, \tag{3.7}$$

where $\mu(x)$ and $\Sigma(x)$ are mean and variance of the predictive posterior $q(f(x) = \bar{f} | \mathcal{D})$.

[2] Notice that this is satisfied by commonly used likelihood functions, e.g., logistic and probit (Kim and Ghahramani, 2006)

Proof. We detail the proof for $\min_{x \in T} \pi(x|\mathcal{D})$. The max case follows similarly.

$$
\begin{aligned}
&\min_{x \in T} \pi(x|\mathcal{D}) \\
&\quad \text{(By definition)} \\
=&\min_{x \in T} \int_{-\infty}^{+\infty} \sigma(\bar{f}) q(f(x) = \bar{f}|\mathcal{D}) d\bar{f} \\
&\quad \text{(By additivity of integrals)} \\
=&\min_{x \in T} \sum_{i=1}^{N} \int_{a_i}^{b_i} \sigma(\bar{f}) q(f(x) = \bar{f}|\mathcal{D}) d\bar{f} \\
&\quad \text{(By monotonicity of } \sigma \text{ and non-negativity of } q) \\
\geq&\min_{x \in T} \sum_{i=1}^{N} \int_{a_i}^{b_i} \sigma(a_i) q(f(x) = \bar{f}|\mathcal{D}) d\bar{f} \\
&\quad \text{(By definition of minimum and of } q) \\
\geq&\sum_{i=1}^{N} \sigma(a_i) \min_{x \in T} \int_{a_i}^{b_i} \mathcal{N}(\bar{f}|\mu(x), \Sigma(x)) d\bar{f}
\end{aligned}
$$

□

Proposition 1 guarantees that the GPC output in T can be bounded by solving N simpler optimisation problems. Each of these problems seeks to find the mean and variance that maximise or minimise the integral of a Gaussian over T. As this problem cannot be safely solved directly in general, we introduce the following notation for safe lower and upper bounds on mean and variance in T (see remark 3 for their necessity):

$$\mu_T^L \leq \min_{x \in T} \mu(x) \quad \mu_T^U \geq \max_{x \in T} \mu(x) \tag{3.8}$$

$$\Sigma_T^L \leq \min_{x \in T} \Sigma(x) \quad \Sigma_T^U \geq \max_{x \in T} \Sigma(x), \tag{3.9}$$

The details on the computation of bounds for the a-posteriori mean and variance are discussed in Section 3.4.4, for now we assume them to be computable. Then by inspection of the derivatives of the integrals in Eqs. (3.6) and (3.7) the following proposition follows.

Proposition 2. *Let* $\mu^m = \frac{a+b}{2}$ *and* $\Sigma^m(\mu) = \frac{(\mu-a)^2-(\mu-b)^2}{2\log\frac{\mu-a}{\mu-b}}$. *Then it holds that:*

$$\begin{aligned}
\max_{x\in T}\int_a^b \mathcal{N}(\bar{f}|\mu(x),\Sigma(x))d\bar{f} &\leq \int_a^b \mathcal{N}(\bar{f}|\overline{\mu},\overline{\Sigma})d\bar{f} \\
&= \frac{1}{2}\left(\operatorname{erf}\left(\frac{\overline{\mu}-a}{\sqrt{2\overline{\Sigma}}}\right)-\operatorname{erf}\left(\frac{\overline{\mu}-b}{\sqrt{2\overline{\Sigma}}}\right)\right) \qquad (3.10)\\
\min_{x\in T}\int_a^b \mathcal{N}(\bar{f}|\mu(x),\Sigma(x))d\bar{f} &\geq \int_a^b \mathcal{N}(\bar{f}|\underline{\mu},\underline{\Sigma})d\bar{f} \\
&= \frac{1}{2}\left(\operatorname{erf}\left(\frac{\underline{\mu}-a}{\sqrt{2\underline{\Sigma}}}\right)-\operatorname{erf}\left(\frac{\underline{\mu}-b}{\sqrt{2\underline{\Sigma}}}\right)\right) \qquad (3.11)
\end{aligned}$$

where: $\overline{\mu} = \arg\min_{\mu\in[\mu_T^L,\mu_T^U]}|\mu^m-\mu|$ *and* $\overline{\Sigma}$ *is equal to* Σ_T^L *if* $\overline{\mu}\in[a,b]$, *otherwise* $\overline{\Sigma} = \arg\min_{\Sigma\in[\Sigma_T^L,\Sigma_T^U]}|\Sigma^m(\overline{\mu})-\Sigma|$. *Analogously, for the minimum we have:* $\underline{\mu} = \arg\max_{\mu\in[\mu_T^L,\mu_T^U]}|\mu^m-\mu|$ *and* $\underline{\Sigma} = \arg\min_{\Sigma\in\{\Sigma_T^L,\Sigma_T^U\}}[\operatorname{erf}(b|\underline{\mu},\Sigma)-\operatorname{erf}(a|\underline{\mu},\Sigma)]$.

Remark 3. *Note that if* $\min_{x\in T}\mu(x)$, $\max_{x\in T}\mu(x)$, $\min_{x\in T}\Sigma(x)$, *and* $\max_{x\in T}\Sigma(x)$ *could be safely computed directly, the inequalities in Eqns. (3.10) and (3.11) could be replaced with equalities (if one also replaced* $\mu_T^L,\mu_T^U,\Sigma_T^L$, *and* Σ_T^U *in the definitions of* $\overline{\mu},\overline{\Sigma},\underline{\mu}$, *and* $\underline{\Sigma}$ *with* $\min_{x\in T}\mu(x)$, $\max_{x\in T}\mu(x)$, $\min_{x\in T}\Sigma(x)$, *and* $\max_{x\in T}\Sigma(x)$ *accordingly) and we could compute the quantities of interest exactly. Unfortunately, due to the non-linear nature of* $\mu(x)$ *and* $\Sigma(x)$ *(see Eq. (2.42)), this is not possible in general and in particular, methods like grid search over the d-dimensional input space cannot be considered safe (i.e. they might return a solution to* $\min_{x\in T}\mu(x)$ *that is strictly greater than the true* $\min_{x\in T}\mu(x)$ *for example). That is why we have to resort to finding the safe over-approximations in Eqns. (3.8) and (3.9), whose computation we describe below in Section 3.4.4.*

Proof. We again provide the proof for the min case, similar arguments hold for the max. By definition of μ_T^L, μ_T^U, Σ_T^L, Σ_T^U we have that:

$$\begin{aligned}
\min_{x\in T}\int_a^b \mathcal{N}(\bar{f}|\mu(x),\Sigma(x))d\bar{f} &\geq \min_{\substack{\mu\in[\mu_T^L,\mu_T^U]\\ \Sigma\in[\Sigma_T^L,\Sigma_T^U]}}\int_a^b \mathcal{N}(\bar{f}|\mu,\Sigma)d\bar{f} \\
&= \frac{1}{2}\min_{\substack{\mu\in[\mu_T^L,\mu_T^U]\\ \Sigma\in[\Sigma_T^L,\Sigma_T^U]}}\left(\operatorname{erf}\left(\frac{\mu-a}{\sqrt{2\Sigma}}\right)-\operatorname{erf}\left(\frac{\mu-b}{\sqrt{2\Sigma}}\right)\right) \\
&:= \frac{1}{2}\min_{\substack{\mu\in[\mu_T^L,\mu_T^U]\\ \Sigma\in[\Sigma_T^L,\Sigma_T^U]}}\Phi(\mu,\Sigma).
\end{aligned}$$

By looking at the partial derivatives we have that:

$$\frac{\partial \Phi(\mu, \Sigma)}{\partial \mu} = \frac{\sqrt{2}}{\sqrt{\pi \Sigma}} \left(-e^{-\frac{(\mu-b)^2}{2\Sigma}} + e^{-\frac{(\mu-a)^2}{2\Sigma}} \right) \geq 0$$
$$\Leftrightarrow \quad \mu \leq \frac{a+b}{2} =: \mu^m$$

and that if $\mu \notin [a, b]$:

$$\frac{\partial \Phi(\mu, \Sigma)}{\partial \Sigma} = \frac{1}{\sqrt{2\pi\Sigma^3}} \left((\mu - b) e^{-\frac{(\mu-b)^2}{2\Sigma}} - (\mu - a) e^{-\frac{(\mu-a)^2}{2\Sigma}} \right) \geq 0$$
$$\Leftrightarrow \quad \Sigma \leq \frac{(\mu-a)^2 - (\mu-b)^2}{2 \log \frac{\mu-a}{\mu-b}} =: \Sigma^m(\mu)$$

otherwise the last inequality has no solutions. As such μ^m and Σ^m will correspond to global maximum w.r.t. to μ and Σ respectively. As Φ is symmetric w.r.t. μ^m we have that the minimum value w.r.t. to μ is always obtained for the point furthest away from μ^c, that is: $\underline{\mu} = \arg\max_{\mu \in [\mu_T^L, \mu_T^U]} |\mu^m - \mu|$. The minimum value w.r.t. to Σ will hence be either for Σ_T^L or Σ_T^U, that is $\underline{\Sigma} = \arg\min_{\Sigma \in \{\Sigma_T^L, \Sigma_T^U\}} \Phi(\underline{\mu}, \Sigma)$. □

That is, given lower and upper bounds for the a-posteriori mean and variance in T, Proposition 2 allows us to analytically bound the N optimisations of Gaussian integrals posed by Equations (3.6) and (3.7). Through this, we can compute values for $\pi_{\min}^L(T)$ and $\pi_{\max}^U(T)$, which satisfy the LHS of Eq. (3.4) and the RHS of Eq. (3.5).

Computation of Inner Bounds

Note that by definition of $\pi_{\min}(T)$ and $\pi_{\max}(T)$, we have that, for every $\bar{x} \in T$, setting $\pi_{\min}^U(T) = \pi_{\max}^L(T) = \pi(\bar{x})$ provides values which satisfy the RHS of Eq. (3.4) and the LHS of Eq. (3.5). However, the closer $\pi_{\min}^U(T)$ and $\pi_{\max}^L(T)$ are to $\pi_{\min}(T)$ and $\pi_{\max}(T)$ respectively, the quicker will the convergence of the branch and bound algorithm which we use to refine the bounds be. As we will see in Section 3.4.4, we obtain four extrema points in T on which the GPC assume the optimal values a-posteriori mean and variance values when we compute μ_T^L, μ_T^U, Σ_T^U and Σ_T^L. As these points belong to T and provide extreme points for the latent function they make promising candidates for the evaluation of $\pi_{\min}^U(T)$ and

$\pi^L_{\max}(T)$. Therefore, we evaluate the GPC on all four extrema and select the one that gives the best inner bound among them.

3.4.3 Computation of Bounds: Special Case of Probit Likelihood

When the (scaled) probit function is chosen for the likelihood, $\sigma(\cdot)$, then the inference integral over $q(f(x^*) = \bar{f}|\mathcal{D})$ in Eq. (3.2) can be expressed in closed form (Rasmussen and Williams, 2006), which leads to a simplification of Proposition 1. For the case that the likelihood σ is taken to be the probit function, that is, $\sigma(\bar{f}) = \Phi(\lambda\bar{f})$ is the cdf of the univariate standard Gaussian distribution scaled by $\lambda > 0$, it holds that

$$\pi(x \,|\, \mathcal{D}) = \Phi\left(\frac{\mu(x)}{\sqrt{\lambda^{-2} + \Sigma(x)}}\right),$$

where $\mu(x)$ and $\Sigma(x)$ are the mean and variance of $q(f(x) = \bar{f}|\mathcal{D})$ (Bishop, 2006). We can use this result to derive analytic upper and lower bounds for Eq. (3.3) without the need to apply Propositions 1 and 2, by relying on upper and lower bounds for the latent mean and variance functions.

Lemma 1. *Let $T \subseteq \mathbb{R}^d$ and the likelihood σ be the probit function, that is, $\sigma(\bar{f}) = \Phi(\lambda\bar{f})$ being the cdf of the univariate standard Gaussian distribution scaled by $\lambda > 0$. Then, we have that*

$$\Phi\left(\frac{\mu_T^L}{\sqrt{\lambda^{-2} + \underline{\Sigma}}}\right) \leq \pi_{\min}(T) \tag{3.12}$$

and

$$\pi_{\max}(T) \leq \Phi\left(\frac{\mu_T^U}{\sqrt{\lambda^{-2} + \overline{\Sigma}}}\right) \tag{3.13}$$

with $\underline{\Sigma} = \Sigma_T^U$ if $\mu_T^L \geq 0$ and Σ_T^L otherwise, while $\overline{\Sigma} = \Sigma_T^L$ if $\mu_T^U \geq 0$ and Σ_T^U otherwise.

Proof. As Φ is a monotonically increasing function, it suffices to optimise for the argument $\phi(\mu, \Sigma) = \frac{\mu}{\sqrt{\lambda^{-2}+\Sigma}}$. By computing the partial derivatives it's straight forward to see that $\frac{\partial\phi(\mu,\Sigma)}{\partial\mu} > 0$ for all values of μ and Σ. As such, for every value of

Σ, the minimum is obtained for $\mu = \mu_T^L$. On the other hand for the derivative w.r.t. to Σ we have that

$$\frac{\partial \phi(\mu_T^L, \Sigma)}{\partial \Sigma} \begin{cases} < 0 & \text{if} \quad \mu_T^L > 0 \\ = 0 & \text{if} \quad \mu_T^L = 0 \\ > 0 & \text{if} \quad \mu_T^L < 0 \end{cases}$$

as $\Sigma > 0$. Hence, given μ_T^L, we have that ϕ is monotonic in Σ. The proposition statement follows directly from this observation. □

3.4.4 Obtaining Bounds on Latent Mean and Variance

We obtain the necessary bounds on latent mean and variance by applying the framework presented by Cardelli et al. (2019b) for computation of μ_T^L, μ_T^U and Σ_T^U, and subsequently extend it to the computation of Σ_T^L. Briefly, assuming continuity and differentiability of the kernel function defining the GPC covariance, it is possible to find linear upper and lower bounds on the covariance vector, $k(x, X)$, which can be propagated through the inference formula for $q(f(x) = \bar{f}|\mathcal{D})$. The bounding functions obtained in this way can be analytically optimised for μ_T^L and μ_T^U [3], while convex quadratic programming is used to obtain Σ_T^U (see Cardelli et al. (2019b) for details). Finally, we solve the concave quadratic problem that arises when computing Σ_T^L by adapting methods introduced by Rosen and Pardalos (1986), which reduces the problem to the solution of $2|\mathcal{D}| + 1$ linear programming problems. This is detailed in the following subsection.

Lower Bound on Latent Variance

Remember that for variance of the predictive latent function value $f(x)$ we get according to Eq. (2.42)

$$\mathbb{V}(f(x)) = k(x, x) - k(x, \boldsymbol{X})[K(\boldsymbol{X}, \boldsymbol{X})^{-1} - K(\boldsymbol{X}, \boldsymbol{X})^{-1}\boldsymbol{M}K(\boldsymbol{X}, \boldsymbol{X})^{-1}]k(\boldsymbol{X}, x).$$

[3] As we have $\mu(x) = k(x, \boldsymbol{X})K(\boldsymbol{X}, \boldsymbol{X})^{-1}\boldsymbol{m}$, which can be written as $\sum_{i=1}^{n} \alpha_i k(x, x_i)$. Thus upper and lower bounding $k(x, \boldsymbol{X})$ enables straightforward derivation of μ_T^L and μ_T^U

For stationary kernels, this can be written as

$$\mathbb{V}(f(x)) = c_k - \mathbf{r}(x)^T S \mathbf{r}(x)$$

with $\mathbf{r}(x) = k(\boldsymbol{X}, x)$, since $k(x, x)$ and $[K(\boldsymbol{X}, \boldsymbol{X})^{-1} - K(\boldsymbol{X}, \boldsymbol{X})^{-1} \boldsymbol{M} K(\boldsymbol{X}, \boldsymbol{X})^{-1}]$ are independent of x for stationary kernels. Thus, we are interested in finding a lower bound for $\min_{x \in T} \left(c_k - \mathbf{r}(x)^T S \mathbf{r}(x) \right) = c_k + \min_{x \in T} \left(-\mathbf{r}(x)^T S \mathbf{r}(x) \right)$. We proceed by introducing the n auxiliary variables $r_i = \mathbf{r}_i(x)$, yielding a quadratic objective function on the auxiliary variable vector $\mathbf{r} = [r_1, \ldots, r_n]$, that is $-\mathbf{r}^T S \mathbf{r}$. Analogously to what is done by Cardelli et al. (2019b) we can compute two matrices A_r, A_x and a vector b such that $\mathbf{r} = \mathbf{r}(x)$ implies $A_r \mathbf{r} + A_x x \leq b$, hence obtaining the quadratic program:

$$\begin{aligned} &\min -\mathbf{r}^T S \mathbf{r} \\ &\text{Subject to:} \quad A_r \mathbf{r} + A_x x \leq b \\ &\qquad\qquad r_i^L \leq r_i \leq r_i^U \qquad i = 1, \ldots, n \\ &\qquad\qquad x_i^L \leq x_i \leq x_i^U \qquad i = 1, \ldots, d \end{aligned} \tag{3.14}$$

whose solution provides a lower bound (and hence a safe approximation) to the original problem $\min_{x \in T} \left(-\mathbf{r}(x)^T S \mathbf{r}(x) \right)$. Unfortunately, as S is positive definite in the cases we consider, i.e. for Laplace and EP approximations (Rasmussen and Williams, 2006), we have that $-S$ is negative definite. Hence the problem posed is a concave quadratic program for which a number of local optima may exist. As we are instead dealing with worst-case scenario analyses, we are actually interested in computing the global minimum. This however is an NP-hard problem (Rosen and Pardalos, 1986) whose exact solution would make a branch and bound algorithm based on it impractical. Following the methods discussed in Rosen and Pardalos (1986), we instead proceed to compute a safe lower bound to it. The main observation is that, with being S symmetric positive definite, there exist a matrix of eigenvectors $U = [\mathbf{u}_1, \ldots, \mathbf{u}_n]$ and a diagonal matrix of the associated eigenvalues λ_i for $i = 1, \ldots, n$, Λ such that $S = U \Lambda U^T$. We hence define $\hat{r}_i = \mathbf{u}_i^T \mathbf{r}$

for $i = 1, \ldots, n$ to be the rotated variables and compute their ranges $[\hat{r}_i^L, \hat{r}_i^U]$ by solution of the following $2n$ linear programming problems:

$$\begin{aligned} \min / \max \quad & \mathbf{u}_i^T \mathbf{r} \\ \text{Subject to:} \quad & A_r \mathbf{r} + A_x x \leq b \\ & r_j^L \leq r_i \leq r_j^U \qquad j = 1, \ldots, n \\ & x_j^L \leq x_i \leq x_j^U \qquad j = 1, \ldots, d. \end{aligned} \tag{3.15}$$

Implementing the change of variables into Problem 3.14 we obtain:

$$\begin{aligned} \min \quad & -\hat{\mathbf{r}}^T \Lambda \hat{\mathbf{r}} \\ \text{Subject to:} \quad & \hat{A}_{\hat{r}} \hat{\mathbf{r}} + A_x x \leq b \\ & \hat{r}_i^L \leq \hat{r}_i \leq \hat{r}_i^U \qquad i = 1, \ldots, n \\ & x_i^L \leq x_i \leq x_i^U \qquad i = 1, \ldots, d \end{aligned} \tag{3.16}$$

where we set $\hat{A} = AU$. We then notice that $\hat{\mathbf{r}}^T \Lambda \hat{\mathbf{r}} = \sum_i \lambda_i \hat{r}_i^2$. By using the methods developed by Cardelli et al. (2019b) it is straightforward to find coefficients of a linear under approximations α_i and β_i such that: $\alpha_i + \beta_i \hat{r}_i \leq -\lambda_i \hat{r}_i^2$ for $i = 1, \ldots, n$. Defining $\beta = [\beta_1, \ldots, \beta_n]$, and $\hat{\alpha} = \sum_{i=1}^n \alpha_i$ we then have that the solution to the following linear programming problem provides a valid lower bound to Problem 3.14 and can be hence used to compute a lower bound to the latent variance:

$$\begin{aligned} \min \quad & \left(\hat{\alpha} + \beta^T \hat{\mathbf{r}} \right) \\ \text{Subject to:} \quad & \hat{A}_{\hat{r}} \hat{\mathbf{r}} + A_x x \leq b \\ & \hat{r}_i^L \leq \hat{r}_i \leq \hat{r}_i^U \qquad i = 1, \ldots, n \\ & x_i^L \leq x_i \leq x_i^U \qquad i = 1, \ldots, d. \end{aligned} \tag{3.17}$$

Proposition 3. *Let $\underline{\sigma}^2$ be the solution to the linear programming problem defined in Eq. (3.17). Then $\Sigma_T^L := c_k + \underline{\sigma}^2$ is a lower bound for the posterior variance $\mathbb{V}(f(x))$ in T.*

3.5 Branch and Bound Algorithm

3.5.1 The Algorithm

In this paragraph we implement the bounding procedure into a branch and bound algorithm and prove convergence up to any a-priori specified $\epsilon > 0$. We summarise our method for computation of $\pi_{\min}(T)$ in Algorithm 1, which we now briefly describe (analogous arguments hold for $\pi_{\max}(T)$).

Algorithm 1 Branch and bound for $\pi_{\min}(T)$

Input: Input space subset T; error tolerance $\epsilon > 0$; latent mean/variance functions $\mu(\cdot)$ and $\Sigma(\cdot)$ of $q(f(x) = \bar{f}|\mathcal{D})$
Output: Lower and upper bounds $\pi^L_{\min}(T)$ and $\pi^U_{\min}(T)$ on $\pi_{\min}(T)$ with $\pi^U_{\min}(T) - \pi^L_{\min}(T) \leq \epsilon$

1: *Initialisation:* Stack of regions $\mathbf{R} \leftarrow \{T\}$; $\pi^L_{\min}(T) \leftarrow -\infty$; $\pi^U_{\min}(T) \leftarrow +\infty$
2: **while** $\pi^U_{\min}(T) - \pi^L_{\min}(T) > \epsilon$ **do**
3: Select region $R \in \mathbf{R}$ with lowest bound $\pi^L_{\min}(R)$ and delete it from stack
4: Find bounds $[\mu^L_R, \mu^U_R]$ and $[\Sigma^L_R, \Sigma^U_R]$ for latent mean and variance functions over R
5: Compute $\pi^L_{\min}(R)$ from $[\mu^L_R, \mu^U_R]$ and $[\Sigma^L_R \Sigma^U_R]$ using Propositions 1 and 2
6: Find $\pi^U_{\min}(R)$ by evaluating GPC in a point in R
7: **if** $\pi^U_{\min}(R) - \pi^L_{\min}(R) > \epsilon$ **then**
8: Split R into two sub-regions R_1, R_2, add them to stack and use $\pi^L_{\min}(R), \pi^U_{\min}(R)$ as initial bounds for both sub-regions
9: **end if**
10: Update $\pi^L_{\min}(T)$ and $\pi^U_{\min}(T)$ with current best bounds found
11: **end while**
12: **return** $[\pi^L_{\min}(T), \pi^U_{\min}(T)]$

After initialising $\pi^L_{\min}(T)$ and $\pi^U_{\min}(T)$ to trivial values and initialising the exploration regions stack $\mathbf{R}$ to the singleton $\{T\}$, the main optimisation loop is entered until convergence (lines 2–10). Among the regions in the stack, we select the region R with the most promising lower bound (line 3), and refine its lower bounds using Propositions 1 and 2 (lines 4–5) as well as its upper bounds through evaluation of points in R (line 6). If further exploration of R is necessary for convergence (line 7), then the region R is partitioned into two smaller regions R_1 and R_2 (for which potentially again tighter bounds can be computed later on), which are added to the regions stack and inherit R's bound values (line 8). Finally, the freshly computed

bounds local to $R \subseteq T$ are used to update the global bounds for T (line 10). Namely, $\pi^L_{\min}(T)$ is updated to the smallest value among the $\pi^L_{\min}(R)$ values for $R \in \mathbf{R}$, while $\pi^U_{\min}(T)$ is set to the lowest observed value yet explicitly computed in line 6.

For our approach to work, it is crucial that Algorithm 1 converges, i.e. that the loop of lines $2-10$ terminates. Given an a-priori specified threshold ϵ, Theorem 1 ensures that there exists a latent space discretisation such that the bounding error (i.e. the difference between the upper and lower bound) for a region $R \subset T$ vanishes as $diam(R) \to 0$. Thanks to the properties of branch and bound algorithms (Balakrishnan et al., 1991), this guarantees that our method converges in finitely many iterations.

Theorem 1. *Assume $\mu : \mathbb{R}^d \to \mathbb{R}$ and $\Sigma : \mathbb{R}^d \to \mathbb{R}$ are Lipschitz continuous in $T \subseteq \mathbb{R}^d$. Then, for $\epsilon > 0$, there exists a partition of the latent space $\mathcal{S}$ and $\delta > 0$ such that, for every $R \subseteq T$ of side length of less than δ, it holds that $|\pi^U_{\min}(R) - \pi^L_{\min}(R)| \leq \epsilon$ and $|\pi^U_{\max}(R) - \pi^L_{\max}(R)| \leq \epsilon$.*

Remark 4. *While Theorem 1 guarantees the convergence of our method w.r.t. the function value of $\pi_{\min}(T)$ and $\pi_{\max}(T)$, it does not give guarantees on the location of the corresponding extremal point (e.g. in the presence of multiple local minima). From an adversarial robustness guarantees point of view, its precise location in T is however not relevant, as we only care about the values of $\pi_{\min}(T)$ and $\pi_{\max}(T)$ to establish safety and robustness (see Definitions 6 and 7).*

Proof. We consider the min case. The max case follows similarly. In order to show the convergence of the branch and bound, we need to show that for any test point x there exists $\delta > 0$ and a partition of the latent space $\mathcal{S} = \{S_i, i = 1, ..., N\}$ such that for an input set I with $diam(I) = \delta$ (w.r.t. a given L_p-norm), we have that for any $\bar{x} \in I$

$$\left| \pi(\bar{x}|\mathcal{D}) - \sum_{i=1}^{N} \sigma(a_i) \min_{x \in I} \int_{a_i}^{b_i} \mathcal{N}(\bar{f}|\mu(x), \Sigma(x)) d\bar{f} \right| \leq \epsilon.$$

In order to do that, we first observe that by the Lipschitz continuity of mean and variance we have that for $x_1, x_2 \in I$, it holds that

$$\begin{aligned} |\mu(x_1) - \mu(x_2)| &\leq K^{\mu}\delta \\ |\Sigma(x_1) - \Sigma(x_2)| &\leq K^{\Sigma}\delta, \end{aligned}$$

for certain $K^{\mu}, K^{\Sigma} > 0$. Now, for $S_i \in \mathcal{S}$, consider x^i such that $\int_{a_i}^{b_i} \mathcal{N}(\bar{f}|\mu(x^i), \Sigma(x^i))d\bar{f} = \min_{x \in I} \int_{a_i}^{b_i} \mathcal{N}(\bar{f}|\mu(x), \Sigma(x))d\bar{f}$. Further, due to the monotonicity and continuity of σ, we can consider a uniform discretisation of the y-axis (i.e. of $[0, 1]$) for σ in N intervals. That is, for all $S_i \in \mathcal{S}$, we have that $\sigma(b_i) = \sigma(a_i) + \frac{1}{N}$ with $\sigma(a_0) = 0$. At this point, for any $\bar{x} \in I$ the following calculations follow

$$|\pi(\bar{x}|\mathcal{D}) - \sum_{i=1}^{N} \sigma(a_i) \int_{a_i}^{b_i} \mathcal{N}(\bar{f}|\mu(x^i), \Sigma(x^i))d\bar{f}| \tag{3.18}$$

(By Definition)

$$\begin{aligned} =| \int \sigma(\bar{f})\mathcal{N}(\bar{f}|\mu(\bar{x}), \Sigma(\bar{x}))d\bar{f} \\ - \sum_{i=1}^{N} \sigma(a_i) \int_{a_i}^{b_i} \mathcal{N}(\bar{f}|\mu(x^i), \Sigma(x^i))d\bar{f}| \end{aligned} \tag{3.19}$$

(By additivity of integral and re-ordering terms)

$$\begin{aligned} =| \sum_{i=1}^{N} \Big(\int_{a_i}^{b_i} \sigma(\bar{f})\mathcal{N}(\bar{f}|\mu(\bar{x}), \Sigma(\bar{x}))d\bar{f} \\ - \int_{a_i}^{b_i} \sigma(a_i)\mathcal{N}(\bar{f}|\mu(x^i), \Sigma(x^i))d\bar{f} \Big)| \end{aligned} \tag{3.20}$$

(As for any $\bar{f} \in S_i$, $\sigma(a_i) \leq \sigma(\bar{f}) \leq \sigma(a_i) + \dfrac{1}{N}$)

$$\leq |\sum_{i=1}^{N} \Big(\int_{a_i}^{b_i} (\sigma(a_i) + \frac{1}{N}) \mathcal{N}(\bar{f}|\mu(\bar{x}), \Sigma(\bar{x})) - \sigma(a_i)\mathcal{N}(\bar{f}|\mu(x^i), \Sigma(x^i))d\bar{f}\Big)| \tag{3.21}$$

(By Triangle Inequality)

$$\leq |\sum_{i=1}^{N} \int_{a_i}^{b_i} \frac{1}{N} \mathcal{N}(\bar{f}|\mu(\bar{x}), \Sigma(\bar{x}))d\bar{f}| + |\sum_{i=1}^{N} \Big(\sigma(a_i) \int_{a_i}^{b_i} \mathcal{N}(\bar{f}|\mu(\bar{x}), \Sigma(\bar{x})) - \mathcal{N}(\bar{f}\mu(x^i), \Sigma(x^i))d\bar{f}\Big)| \tag{3.22}$$

(By Re-ordering terms and Triangle Inequality)

$$\leq |\frac{1}{N} \int \mathcal{N}(\bar{f}|\mu(\bar{x}), \Sigma(\bar{x}))d\bar{f}| + \sum_{i=1}^{N} \sigma(a_i)| \int_{a_i}^{b_i} \mathcal{N}(\bar{f}|\mu(\bar{x}), \Sigma(\bar{x})) - \mathcal{N}(\bar{f}\mu(x^i), \Sigma(x^i))d\bar{f}| \tag{3.23}$$

(By properties of integrals and $\sigma(f) \in [0, 1]$)

$$\leq \frac{1}{N} + \sum_{i=1}^{N} | \int_{a_i}^{b_i} (\mathcal{N}(\bar{f}|\mu(\bar{x}), \Sigma(\bar{x})) - \mathcal{N}(\bar{f}|\mu(x^i), \Sigma(x^i)))d\bar{f}| \tag{3.24}$$

Now, as $|\mu(\bar{x}), -\mu(x^i), | \leq K^{\mu}\delta$ and $|\Sigma(\bar{x}) - \Sigma(x^i)| \leq K^{\Sigma}\delta$, we have that as $\delta \to 0$ both mean and variance converge to the same value. Hence, this implies that for each $S_i \in S$

$$\lim_{\delta \to 0} \Big(\int_{a_i}^{b_i} \mathcal{N}(\bar{f}|\mu(\bar{x}), \Sigma(\bar{x}))d\bar{f} - \int_{a_i}^{b_i} \mathcal{N}(\bar{f}|\mu(x^i), \Sigma(x^i))d\bar{f}\Big) = 0.$$

As a consequence, for any $\epsilon > 0$, we can choose $N = \lceil \frac{2}{\epsilon} \rceil$ and then select δ such that the second term in Eq. (3.24) is bounded by $\frac{\epsilon}{2}$. □

3.5.2 Computational Complexity

Proposition 2 implies that the bounds in Proposition 1 can be obtained in $\mathcal{O}(N)$, with N being the number of intervals the real line is being partitioned into (this scales like $\frac{1}{\epsilon}$, as discussed in the proof of Theorem 1). Computation of μ_T^L and μ_T^U is performed in $\mathcal{O}(n)$, where n is the number of training points, while obtaining Σ_T^U involves the solution of a convex quadratic problem in $d + n$ variables, where d is the dimension of the input space. Solving for Σ_T^L requires the solution of $2n + 1$ linear programming problems in $d + n$ dimensions. Refining through branch and bound has a worst-case cost exponential in the number of non-trivial dimensions of

T. An empirical analysis of the CPU time required for convergence of our method is provided in Section 3.7.5. In general, we found that this worst-case exponential computational complexity unfortunately prohibits the use of our method for large compact sets when the input dimensionality d is higher than ≈ 20.

3.6 Guarantees for Multiclass Classification

In this section we show how the results for binary classification can be generalised to the multi-class case. Given a class index $c \in \{1, \ldots, C\}$, we are interested in computing upper and lower bounds on $\pi(x|\mathcal{D})_c$ for every $x \in T$. In order to do so, we proceed as follows. First, we extend Proposition 1 to the multi-class case in Proposition 4. Then, in Proposition 5, we show that the resulting multi-variate Gaussian integrals can be reduced to the two-class case by marginalisation. As a result, Proposition 2 can be applied iteratively to the resulting product of univariate Gaussians to bound the multivariate Gaussian integrals in Proposition 4. Along the way we will proof auxiliary results in Lemmata 2 and 3.

Proposition 4. *Let $\mathcal{S} = \{S_i \mid i \in \{1, ...N\}\}$ be a finite partition of $\mathbb{R}^C$ (the latent space). Then, for $c \in \{1, ..., C\}$:*

$$\pi_{\min}(T)_c \geq \sum_{i=1}^{N} \min_{x \in S_i} \sigma(x)_c \min_{x \in T} \int_{S_i} \mathcal{N}(\bar{f}|\mu(x), \Sigma(x)) d\bar{f}$$
$$\pi_{\max}(T)_c \leq \sum_{i=1}^{N} \max_{x \in S_i} \sigma(x)_c \max_{x \in T} \int_{S_i} \mathcal{N}(\bar{f}|\mu(x), \Sigma(x)) d\bar{f}$$

Proof. We detail the proof for $\min_{x \in T} \pi_c$. The max case follows similarly.

$$\begin{aligned}
&\min_{x \in T} \pi(x|\mathcal{D})_c \\
&\quad \text{(By definition)} \\
&= \min_{x \in T} \int \sigma(\bar{f})_c q(f(x) = \bar{f}|\mathcal{D}) d\bar{f} \\
&\quad \text{(By additivity of integral)}
\end{aligned}$$

$$
\begin{aligned}
&= \min_{x \in T} \sum_{i=1}^{N} \int_{S_i} \sigma(\bar{f})_c q(f(x) = \bar{f} | \mathcal{D}) d\bar{f} \\
&\quad \text{(Because } q \text{ is non-negative)} \\
&\geq \min_{x \in T} \sum_{i=1}^{N} \int_{S_i} \min_{y \in S_i} \sigma(y)_c q(f(x) = \bar{f} | \mathcal{D}) d\bar{f} \\
&\quad \text{(By definition of infimum)} \\
&\geq \sum_{i=1}^{N} \min_{y \in S_i} \sigma(y)_c \min_{x \in T} \int_{S_i} q(f(x) = \bar{f} | \mathcal{D}) d\bar{f} \\
&\quad \text{(By Definition of } q) \\
&= \sum_{i=1}^{N} \min_{y \in S_i} \sigma(y)_c \min_{x \in T} \int_{S_i} \mathcal{N}(\bar{f} | \mu(x), \Sigma(x)) d\bar{f}
\end{aligned}
$$

□

Proposition 4 implies that if we can compute minimum and maximum of the softmax over a set of the latent space (shown in Lemma 2) and the mean and covariance matrix that maximise a Gaussian integral (shown in Proposition 5), then upper and lower bounds on $\pi_{\min}(T)$ and $\pi_{\max}(T)$ can be derived.

Lemma 2. *Let $S \subset \mathbb{R}^C$ be an axis-parallel hyper-rectangle. Call $f^{max} = \arg\max_{f \in S} \sigma(f)_c$ and $f^{min} = \arg\min_{f \in S} \sigma(f)_c$. Assume σ is the softmax function. Then, f^{max} and f^{min} are vertices of S.*

Proof. S is an axis-parallel hyper-rectangle. As a consequence, it can be written as intersection of constraints of the form $-f_i \leq -k_{i,1}$ and $f_i \leq k_{i,2}$, where f_i is the i-th component of vector f. Hence, the optimisation problem for the maximisation case (minimisation case is equivalent) can be rewritten as follows:

$$
\begin{aligned}
&\max \sigma(f)_c \\
&\text{such that } \forall i \in \{1, ..., C\} - f_i \leq -k_{i,1}, \quad f_i \leq k_{i,2}.
\end{aligned}
$$

In order to solve this problem we can apply the Karush-Kuhn-Tucker (KKT) conditions. Being the constraints independent of f, the KKT conditions imply that in order to conclude the proof we just need to show that for all $f \in S, c \in \{1, ..., C\}$, $\frac{d\sigma(f)_c}{df_c} \neq 0$. This is shown in what follows.

For $f \in \mathbb{R}^C$ and $c \in \{1, ...C\}$ We have

$$\sigma(f)_c = \frac{\exp(f_c)}{\sum_{j=1}^{C} \exp(f_j)}.$$

Then, we obtain

$$\frac{d\sigma(f)_c}{df_c} = \frac{\exp(f_c)(\sum_{j \neq c} \exp(f_j))}{(\sum_{j=1}^{C} \exp(f_j))^2},$$

while for $i \neq c$ we have

$$\frac{d\sigma(f)_c}{f_i} = -\frac{\exp(f_c)\exp(f_i)}{(\sum_{j=1}^{C} \exp(f_j))^2}.$$

This implies that for $f \in \mathbb{R}^n$ and $i \neq c$ we always have

$$\frac{d\sigma(f)_c}{df_c} > 0 \qquad \frac{d\sigma(f)_c}{df_i} < 0.$$

□

Note that in Lemma 2 we assumed that S is an hyper-rectangle. However, it can be extended to more general sets given by the intersection of arbitrarily many half-spaces generated by hyper-planes perpendicular to one of the axis.

Proposition 4 guarantees that, for all $x \in T$, $\pi(x|\mathcal{D})_c$ can be upper- and lower-bounded by solving $2N$ optimisation problems. In Proposition 5, we show that upper and lower bounds for the integral of a multi-dimensional Gaussian distribution, such as those appearing in Proposition 4, can be obtained by optimising uni-dimensional integrals over both the input and latent space. In what follows, we call $\mu_{i:j}(x)$ the subvector of $\mu(x)$ containing only the components from i to j (μ_i being only its i-th component), and similarly we define $\Sigma_{i:k,j:l}(x)$, the submatrix of $\Sigma(x)$ containing rows from i to k and columns from j to l ($\Sigma_{i,i}$ being the i-th entry on the diagonal of Σ).

Proposition 5. *Let $S = \prod_{i=1}^{C}[k_i^1, k_i^2]$ be an axis-parallel hyper-rectangle. For $i \in \{1, \ldots, C-1\}$ and $f \in \mathbb{R}^{C-1-i}$, define $\mathcal{I} := i+1 : C$ and*

$$\mu_i^f(x) = \mu_i(x) - \Sigma_{i,\mathcal{I}}(x)\Sigma_{\mathcal{I},\mathcal{I}}^{-1}(f - \mu_{\mathcal{I}}(x))$$
$$\Sigma_i^f(x) = \Sigma_{i,i}(x) - \Sigma_{i,\mathcal{I}}(x)\Sigma_{\mathcal{I},\mathcal{I}}^{-1}\Sigma_{i,\mathcal{I}}^T(x).$$

Let $S^{i+1} = \prod_{j=i+1}^{C}[k_j^1, k_j^2]$*, then we have that*

$$\max_{x\in T}\int_S \mathcal{N}(z|\mu(x),\Sigma(x)) \leq \max_{x\in T}\int_{k_C^1}^{k_C^2}\mathcal{N}(z|\mu_C(x), \Sigma_{C,C}(x))dz \prod_{i=1}^{C-1}\max_{\substack{x\in T\\ f\in S^{i+1}}}\int_{k_i^1}^{k_i^2}\mathcal{N}(z|\mu_i^f(x),\Sigma_i^f(x))dz$$

$$\min_{x\in T}\int_S \mathcal{N}(z|\mu(x),\Sigma(x)) \geq \min_{x\in T}\int_{k_C^1}^{k_C^2}\mathcal{N}(z|\mu_C(x), \Sigma_{C,C}(x))dz \prod_{i=1}^{C-1}\min_{\substack{x\in T\\ f\in S^{i+1}}}\int_{k_i^1}^{k_i^2}\mathcal{N}(z|\mu_i^f(x),\Sigma_i^f(x))dz.$$

The following Lemma is needed to prove Proposition 5.

Lemma 3. *Let* X *and* Y *be random variables with joint density function* f*. Consider measurable sets* A *and* B*. Then, it holds that*

$$P(X \in A|Y \in B) \leq \sup_{y\in B} P(X \in A|Y = y).$$

Proof.

$$\begin{aligned}
&P(X \in A|Y \in B)\\
&= \frac{P(X \in A \wedge Y \in B)}{P(Y \in B)}\\
&= \frac{\int_{x\in A}\int_{y\in B} f(X = x \wedge Y = y)dxdy}{P(Y \in B)}\\
&= \frac{\int_{x\in A}\int_{y\in B} f(X = x|Y = y)f(Y = y)dxdy}{P(Y \in B)}\\
&\leq \frac{\int_{x\in A}\int_{y\in B} \sup_{\bar{y}\in B} f(X = x|Y = \bar{y})f(Y = y)dxdy}{P(Y \in B)}\\
&= \frac{\int_{x\in A}\sup_{\bar{y}\in B} f(X = x|Y = \bar{y})dx\int_{y\in B} f(Y = y)dy}{P(Y \in B)}\\
&= \frac{\sup_{y\in B} P(X \in A|Y = y)P(Y \in B)}{P(Y \in B)}\\
&= \sup_{y\in B} P(X \in A|Y = y),
\end{aligned}$$

□

Proof of Proposition 5. Now let's proceed with the proof of Proposition 5. We consider the supremum case. The infimum follows analogously. Let $\mathbf{y}(x)$ be a

normal random variable with mean $\mu(x)$ and covariance matrix $\Sigma(x)$. Then, we have

$$\begin{aligned}
&\sup_{x\in T}\int_S \mathcal{N}(\bar{f}|\mu(x),\Sigma(x))d\bar{f}\\
=&\sup_{x\in T} P(\mathbf{y}(x)\in S)\\
=&\sup_{x\in T} P(\wedge_{i=1}^{C} k_i^1 \leq \mathbf{y}_i(x) \leq k_i^2)\\
=&\sup_{x\in T}\prod_{i=1}^{C} P(k_i^1 \leq \mathbf{y}_i(x) \leq k_i^2|\wedge_{j=i+1}^{C} k_j^1 \leq \mathbf{y}_j(x) \leq k_j^2)\\
&\quad\text{(By Lemma 3)}\\
\leq&\sup_{x\in T}\prod_{i=1}^{C}\sup_{f\in S^{i+1}} P(k_i^1 \leq \mathbf{y}_i(x) \leq k_i^2|\wedge_{j=i+1}^{C}\mathbf{y}_j(x)=f_{j-i})\\
\leq&\prod_{i=1}^{C}\sup_{x\in T, f\in S^{i+1}} P(k_i^1 \leq \mathbf{y}_i(x) \leq k_i^2|\wedge_{j=i+1}^{C}\mathbf{y}_j(x)=f_{j-i})
\end{aligned}$$

Notice that for each $i\in\{1,...,C\}$, $P(k_i^1 \leq \mathbf{y}_i(x) \leq k_i^2|\wedge_{j=i+1}^{C}\mathbf{y}_j(x)=f_{j-i})$ is the integral of a uni-dimensional Gaussian random variable, as a Gaussian random variable conditioned to a jointly Gaussian random variable is still Gaussian. □

Proposition 5 reduces the computation of the bounds for the multi-class case to a product of extrema of univariate Gaussian distributions for which Proposition 2 can be iteratively applied. Analogously to what we discussed for the binary case, the resulting bound can be refined through a branch and bound algorithm to ensure convergence up to any desired tolerance $\epsilon > 0$.

3.7 Experimental Results

3.7.1 Experimental Set Up

We employ our methods to experimentally analyse the robustness profile of GPC models in adversarial settings[4]. In Section 3.7.2 we check for adversarial local safety in neighbourhoods around specific test points, and compare the guarantees we provide with an adversarial attack for GPC models (GPFS attacks, Grosse

[4]Code: *https://github.com/arblox/robustness-GPclass*

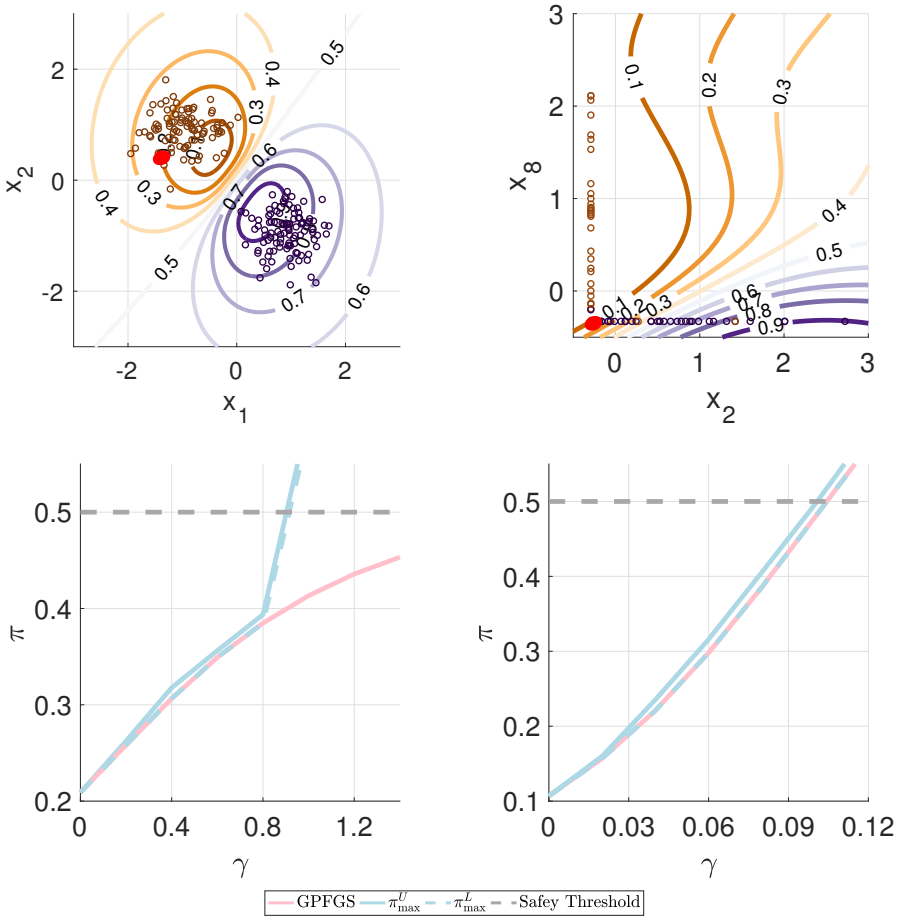


Figure 3.2: First row: Contour plot and test points for Synthetic2D (left); projected contour plot and test points for 2 dimensions of SPAM (right, dimensions 2 and 8 as selected by L_1-penalised logistic regression); red dots mark selected test points. **Second row**: Safety analysis for the two selected test point. Shown are the upper and lower bounds for tolerance $\epsilon = 0.02$ on $\pi_{\max}(T)$ (solid and dashed blue curves) and the GPFGS adversarial attack (pink curve). The true values of $\pi_{\max}(T)$ are guaranteed to be between the solid and dashed blue curves.

et al. (2018)). In Section 3.7.3 we empirically compare different training regimes for Bayesian inference with GPC, analysing the effect of the approximate Bayesian techniques used and hyperparameter optimisation. Finally, in Section 3.7.4 we show how adversarial local robustness measures can be used for interpretability analysis, and compare our results with those obtained by the well-established interpretability benchmark LIME (Ribeiro et al., 2016). We give results for three data sets: (i) Synthetic2D, generated by shifting a two-dimensional standard-normal either along the first dimension (class 1) or the second one (class 2); (ii) the SPAM data set

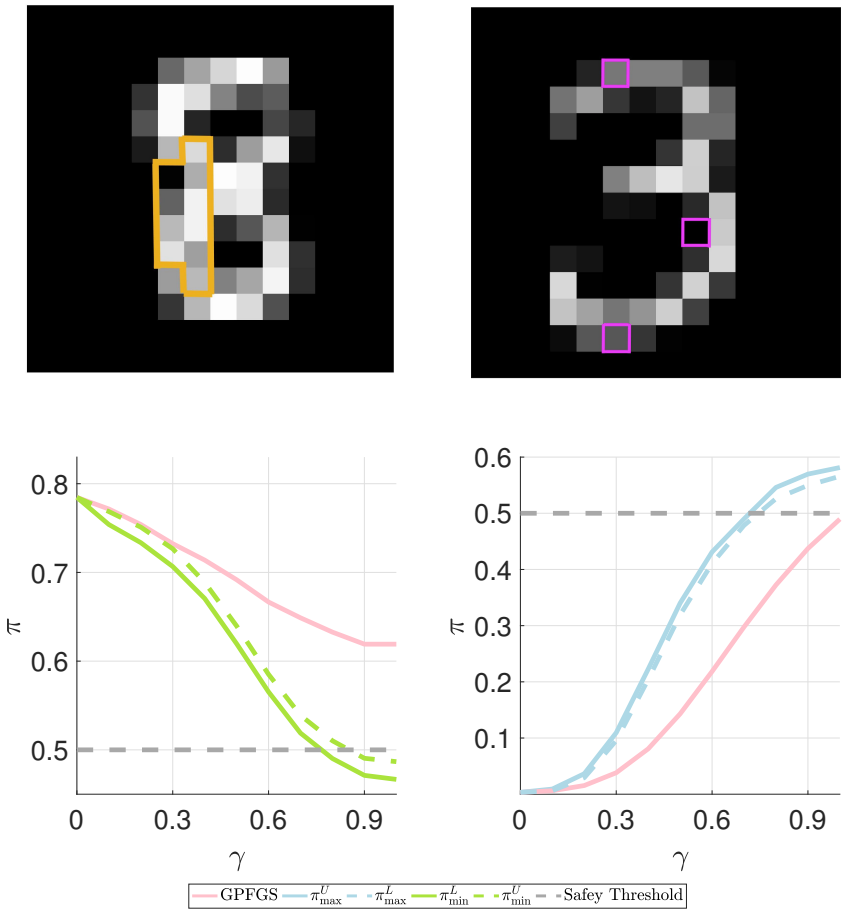


Figure 3.3: First row: Sample of 8 from MNIST38 along with 10 pixels selected by SIFT (left) and sample of 3 from MNIST38 along with the 3 pixels that have the shortest lengthscales after GPC training (right). **Second row**: Safety analysis for the two images. Shown are the upper and lower bounds for $\epsilon = 0.02$ on either $\pi_{\max}(T)$ or $\pi_{\min}(T)$ (solid and dashed blue resp. green curves) and the GPFGS adversarial attack (pink curve). The true values of $\pi_{\max}(T)$ or $\pi_{\min}(T)$ are guaranteed to be between the solid and dashed blue (green) curves.

(Dua and Graff, 2017); (iii) a subset of the MNIST data set (LeCun, 1998) with classes 3 and 8 (MNIST38) and a subset with classes 3, 5 and 8 (MNIST358). As we are interested in worst-case guarantuees, we use the L_∞-norm induced metric when defining γ-balls around test points, as by definition these contain the γ-balls of any other L-norm induced metric. For scalability, results for MNIST38 are given for feature-level analysis (as done in Ruan et al. (2018) for deep networks). Namely, we analyse either salient patches detected by SIFT (Lowe, 2004) or we select the relevant pixels corresponding to the shortest length-scales of the optimised ARD kernel (see Section 2.1.3). Importantly, our approach is independent of the

particular feature selection method chosen.

Data Sets

Synthetic2D, our synthetic two-dimensional data set contains 1,200 points, of which 50 % belong to class 1 and 50 % belong to class 2. The points were generated by shifting draws from a two-dimensional standard-normal random variable by 5, either along the first dimension (class 1) or along the second dimension (class 2). Subsequently, we normalise the data by subtracting its mean and dividing by its standard deviation.

SPAM is a binary data set that contains 4,601 samples, of which 60% are benign. Each sample consists of 54 real-valued and three integer-valued features. However, identical or better prediction accuracies can be achieved with models involving only 11 of those 57 variables, among them e.g. the frequency of the word 'free' in the email, the share of $ signs in its body, or the total number of capital letters, which is why we only use these 11 selected variables. We normalise the data by subtracting its mean and dividing by its standard deviation.

MNIST38 contains 10,398 samples of images of handwritten digits, of which roughly 50 % are 3s and 50 % are 8s. Each sample consists of a 28×28 pixel image in gray scale (integer values between 0 and 255) which following convention, we rescale by dividing by 255. For better scalability we then downsample to 14×14 pixels.

MNIST358 contains the subset of MNIST consisting of images with label 5 in addition to MNIST38, resulting in a data set of which roughly 36 % are 3s, 31 % are 5s and 34 % are 8s. Each sample consists of a 28×28 pixel image in gray scale (integer values between 0 and 255) which following convention, we rescale by dividing by 255. For better scalability we then downsample to 14×14 pixels.

GPC Model Training

For the binary experiments, we use 1,000 randomly selected points as training set and 200 randomly selected points as test set. For the multiclass experiments, scalability of GPs is even more of an issue so we just work with 500 randomly selected points as training set. For the GP training of binary classification problems, we use the GPML

package for Matlab. For the GP training of multiclass classification problems, we use the GPstuff package (since GPML does not support multiclass classification).

For the safety verification experiments in Section 3.7.2, we used a GPC model with a probit likelihood function and the Laplace approximation for the posterior. For the synthetic 2D data, the number of epochs (marginal likelihood evaluations) performed during hyper-parameter optimisation was restricted to 20. For the SPAM data, it was restricted to 40. Finally for the attacks on MNIST38 it was restricted to 10 (and 20). This way we achieved 100%, 93% and 97% (98%)[5] test set accuracies on the three data sets respectively. The chosen number of epochs reflect the minimum number of epochs (in intervals of 10) required to achieve such test set accuracies for each task.

For the robustness experiments in Section 3.7.3, we vary the specifications of training to analyse their effect on GPC robustness.

For the interpretability experiments in Section 3.7.4, we use a multiclass GPC model with softmax likelihood function and the Laplace approximation for the posterior. We limit the number of iterations performed during hyper-parameter optimisation to 10, achieving a test set accuracy of 93%.

Implementation of Benchmarks

The code for the GPFGS attacks as well as LIME was implemented by us in Matlab according to the original Python code provided by the authors. Below, we briefly outline both concepts on a high level.

GPFGS Attacks. GPFGS attacks have been developed by Grosse et al. (2018) as the GP analogue of the well-known fast gradient sign method (FGSM) attack for neural networks (Goodfellow et al., 2014). Given a perturbation budget γ, FGSM attacks a model at test point x^* by perturbing each dimension by γ in the direction of its gradient at x^*. Since for GPs, the integral in Eq. (3.1) makes calculating the

[5]Surprisingly, despite the high input dimensionality of $14 \times 14 = 196$, the GPC achieved such high test accuracies after such short training. We believe this is due to the fact that many pixels stay constant (black) for all training points, reducing the effective dimensionality of the problem.

gradient on π hard, Grosse et al. (2018) propose to base the attack on the gradient of the latent mean function $\mu(\cdot)$ instead. The GPFGS attack is thus defined by

$$x_{adv} = x^* + \gamma \operatorname{sign}(\nabla_x \mu(x^*)) \tag{3.25}$$

LIME. LIME (Local Interpretable Model-agnostic Explanations) is a model-agnostic interpretability technique that relies, in its most commonly used form (Ribeiro et al., 2016), on linear local approximations to derive interpretable metrics to explain model behaviour (the parameters of the linear approximation). By randomly sampling a number of local perturbations $\{z^{(i)}\}_{1\leq i\leq s}$ around a data point x and evaluating the model f under inspection at each of these points, one gets a local training set $\mathcal{D}_s = \{(z^{(i)}, f(z^{(i)}))\}_{1\leq i\leq s}$. For each point in $\mathcal{D}_s$, a weight $w_x(z^{(i)})$ is calculated based on its distance to the original data point x (smaller distance resulting in larger weight). Then, a linear model g is fitted minimising the weighted mean squared error w.r.t. $\mathcal{D}_s$. By heavy L_1-regularisation, the resulting linear model coefficients can then be used as interpretability in the sense of sensitivity metrics - a positive coefficient w.r.t. a specific class meaning that increasing the corresponding input dimension would encourage a classification decision in favour of that particular class.

3.7.2 Adversarial Local Safety

We depict the local adversarial safety results for four points manually selected for illustratory purposes from the Synthethic2D, SPAM, and MNIST38 data sets in Figures 3.2 and 3.3. To this end, we set $T \subseteq \mathbb{R}^d$ to be a L_∞ $\gamma-$ball around the chosen test point and iteratively increase γ (x-axis in the second row plots), checking whether there are adversarial examples in T. Namely, if the point is originally assigned to class 1 (respectively class 2) we check whether the minimum classification probability in T is below the decision boundary threshold, that is, if $\pi_{\min}(T) < 0.5$ (resp. $\pi_{\max}(T) > 0.5$). We compare the values provided by our method (blue solid and dashed line for class 2, green solid and dashed line for class 1) with GPFGS (Grosse et al., 2018), a gradient based heuristic attack for GPC

(pink line).

Naturally, as γ increases, the neighborhood region T becomes larger, hence the confidence for the initial class can decrease. Interestingly, while our method succeeds in finding adversarial examples in all cases shown (i.e. both the lower and upper bound on the computed quantity cross the decision boundary), the heuristic attack fails to find adversarial examples in the Synthetic2D and in the MNIST38 case. This happens as GPFGS builds on linear approximations of the GPC function, hence failing to find solutions to Eq. (3.3) when there are non-linearities. In particular, near the point selected for the Synthetic2D data set (red dot in the contour plot) the gradient of the GPC points away from the decision boundary. Hence, no matter the value of γ, GPFGS will not go above 0.5 in this case (pink line of the bottom-left plot). On the other hand, for the SPAM data set, the GPC model is locally linear around the selected test point (red dot in top right contour plot). As such, following the gradient around it will lead to values close to optimality. Interestingly, the MNIST38 examples (Figure 3.3) provide results analogous to those of Synthetic2D. While our method finds adversarial examples on both occasions, GPFGS fails to do so (even with $\gamma = 1.0$ which is the maximum region possible for the rescaled pixel values). This calls for caution when using heuristic methods to estimate the adversarial robustness of GPC methods in real-world data sets, as the four chosen examples show that for some points and GPC models, this can lead to a false sense of safety.

3.7.3 Adversarial Local Robustness

We evaluate the empirical distribution of δ-robustness (see Definition 7) on 50 randomly selected test points for each of the three data sets considered. That is, given T, we compute $\delta = \pi_{\max}(T) - \pi_{\min}(T)$. Notice that a smaller value of δ implies a more robust model. In particular, we analyse how the GPC model robustness is affected by the training procedure used. We compare the robustness obtained when using either the Laplace or the EP posterior approximations technique. Further, we

investigate the influence of the number of marginal likelihood evaluations (epochs) performed during hyper-parameter optimisation on robustness.

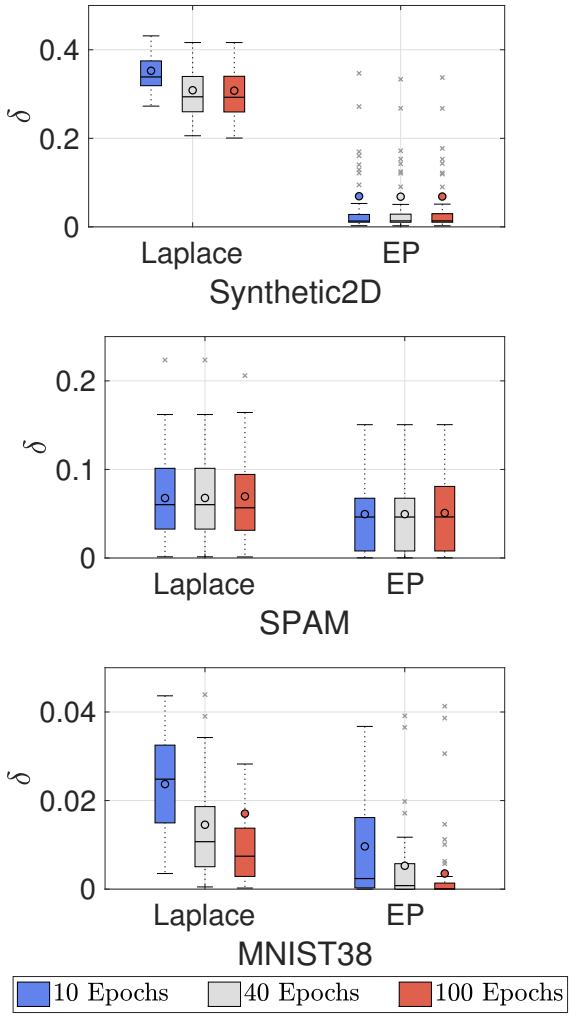


Figure 3.4: Boxplots for the distribution of robustness on the three data sets, comparing Laplace and EP approximation (a smaller δ implies a more robust model).

Results are depicted in Figure 3.4, for 10, 40 and 100 hyper-parameter optimisation epochs. Note that the analyses for the MNIST38 samples are restricted only to the most influential SIFT feature, and thus δ values for MNIST38 are smaller in magnitude than for the other two data sets (for which all the input variables are simultaneously changed). Interestingly, this empirical analysis demonstrates that GPCs trained with EP are consistently more robust than those trained using Laplace (see also Kuss et al. (2005), who find EP apprixmations superior on clean train and test data). In fact, for both Synthetic2D and MNIST38, EP yields a model

about 5 times more robust than Laplace. For SPAM, the difference in robustness is the least pronounced. We compare Laplace and EP posterior approximations with that made by Hamiltonian Monte Carlo (HMC) - that is, as in Minka (2001) we use HMC as gold standard. The empirical distances found on the posterior approximation w.r.t. HMC are on average as follows (smaller values are better): (i) Synthetic2D - Laplace: 1.04, EP: 0.14; (ii) SPAM - Laplace: 0.35, EP: 0.32; (iii) MNIST38 - Laplace: 0.52, EP: 0.32. This shows a correlation between the robustness and the posterior approximation quality in the data sets considered. These results quantify and corroborate for GPCs the hypothesis that a more refined approximation of the posterior seems beneficial for model adversarial robustness (Cardelli et al., 2019a). Note that for a full confirmation of this finding it would be necessary to also analyse the robustness profile of the HMC approximated GPC, for which our methods unfortunately don't work (as we need a Gaussian closed form approximation). Interestingly, the values of δ decrease as the number of training epochs increases, thus robustness improves with training epochs. This is in contrast to what is observed in the deep learning literature (Tsipras et al., 2018). More training in the Bayesian settings may imply better calibration of the latent mean and variance function to the observed data.

3.7.4 Sensitivity Analysis

Finally, we show how adversarial robustness can be used for sensitivity analysis for GPC models, which can be used as interpretability tool for machine learning models (Ribeiro et al., 2016). We provide comparison with pixel-wise LIME (Ribeiro et al., 2016), a model-agnostic sensitivity / interpretability technique that relies on local linear approximations. Given a test point x^* consider the one-sided intervals $T^i_\gamma(x^*) = [x^*, x^* + \gamma e_i]$ (with e_i being the vector of 0s except for 1 in dimension i). We compute how much the maximum and minimum values can change over the one-sided intervals in both directions:

$$\begin{aligned}\boldsymbol{\Delta}^i_\gamma(x^*) = &\left(\pi_{\max}(T^i_\gamma(x^*)) - \pi_{\max}(T^i_{-\gamma}(x^*))\right) \\ &+ \left(\pi_{\min}(T^i_\gamma(x^*)) - \pi_{\min}(T^i_{-\gamma}(x^*))\right).\end{aligned}$$

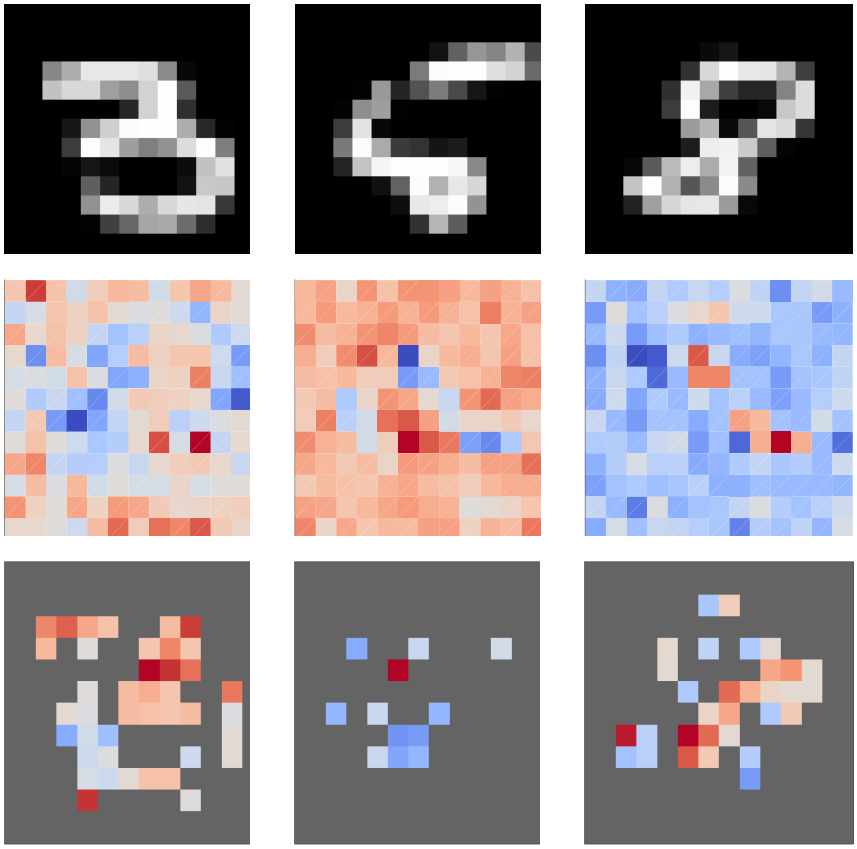


Figure 3.5: First row: Samples selected from MNIST358. **Second row**: Interpretability metric estimation using our method. **Third row**: Results obtained using LIME.

If increasing the value of dimension i makes the model favour assigning lower class probabilities, we would expect this value to be negative and vice versa. Intuitively, this provides a non-linear generalisation of numerical gradient estimation which resembles exactly the metric used by Ribeiro et al. (2016) as γ shrinks to 0 or if the model considered is linear. Global estimation measures can be computed by estimating the expected value of $\mathbf{\Delta}^i_\gamma(x^*)$ with x^* sampled from a test set, i.e. by computing $\mathbf{\Delta}^i_\gamma = \frac{1}{M}\sum_{j=1}^{M}\mathbf{\Delta}^i_\gamma(x^j)$. Notice however that, since our method relies on the analytic form of the inference equations of GPs (rather than being model-agnostic as LIME is), we are able to formally bound these quantities, which allows us to have guarantees over the interpretability results.

In the next two paragraphs, we first conduct local sensitivity analyses on MNIST358 and then graphically demonstrate why linear approximation can be misleading for global sensitivity analysis in the 2D-synthetic and SPAM data set.

Local Sensitivity for MNIST358. Figure 3.5 shows the results for three samples selected from MNIST358 (top row), with the heat maps depicting the results of our method (second row) and those for LIME (third row, greyed out pixels are marked as irrelevant by LIME). The colour gradient varies from red (positive impact, pixel value increase causing increased class probability of shown digit) to blue (negative impact, pixel value increase decreasing the class probability). For digit 3, our method obtains for example a contiguous blue patch on the left. Increasing the values of these pixels would modify the 3 into an 8. Indeed, when whitening the pixels of the blue patch, the class 3 probability assigned by the model decreases from 0.58 to 0.40. Similarly, for digit 5, our methods identify a blue patch that would change the 5 into an 8 and again the GPC model indeed lowers its class 5 probability when the patch is whitened. Similarly, for digit 8, our method identifies a blue patch of 3 pixels towards the top left, which would turn it into something resembling digit 3 if whitened. This happens indeed to the classification probabilities, which increase from 0.19 to 0.22 for class 3 while decreasing from 0.54 to 0.5 when adding 0.5 to only these 3 pixels. The patterns found by LIME in these examples are either not as clear or different. We argue that the different outcomes are due to the fact that, unlike LIME, our metric does not rely on the assumption of local linearity of the prediction model, which we inspect further in the next paragraph.

Global Sensitivity for the Binary Data Sets. We perform global sensitivity analysis on the GPC models trained on the Synthetic2D and SPAM data sets, using 50 random test points. The results are shown in Figure 3.6. For Synthetic2D (top row), LIME suggests that a higher probability of belonging to class 1 (depicted as the direction of the arrow in the plot) corresponds to lower values along dimension 1 and higher values along dimension 2. As can be seen in the corresponding contour plot in Figure 3.2 (top left), the exact opposite is true however. LIME, being built on local linearity approximations, fails to take into account the global behaviour of the GPC. When using a small value of γ our approach obtains similar results to LIME. However, with $\gamma = 2.0$ the global relationship between input and output

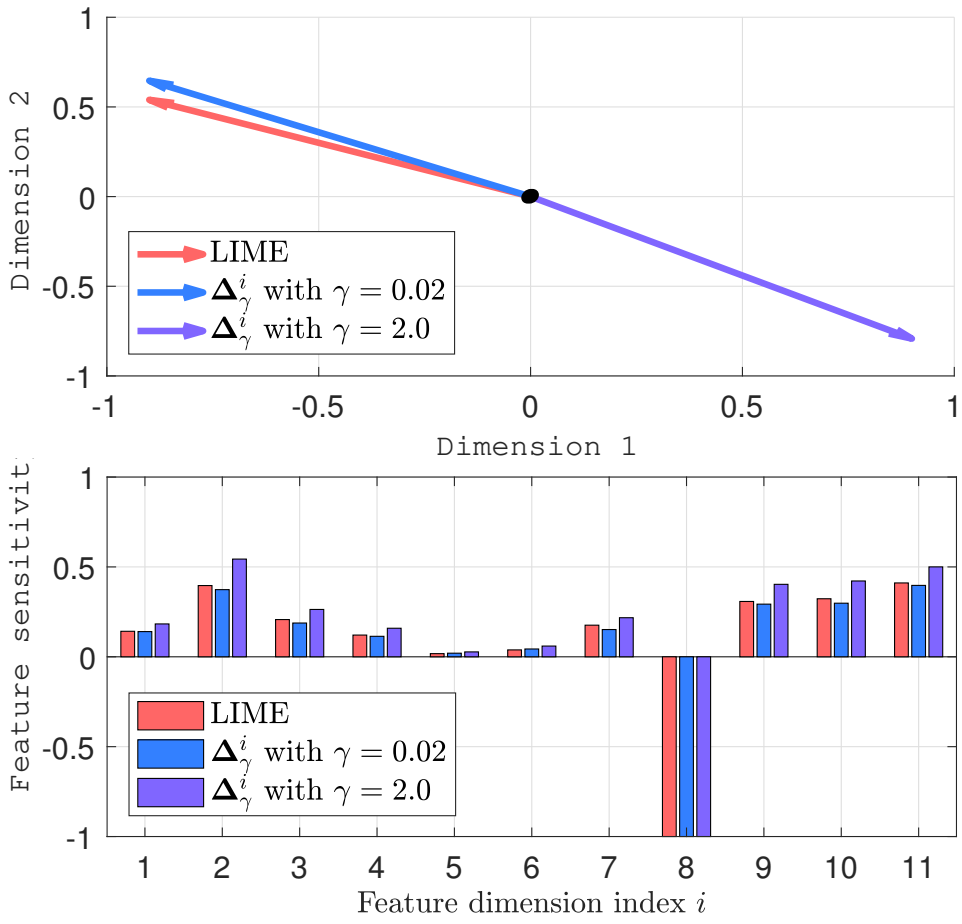


Figure 3.6: Global feature sensitivity analysed by LIME and our metric $\mathbf{\Delta}^i_\gamma$. All values normed to unit scale for better comparison. **Top:** Results for Synthetic2D data set mapped out on plane. **Bottom:** Results for SPAM data set.

values is correctly captured. For SPAM, on the other hand (Figure 3.6, bottom), due to linearity of the data set and GPC, a local analysis correctly reflects the global picture. In this case, LIME is probably a better choice for model interpretation as it takes less time to calculate its metrics.

3.7.5 Runtime Analysis

In this section we empirically analyse the CPU time required for convergence of Algorithm 1 in the MNIST38 data set. For the first 50 test points and a $\gamma-$ball T of dimensionality d, we calculated $\pi_{\max}(T)$ up to a pre-specified error tolerance ϵ. We use $\gamma = 0.125$ and $\gamma = 0.25$, corresponding to up to 50% of the normalised input domain. All runtimes analysed below were obtained on a MacBook Pro with a 2.5

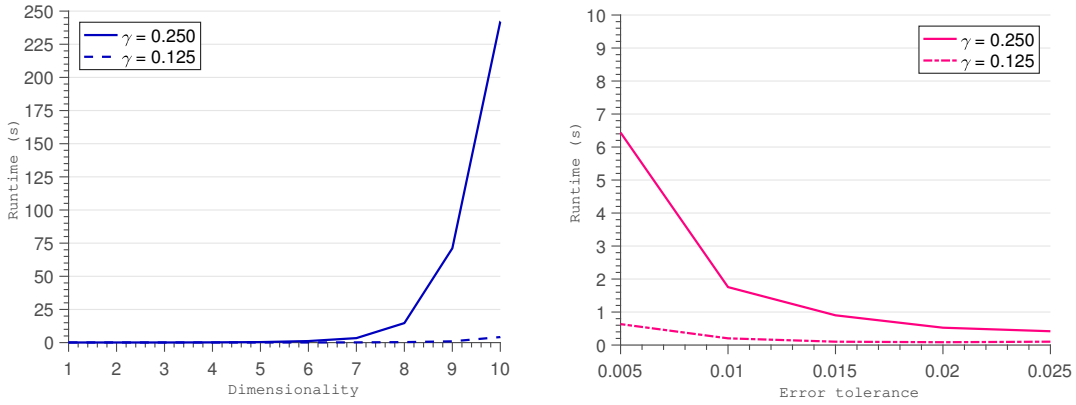


Figure 3.7: Average runtimes of Algorithm 1 to calculate $\pi_{\max}(T)$ up to specified error tolerance ϵ among the first 50 test points of MNIST38. **Left**: Average runtimes for increasing number of dimensions at $\epsilon = 0.025$. **Right**: Average runtimes for different values of ϵ with number of dimensions $d = 5$.

GHz Intel Core i7 processor and 16GB RAM running on macOS Mojave 10.14.6.

Runtime Depending on Dimension of Compact Subset

First, we analysed the effect of increasing d, by fixing $\epsilon = 0.025$ and increasing the number of pixels selected by SIFT to define T from 1 to 10. The results are shown in terms of average runtime in Figure 3.7 on the left. For $\gamma = 0.25$, we can observe the exponential behaviour of the computational complexity in terms of number of dimensions, as the runtime quickly grows from below 5 seconds to almost 250 seconds beyond 7 dimensions. However, for $\gamma = 0.125$ the exponential curve is to be shifted further to the right, as still for 10 dimensions Algorithm 1 terminates in only a few seconds. Given that for $\gamma = 0.125$, T spans up to 25% of the input domain (on the selected pixels), we consider this quite fast.

Runtime Depending on Error Tolerance

Second, we analysed the effect of the error tolerance ϵ, by calculating the bounds for each $\epsilon \in \{0.005, 0.01, 0.015, 0.02, 0.025\}$ with the number of pixels selected by SIFT (i.e. d) fixed to 5. The results are shown in Figure 3.7 on the right. The behaviour seems to be roughly inversely exponential this time with lower error tolerance ϵ naturally demanding higher runtimes. In practice, one would seldom

expect to require precision of $\epsilon < 0.01$ though, at which point Algorithm 1 still terminates in under 2 seconds on average even for $\gamma = 0.25$.

3.8 Improved Bounds for Binary Classification

As we see, the runtime can be problematic for high dimensional inputs due to the worst-case exponential computational complexity of the branch-and-bound algorithm. This is even more true when non-probit likelihoods (e.g. logistic) are used in situations where lower bounding the latent variance is difficult. In such cases, the sum in Proposition 1 becomes a very crude over-approximation as $\Sigma_T^L \approx 0$ means that for each part of the sum, a different point mass can be placed to yield worst-case behaviour. In our experiments, this was for example the case in the experiments on SPAM, which seems so easy to learn that the predictive variance is low in general (see contour plot in Figure 3.2). To deal with this problem, we have derived a complementary bound for general likelihood functions that relies on the Lipschitz continuity of the likelihood and uses the Wasserstein distance between two univariate Gaussians to get a closed-form bound instead of the bound proposed in Proposition 1.

3.8.1 Tighter Outer Bounds via Wasserstein Distance Bounding

Consider the logistic likelihood $\sigma(x) = 1/(1 + e^{-x})$ and the binary GPC with predictive distribution $\pi(x^*|\mathcal{D}) = \int \sigma(\bar{f})q(f(x^*) = \bar{f}|\mathcal{D})d\bar{f}$ with Gaussian approximation $q(f(x^*) = \bar{f}|\mathcal{D})$ (e.g. obtained through Laplace or EP) for which upper and lower bounds to mean μ and variance Σ are known for the compact set T under consideration. It can be easily shown that the Lipschitz constant of σ is $1/4$ (for the probit likelihood function it would be $1/\sqrt{2\pi}$ for example). Then we can bound

the maximum change of π in T by the Wasserstein distance as follows:

$$\begin{aligned} 4[\pi(x^*) - \pi(x)] &= \int 4\sigma(\bar{f})[q(f(x^*) = \bar{f}|\mathcal{D}) - q(f(x) = \bar{f}|\mathcal{D})]d\bar{f} \qquad (3.26) \\ &\leq \sup_{g:\mathrm{Lip}(g)\leq 1} \int g(\bar{f})[\mathcal{N}(\bar{f}|\mu(x^*), \Sigma(x^*)) - \mathcal{N}(\bar{f}|\mu(x), \Sigma(x))]d\bar{f} \\ &= W_1(q_{x^*}, q_x) = \int_0^1 |Q_{x^*}^{-1}(z) - Q_x^{-1}(z)|dz \\ &\leq E[|q_{x^*} - q_x|] \leq \bar{\Sigma}\sqrt{2/\pi}\exp\left(-\bar{\mu}^2/2\bar{\Sigma}^2\right) - \bar{\mu}(1 - \Phi(\bar{\mu}/\bar{\Sigma})) \end{aligned}$$

where Q^{-1} depicts the inverse cdfs and $\bar{\mu} = \mu(x^*) - \mu(x)$ and $\bar{\Sigma} = \Sigma(x^*) - \Sigma(x)$. For a worst case bound over all $x \in T$, we would take the max on both sides and divide by 4. Maximising $\bar{\mu}$ and $\bar{\Sigma}$ can be done easily if lower and upper bounds of mean and variance function over T are known (which we can calculate with our algorithm).

Unfortunately, this new bound can not be universally tighter than the existing bound (old bound) from Propositions 1 and 2, as becomes clear from the following example:

Example 1 (Example of new bound being looser)**.** *Let T be such that we have $\mu_T^L = 10$, $\mu_T^U = 11$, $\Sigma_T^L = 0.5$, $\Sigma_T^U = 2.0$, and $x^* \in T$ be the correctly classified test point with $\mu(x^*) = 10.5$ and $\Sigma(x^*) = 1.4$. Calculating a lower bound on the minimum with the old bound which relies on discretising/partitioning the integral and computing lower bounds over each interval yields a lower bound of ≈ 1, which is natural given that the smallest possible mean value μ_T^L is still giving a value for σ of basically 1 and the variance is such that even with the highest variance Σ_T^U, any value of l that is smaller than 0.99 is more than 5 standard deviations away from that smallest mean value. However, the new bound will give a lower bound on the minimum of ≈ 0.75, which is the worst case Wasserstein-1 distance between q_{x^*} and q_x for $x \in T$ given the bounds on mean and variance as above. This is because the by taking* sup *over all $1-$Lipschitz function in Eq. (3.26), we over-approximate a lot in this special case where σ is basically constant in the region where q_{x^*} and q_x put probability mass.*

However, the new bound can sometimes be closer than the old one, when T is such that q_{x^*} and q_x can put significant probability mass around 0, i.e. where σ changes the most (as is for example the case when $\Sigma_T^L \approx 0$):

Example 2 (Example of new bound being tighter)**.** *Let T be such that we have $\mu_T^L = 0.1$, $\mu_T^U = 0.8$, $\Sigma_T^L = 0.3$, $\Sigma_T^U = 2.0$, and $x^* \in T$ be the correctly classified test point with $\mu(x^*) = 0.5$ and $\Sigma(x^*) = 1.4$. Calculating a lower bound on the minimum with the old bound which relies on discretising/partitioning the integral and computing lower bounds over each interval yields a lower bound of ≈ 0.25. However, the new bound will give a lower bound on the minimum of ≈ 0.375, which is the worst case Wasserstein-1 distance between q_{x^*} and q_x for $x \in T$ given the bounds on mean and variance as above.*

Figure 3.8 sums up these two examples in a qualitative comparison of new and old bound. As can be seen, for fixed $\mu(x^*)$, μ_T^L, μ_T^U, different combinations of Σ_T^L and Σ_T^U can give a different outcome of the comparison. In particular, the closer Σ_T^L is to 0, the better performs the new bound in comparison to the old bound. Given this complementary nature of the new bound, we thus propose computing both bounds (thanks to the close-form nature of the new bound this does not significantly increase compute) and choose the tighter bound at each iteration of the branch-and-bound algorithm in Algorithm 1. In the following, we show how this can help speeding up convergence for SPAM.

3.8.2 Experiments Using the New Bound

To see the effect of the combined bound, we return to the SPAM data set and train a GPC model with logistic likelihood function. We then run Algorithm 1 on 10 randomly chosen test points (namely, the first 10 test points in the data set as we load it) for 5,000 iterations and give the outer bound values obtained over a fixed $\gamma-$ball around each test point with both the old procedure (old bound only, in brackets) and the new proposed procedure combining old and new bound. The results are shown in Table 3.1.

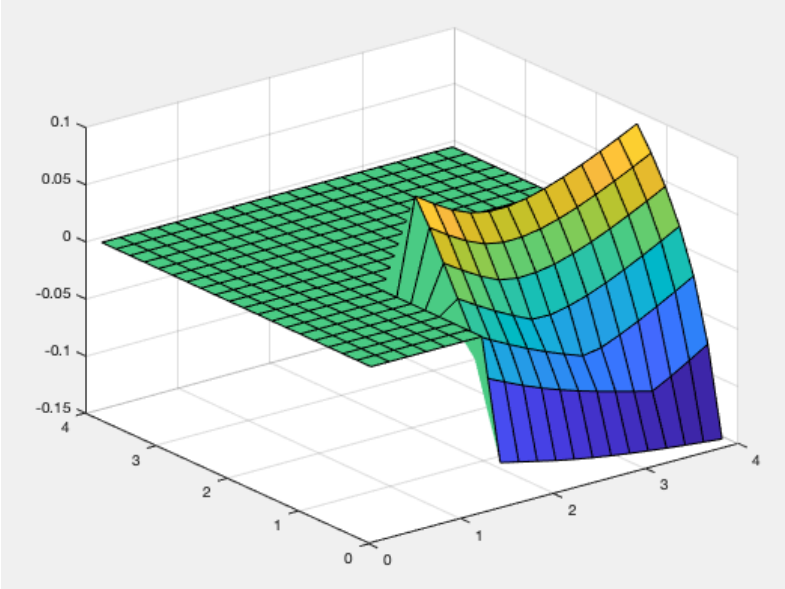


Figure 3.8: Difference of old bound vs. new bound for $\pi_{\min}(T)$, i.e. $\pi^{old}_{\min}(T) - \pi^{new}_{\min}(T)$, depending on values of Σ^L_T (axis to left rear) and Σ^U_T (axis to right rear) for fixed values of $\mu(x^*) = 0.3, \Sigma(x^*) = 1.4, \mu^L_T = 0.1$ and $\mu^U_T = 0.8$. Note that the comparison by definition of Σ^L_T and Σ^U_T) only makes sense where $\Sigma^L_T < \Sigma(x^*) = 1.4 < \Sigma^U_T$), and has been set to 0 otherwise.

Table 3.1: Experiment on GPC with logistic likelihood trained to 98.5% accuracy on SPAM data set using Laplace approximation. We consider 10 random test points and for each test point x^* inspect the GPC values over $T = \mathcal{B}^\infty_\gamma(x^*)$ with $\gamma = 0.03$. We show the bound values obtained with a maximum of 5,000 iterations inside Algorithm 1 with a combination of new and old bound. Values in brackets show bounds obtained for the same number of maximum iterations when using only the old bound inside the branch and bound. Since Algorithm 1 is deterministic, we only show results of a single run.

Testpoint	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Upper bound	0.94	0.30	0.17	0.44	0.11	1.0	1.0	1.0	0.25	0.22
on $\pi_{\max}(T)$	(0.94)	(0.49)	(0.17)	(0.65)	(0.15)	(1.0)	(1.0)	(1.0)	(0.35)	(0.28)
$\pi(x^*\|\mathcal{D})$	0.87	0.16	0.03	0.30	0.01	0.99	0.92	0.99	0.09	0.07
Lower bound	0.82	0.06	0.00	0.15	0.0	0.91	0.80	0.89	0.01	0.02
on $\pi_{\min}(T)$	(0.82)	(0.06)	(0.00)	(0.12)	(0.00)	(0.50)	(0.50)	(0.50)	(0.01)	(0.02)

As one can see, in 8 out of the 10 cases, the approach using a combination of old and new bound yields tighter bounds in fewer branch and bound iterations than the approach only using the old bound proposed in Section 3.4.2.

3.9 Discussion

We presented a method for computing, for any compact set of input points, the class probability range of a GPC model across all points in that set, up to any precision $\epsilon > 0$. This allows us to analyse robustness and guarantee safety against adversarial attacks, which we have demonstrated on multiple data sets and approximate Bayesian inference techniques for GPC models. In particular, we saw that such guarantees can be necessary for GPC models trained on real data, in instance where empirical approaches based on adversarial attacks fail to find adversarial examples, resulting in a false sense of security. Also, we found evidence that the adversarial robustness of GPC models increases with the quality of the approximation of the true posterior.

A big drawback of the presented method is its computational complexity. In practice, despite the incremental research resulting in tighter bounds for general likelihoods presented in the previous section, the worst-case exponential computational complexity of the branch-and-bound approach (w.r.t. the number GPC input dimensions) still prohibits the use of our method for large compact sets when the dimensionality is higher than ≈ 20. Given the hardness of the problem, it is however questionable if a fundamentally better worst-case computational complexity (e.g. polynomial) could ever be achieved. Thus, stronger adversarial attacks for GPCs inspired by the DNN literature could still help to get a sense of adversarial robustness in high dimensional settings, if formal guarantees are not needed.

4

Does Being More Bayesian Increase the Adversarial Robustness of Bayesian Neural Networks?

4.1 Introduction

In the previous chapter, we have analysed how robust GPC models are against adversarial attacks. In this chapter, we now turn our focus to BNNs, since they have gained importance in many real-life applications (LeCun et al., 2015). In practice, most of the deployed BNNs rely on MAP approximations of the posterior in the form of weight decay, as those are cheaper to obtain both in terms of computational complexity and memory usage and yield good predictive performance on clean test data (Simonyan and Zisserman (2014); He et al. (2016), for the equivalence of weight decay and MAP approximation see Section 2.1.2). While this good predictive performance quickly degrades in light of adversarial attacks, most existing work on the adversarial robustness of BNNs focusses on BNNs with MAP approximated posteriors in light of their dominant prevalence in both research and real-life applications (Madry et al., 2017; Carlini and Wagner, 2017b; Croce and Hein, 2020). Yet, in the experimental section of the previous chapter (Section 3.7.3), we found evidence that a better approximation of the true posterior distribution might

actually lead to increased adversarial robustness, at least for GPC models. While this observation was made for GPC models, the relationships between GPs and BNNs highlighted in Section 2.1.3 gives us hope that it might also be the case for BNNs.

In this chapter, we thus aim to explore the effect of a better posterior approximation for BNNs on their adversarial robustness. To this end, we compare the adversarial robustness of BNNs with HMC based posterior approximations, the most accurate posterior approximations known to us, to the robustness of identical BNNs for which the commonly used MAP approximation has been used to approximate the posterior.

4.2 Related Work

The question whether HMC approximation based BNNs are more robust against adversarial attacks than MAP approximated BNNs (we also refer to these as deterministic NNs), has to the best of our knowledge not been directly investigated so far. Yet, there have been a number of works that indirectly investigate this question and also indicate, if anything, that one would expect a stronger adversarial robustness from a HMC approximation based BNN. We briefly present them below.

4.2.1 Works on Bayesian Neural Networks

Liu et al. (2018a) propose a MFVI approximated posterior inference to increase the adversarial robustness of an adversarially trained BNN over its MAP approximation. Using empirical estimations of the robustness by analysing the BNNs vulnerability to PGD attacks (Madry et al., 2017), they obtain impressive results on the increase in adversarial robustness achievable through the better posterior approximation compared to MAP. However, Zimmermann (2019) shows that these results are misleading, as the empirical robustness stems from gradient obfuscation that can be easily circumvented. Namely, Liu et al. (2018a) sample only one single parameter vector θ from the approximated posterior $p(\theta|\mathcal{D}_n)$ at each iteration of the PGD attack. This makes the gradient in the PGD attack extremely noisy and misleading, as it effectively corresponds to a one sample MC estimate of the desired quantity.

A simple increase in samples per iteration can massively improve the quality of the MC estimate (as it lowers its variance) and as a result the power of the PGD attack, leading Zimmermann (2019) to inconclusive results that at best slightly favour the hypothesis of higher adversarial robustness through the better posterior approximation.

Carbone et al. (2020) show that in the limit of an overparametrised BNN with uninformative prior (e.g. uniform or very high variance Gaussian), the expected gradient w.r.t. to inputs is 0 on the data manifold for the true posterior, making gradient based attacks such as FGSM and PGD at least partially ineffective. In experiments on MNIST and FashionMNIST, they show that this empirically results in a higher robustness of HMC and MFVI based BNNs against FGSM attacks than typically achieved MAP based ones. However, FGSM attacks rely on only one single gradient step starting from a point on the data manifold, which as shown in the paper becomes uninformative for BNNs with good posterior approximation. If instead multi-step attacks initialised outside the manifold (e.g. randomly in a neighbourhood of the attacked test point) or even gradient free attacks were used, this observation might not hold. So while their theoretical and experimental findings could speak for the hypothesis of higher robustness through better posterior approximations, they are by no means conclusive and leave room for alternative hypotheses.

Apart from these two indirect comparisons of robustness, there are a number of works that inspect the adversarial robustness of BNNs with approximations that go beyond MAP against adversarial attacks without any comparison. Cardelli et al. (2019a) derive statistical guarantees on the existence of adversarial examples by verifying their existence on a number of samples from the BNN. Wicker et al. (2020) establish a method to compute the probability of drawing a sample from a posterior $p(\theta|\mathcal{D}_n)$ of a MFVI approximated BNN for which an adversarial example exists near a test point. Yet, in neither of these two works, the robustness of the resulting BNN has been compared to the robustness of MAP approximated networks.

4.2.2 Randomised Smoothing Techniques

Randomised smoothing has been proposed as a technique to increase adversarial robustness (Cohen et al., 2019; Dvijotham et al., 2019). In a series of experiments it leads to improved adversarial robustness, and most importantly this robustness can also be certified. The idea behind randomised smoothing is that while adversarial examples can be found, they first have to be looked for, i.e. just a random (small) perturbation will typically not result in an adversarial example. Therefore, basing a classification decision on a majority vote of classifications of points randomly sampled in the neighbourhood of the point to be classified should still result in correct predictions for unperturbed (clean) inputs, but hopefully also avoid misclassification of an adversarially perturbed point as their neighbourhood will still predominantly be classified like the class of the unperturbed point. The link to a posterior approximation of a BNN becomes clear when investigating the effect of randomised smoothing a bit closer. Writing the random perturbation as $\nu \sim p_\nu$ drawn from some distribution p_ν, we get

$$\begin{aligned}
\hat{f}^{(n)}(x) &= \mathbb{E}_\nu[f^\theta(x+\nu)] \\
&= \mathbb{E}_\nu[f^{\theta_K}(f^{\theta_{K-1}}(\dots\phi^1(W^1(x+\nu)+b^1)))] \\
&= \mathbb{E}_\nu[f^{\theta_K}(f^{\theta_{K-1}}(\dots\phi^1(W^1x+(W^1\nu+b^1))))] \\
&= \mathbb{E}_{\tilde{\nu}}[f^{\theta_K}(f^{\theta_{K-1}}(\dots\phi^1(W^1x+\tilde{\nu})))] \\
&= \mathbb{E}_{\tilde{\nu}}[f^{\tilde{\nu}}(x)]
\end{aligned} \tag{4.1}$$

with $\tilde{\nu} = (W^1\nu + b^1)$. Thus, the prediction function corresponds to the mean prediction function of a BNN where all parameters except for the bias in the first layer are fixed and for which the inferred posterior over that bias follows the distribution of $\tilde{\nu}$, which as expressed in the last equation can be viewed of exact Bayesian posterior inference for a parametric model $f^{\tilde{\nu}}$ (not exactly a neural network anymore). In this restricted case, exact Bayesian posterior inference has been found to be more robust than an MAP approximation (which corresponds to using a Dirac delta function centered on b^1).

4.2.3 Ensemble Methods

Lastly, some works have successfully increased the adversarial robustness of a given DNN model by forming an ensemble of a number of models of that same type which have each been trained slightly differently. The result can be seen as an MC estimate of the mean prediction function of a BNN, with a posterior distribution over θ that puts large probability mass on the configurations that yield the involved ensemble models. However, the inference in these ensemble methods is not done in a Bayesian way and it is unclear how such a posterior distribution could be conceived following the Bayesian formalism. As a notable example, Wang et al. (2019) yield the strongest adversarial robustness among models involving uncertainty in their predictions in a benchmark study (Croce and Hein, 2020) by using an ensemble of 5 ResNet NN models (He et al., 2016) which each have a Gaussian noise injection in the skipping connections. While this combines the robustness inducing randomised smoothing with an ensemble technique, other works indicate that already an ensemble consisting of 5 differently initialised DNNs can increase adversarial robustness against any single model involved by a large margin (Abbasi et al., 2020).

In summary, we've found that existing work that makes (incomplete) comparisons of the adversarial robustness of well approximated BNNs vs. MAP approximated BNNs remains inconclusive, but if anything slightly favours the hypothesis that the mean prediction function of a well approximated BNN exhibits higher adversarial robustness than the corresponding MAP approximated prediction function. Research on related models also favours this hypothesis. In the next section we formalise the problem definition and research question that we pose to investigate this hypothesis further.

4.3 Problem Definition, Research Questions, and Outline

Since we are not just interested in robustness and safety on a local scale, but in a more general and global sense, we use the following definition to compare adversarial robustness among BNNs.

Definition 8 (Adversarial Global Accuracy)**.** *Consider a data generating process $\mathcal{G}$ described by a probability measure $\mathbb{P}_{\mathcal{X},\mathcal{Y}} : \mathcal{F}_{\mathcal{X}} \times \mathcal{F}_{\mathcal{Y}} \to [0,1]$ with $\mathcal{Y} = \{1, 2, \ldots, C\}$. For a data set $\mathcal{D}_n$ sampled from $\mathcal{G}$ let $\hat{f}^{(n)} : \mathcal{X} \times \mathcal{Y} \to [0,1]$ be the prediction function of a model trained on $\mathcal{D}_n$ which maps inputs $x \in \mathcal{X}$ to the probability $\hat{f}^{(n)}(x)_c$ it assigns to x belonging to class c. Let $\gamma > 0$ and $\|\cdot\|_p$ be an L_p-norm induced metric on $\mathcal{X}$. Using the definition of Local Maximal Safety Radius (Definition 5), we define the Vulnerable Set of $\hat{f}^{(n)}$ as*

$$\begin{aligned} VS_p(\hat{f}^{(n)}, \gamma) &= \{(x,y) | LMSR_p(x,y) < \gamma\} \\ &= \{(x,y) | \exists \delta \in \mathcal{X} \text{ with } \|\delta\|_p \leq \gamma \text{ s. t. } \arg\max_{c \in \mathcal{Y}} \hat{f}^{(n)}(x)(x+\delta)_c \neq y\} \end{aligned} \tag{4.2}$$

Then the adversarial global accuracy (AGA) of $\hat{f}^{(n)}$ w.r.t. $\mathcal{G}$ at perturbation budget γ is defined as

$$AGA_p^{\mathcal{G}}(\hat{f}^{(n)}, \gamma) = 1 - \mathbb{P}_{\mathcal{X},\mathcal{Y}}\left(VS_p(\hat{f}^{(n)}, \gamma)\right) \tag{4.3}$$

This definition of Adversarial Global Accuracy (AGA) is a way to formalise a global measure of robustness based on the Local Maximal Safety Radius in Definition 5, which is similar to e.g. Pinot et al. (2019); Diochnos et al. (2018).
We only use the mean posterior prediction to analyse AGA, i.e. $\hat{f}^{(n)}(x)_c = \int f^{\theta}(x)_c p(\theta) d\theta$ (see Eq. (2.22)) and restrain from including further properties of the posterior predictive distribution, like for example predictive uncertainty (Smith and Gal, 2018) into an additional detection mechanism for adversarial attacks (Feinman et al., 2017) for two reasons. Firstly, it has been demonstrated that including such artifacts into the attack objective can circumvent these mechanisms (Grosse et al., 2018). Second, a large body of previous work on the robustness of Bayesian neural networks purely focusses on the mean and thus following their approach allows

for comparison and joint analysis of results (Yuan et al., 2020; Liu et al., 2018a; Zimmermann, 2019; Carbone et al., 2020).

Furthermore, like in the previous chapter, we focus on the $L_\infty-$norm induced metric which is commonly used in the literature (Croce and Hein, 2020) and has been described as most aligned with human perception (Warde-Farley and Goodfellow, 2016). The first research question we aim to answer in this chapter is thus:

Research Question No1 *Given a fixed NN architecture $f^\theta(\cdot)$ and a fixed training data set $\mathcal{D}_n$, does the Bayesian mean prediction function of the NN exhibit higher Adversarial Global Accuracy w.r.t. $L_\infty-$norm (Def. (8)) if it is approximated using HMC rather than MAP?*

To answer this question, we will conduct a series of experiments in this chapter involving a variation of different data sets and NN architectures $f^\theta(\cdot)$ for which good posterior approximations using HMC are possible. Namely, we will work with both the MNIST and FashionMNIST data set (LeCun, 1998; Xiao et al., 2017). We use both a zero-hidden layer NN (equivalent to multinomial logistic regression) and a three-hidden layer NN. As predictions and robustness of the mean posterior prediction function could potentially change significantly with changes in the prior (MacKay, 1992; Wilson and Izmailov, 2020), we need to inspect a variety of prior distributions in order to draw conclusions on the first research question. To do this, we use prior distributions of the form $p(\theta) = \mathcal{N}(\theta|0, \alpha^2 I)$ and vary the prior variance α^2.

Incidentally, this also allows us to simultaneously inspect the following second research question:

Research Question No2 *Given a fixed NN architecture $f^\theta(\cdot)$, and a fixed training data set $\mathcal{D}_n$, how big is the effect of the prior weight distribution on the Adversarial Global Accuracy w.r.t. $L_\infty-$norm (Def. (8)) of a BNN?*

The rest of this chapter is structured as follows: In the next section we describe the experimental set up in more detail. This is followed by a section in which we present the results for the multinomial logistic regression model and another section in which the results for the three-hidden layer NN are presented. Finally we conclude with a discussion of the implications of our results and an outlook.

4.4 Experimental Set Up

4.4.1 Data

The two data sets we use are MNIST (LeCun, 1998) and FashionMNIST (Xiao et al., 2017)[1]. Both sets constitute of $60,000$ training and $10,000$ test images that each belong to one of a total of 10 classes. In the case of MNIST, these classes are digits $(0-9)$, in the case of FashionMNIST they are fashion items (e.g. 'T-shirt/top', 'Trouser', 'Bag', etc.). The images consists of 28×28 pixels in each case (i.e. amount to a total input dimensionality of 784) and are integer valued between 0 and 255, representing their intensity on a gray scale. Figure 4.1 shows samples of both data sets. Before running experiments we divide inputs by 255, rescaling input values

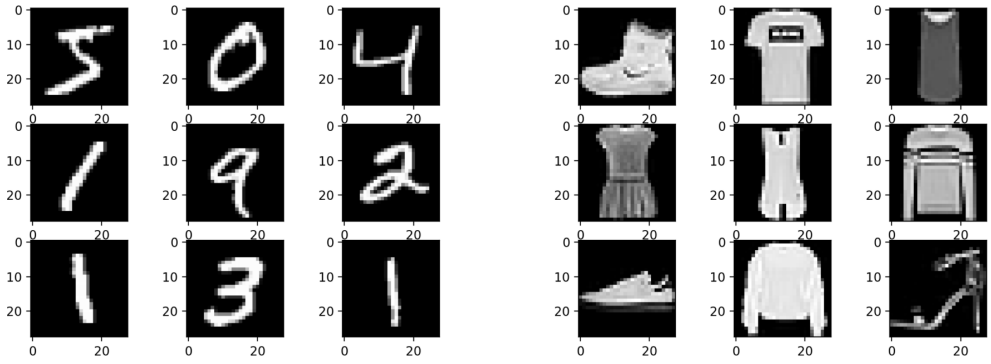


Figure 4.1: Samples from MNIST (left) and FashionMNIST (right) data set.

to $[0, 1]$ for both data sets. Since we found that good training and test accuracies

[1]Our plan was to also conduct experiments on CIFAR-10 to have a broader and more diverse set of experiments. However, we learnt from personal communication with the authors of the Hamiltorch package (Cobb et al., 2019), which we use for the HMC based inference, that they had tried to achieve $> 50\%$ test set accuracies for CIFAR-10 in vain for almost a year, so we shied away from assessing robustness on a data set on which we would not even get good clean test set accuracies

can be achieved with fewer data points, we randomly subsample $5,000$ training and $1,000$ test points for our experiments to speed up computations.

4.4.2 Models

We use two different types of fully connected feedforward NN architectures: a simple (linear) multinomial logistic regression model (Mlogreg, corresponding to a zero-hidden layer NN with softmax activation function) and a three-hidden layer NN which we refer to as DNN3 [2].

In the Mlogreg model, depicted in Figure 4.2, input values are directly mapped to the logits of the outputs in a linear way. Since the softmax function that maps the logits to predicted class probabilities $\hat{f}_c^{(n)}$ preserves rank, this means that Mlogreg is a linear classifier. In the DNN3 model, depicted in Figure 4.3, input values

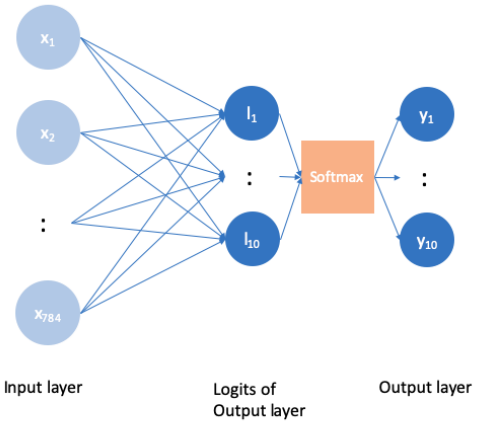


Figure 4.2: The linear Mlogreg model used in our experiments.

are mapped to the logits of the outputs via linear combinations in three hidden layers with 32, 64, and 32 hidden units respecitvely. Sigmoid activation functions of the form $s(z) = 1/(1 + \exp(-z))$ introduce non-linearity and allow for a larger expressive power of the model. The prior distributions over the model parameters are for both models of the form form $p(\theta) = \mathcal{N}(\theta|0, \alpha^2 I)$. This fairly vanilla form

[2]Note that while more state-of-the-art network architectures, like CNNs or ResNets, would have been interesting to analyse as well, the libraries (Hamiltorch, Cobb et al. (2019)) and hardware (single GPUs on university compute clusters) we use are not suitable for these more advanced architectures w.r.t. the HMC sampling required

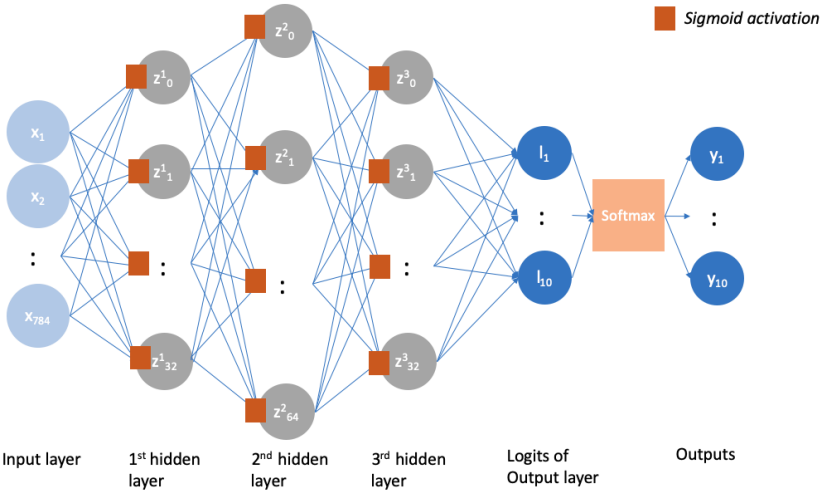


Figure 4.3: The 3 hidden layer neural network DNN3 model used in our experiments.

of prior has been used extensively in the literature and can be considered the standard parameter prior when it comes to BNNs (MacKay, 1992; Blundell et al., 2015; Wilson and Izmailov, 2020). While too small values of α^2 correspond to a too strong weight decay term for MAP training to converge, too large values of α^2 made it difficult to obtain convergence of the HMC sampling algorithm when inferring $p(\theta|\mathcal{D}_n)$. The values of the prior variance α^2 we use across data sets and models are thus $\{0.1, 1, 10, 100\}$.

4.4.3 Inference

Given a training data set $\mathcal{D}_n$ and a model consisting of NN architecture $f^\theta(\cdot)$ and a parameter prior $p(\theta) = \mathcal{N}(\theta|0, \alpha^2 I)$, we do two different types of approximate inference: MAP and HMC.

To obtain the MAP prediction function

$$\hat{f}^{(n),MAP}(x)_c = f^{\hat{\theta}}(x)_c \tag{4.4}$$

with $\hat{\theta} = \arg\max_\theta p(\theta|\mathcal{D}_n)$, we maximise

$$\log p(\theta|\mathcal{D}_n) \propto \sum_{i=1}^{n} \log f^\theta(x^{(i)})_{y^{(i)}} - \frac{1}{2\alpha^2}\|\theta\|_2^2 \tag{4.5}$$

which is equivalent to minimising

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} -\log f^\theta(x^{(i)})_{y^{(i)}} + \frac{1}{2\alpha^2}\|\theta\|_2^2 \tag{4.6}$$

Note that for $\alpha^2 \to \infty$, this becomes equivalent to finding the MLE prediction function. This is done using the Adam optimiser (Kingma and Ba, 2014) as implemented in PyTorch (Paszke et al., 2019) for 100 epochs in batches of size 32. We use an initial learning rate of 0.01 which is decayed by a factor of 0.2 after $50, 60, 70$, and 80 epochs. In order to ensure convergence also for low values of α^2, we use a wider initialisation of θ than the built-in one, namely we initialise with random draws from a $\mathcal{N}(0, 10I)$ prior. These hyperparameter settings have been selected to obtain high training accuracies across all priors and experiments.

For the HMC prediction function we obtain samples from the posterior $p(\theta|\mathcal{D}_n)$ using the Hamiltorch package and its default implicit integrator. (Cobb et al., 2019). Depending on data set and model we adjust the hyperparameters in order to achieve good mixing and convergence towards the true posterior. Our definition of good mixing is slightly ad-hoc and is a combination of three elements: 1) acceptance rates between ≈ 0.3 and ≈ 0.95, 2) reasonable visual appearance of a random set of four (sampled) posterior marginal distributions of model parameters (e.g. not Delta dirac shaped), and 3) train and test set accuracies of the resulting HMC prediction function that is roughly in line with that of the MAP prediction function. The selected hyperparameters are shown in Table 4.1. For the Mlogreg models we

	MNIST		FashionMNIST	
Hyperparameter	Mlogreg	DNN3	Mlogreg	DNN3
Step size	0.015	0.003	0.007	0.002
Trajectory length	30	20	50	30
Mass	1.0	0.5	0.8	0.3

Table 4.1: Hyperparameter settings for HMC algorithms per model and data set. Mass is the Mass matrix diagonal scale.

sample until we have $1,000$ samples, while for the DNN3 models we sample for $10,000$ samples. We then randomly select 100 samples $\{\theta^{(j)}\}_{1 \leq j \leq 100}$ [3] from the latter

[3] Unfortunately, a significantly higher number of samples is not feasible on the hardware we use (single GPUs), as we run out of memory (at least for DNN3)

50%[4] of the samples collected in total as the samples from the posterior $p(\theta|\mathcal{D}_n)$ that we use to approximate the HMC mean prediction function as a MC estimate

$$\hat{f}^{(n),HMC}(x)_c \approx \frac{1}{100}\sum_{j=1}^{100} f^{\theta^{(j)}}(x)_c \tag{4.7}$$

4.4.4 AGA Estimation

In light of finite data, the adversarial global accuracy in Definition 8 can only be estimated from the data at hand. To this end we use the test data $\mathcal{T}_m$ and estimate the AGA w.r.t. the $L_\infty-$norm at perturbation level γ as the share of test points for which adversarial local safety can be established, i.e. for which $\gamma > LMSR$:

$$AGA_\infty^{\mathcal{T}_m}(\hat{f}^{(n)}, \gamma) = 1 - \frac{|\mathcal{S}_{\hat{f}^{(n)}}(\mathcal{T}_m)|}{m} \tag{4.8}$$

with

$$\mathcal{S}_{\hat{f}^{(n)}}(\mathcal{T}_m) = \{(x^{(i)}, y^{(i)}) \in \mathcal{T}_m | \exists \delta \in \mathcal{X} \text{ with } \|\delta\|_\infty \leq \gamma \text{ s. t. } \arg\max_{c\in\mathcal{Y}} \hat{f}^{(n)}(x^{(i)}+\delta)_c \neq y^{(i)}\} \tag{4.9}$$

For any single test point $(x^{(i)}, y^{(i)})$, determining the existence or absence of a δ fulfilling the conditions in the RHS of Eq. (4.9) is a challenging problem and in general NP-hard (w.r.t. number of input dimensions or hidden units (Ruan et al., 2018)). Combined with the fact that for the HMC mean prediction function in Eq. (4.7) we would need to conduct it 100 times, we resort to analysing an upper bound to $AGA_\infty^{\mathcal{T}_m}(\hat{f}^{(n)}, \gamma)$ instead. This is done by attacking the model predictive functions with the following adversarial attack from Croce and Hein (2020). It consists of two runs of an untargeted PGD-20 attack (Definition 4 and Madry et al. (2017)) starting from a random (uniform) point inside the $\gamma-$ball around x. The first run is using the standard cross entropy (CE) loss, while for the second run the scaling invariant difference in logits ratio (DLR) loss is used (see Section 2.2.2).

The DLR loss is both shift and rescaling invariant, unlike the standard CE loss, which is only shift invariant. In particular this is important as a lack of scaling

[4]We discard the first 50% of samples for burn-in. In our experiments, this ensured sufficiently good mixing as defined above

invariance can lead to gradient masking and a subsequent failure mode of gradient based attacks such as PGD if all logits are scaled by a large constant $\beta \gg 0$ (Croce and Hein, 2020). When attacking $\hat{f}^{(n),HMC}$ we approximate its gradient w.r.t. input again using a MC estimate based on the same 100 samples $\{\theta^{(j)}\}_{1\leq j\leq 100}$ from the posterior $p(\theta|\mathcal{D}_n)$. While in a practical attack scenario, the attacking algorithm might face new samples from $p(\theta|\mathcal{D}_n)$ forming the MC estimate $\hat{f}^{(n),HMC}$ in Eq. (4.7) after each PGD iteration, we are only interested in the question whether the original HMC mean prediction function that is used at the test point is susceptible to an adversarial attack and hence reuse the same 100 samples for the MC estimate of its gradient at each PGD iteration.

Finally, for the MAP predictive function of Mlogreg, we also employ interval bound propagation (IBP, Gowal et al. (2018)) to get a lower bound to $AGA_{\infty}^{\mathcal{T}_m}(\hat{f}^{(n)}, \gamma)$ (by establishing adversarial local safety in an overly conservative way). Unfortunately, meaningful bounds are only obtained for this shallow model, already for DNN3, IBP only returns vacuous bounds, that is a lower bound of 0, which is trivial for probabilities [5]. We will designate the upper bound on $AGA_{\infty}^{\mathcal{T}_m}(\hat{f}^{(n)}, \gamma)$ conceived by the attack described above as $\overline{AGA}(\gamma)$ and its lower bound conceived with IBP as $\underline{\text{AGA}}(\gamma)$, omitting all further specifications from the notation for brevity. In order to get a more holistic picture, we calculate all metrics for a range of perturbation budgets $\gamma \in \{0.1, 0.2, 0.3, 0.4\}$. Also, all experiments are repeated for five different random seeds ($\{10, 20, 30, 40, 50\}$).

[5] Recently, Wicker et al. (2021) have developed methods that extend IBP based guarantees also to HMC approximated BNNs. We only became aware of these results after the original submission of this thesis, but do not think that using them to compute $\underline{\text{AGA}}(\gamma)$ also for $\hat{f}^{(n),HMC}$ would change our findings. This is because in the Mlogreg case we see in Section 4.5.1 that $\overline{AGA}^{HMC} < \underline{\text{AGA}}^{MAP}$, so since $\underline{\text{AGA}}^{HMC} < \overline{AGA}^{HMC}$ by definition, we would simply see $\underline{\text{AGA}}^{HMC} < \underline{\text{AGA}}^{MAP}$ as well. For DNN3 on the other hand, the IBP bounds are already vacuous for $\hat{f}^{(n),MAP}$, so we would expect the same to be true for $\hat{f}^{(n),HMC}$ unfortunately

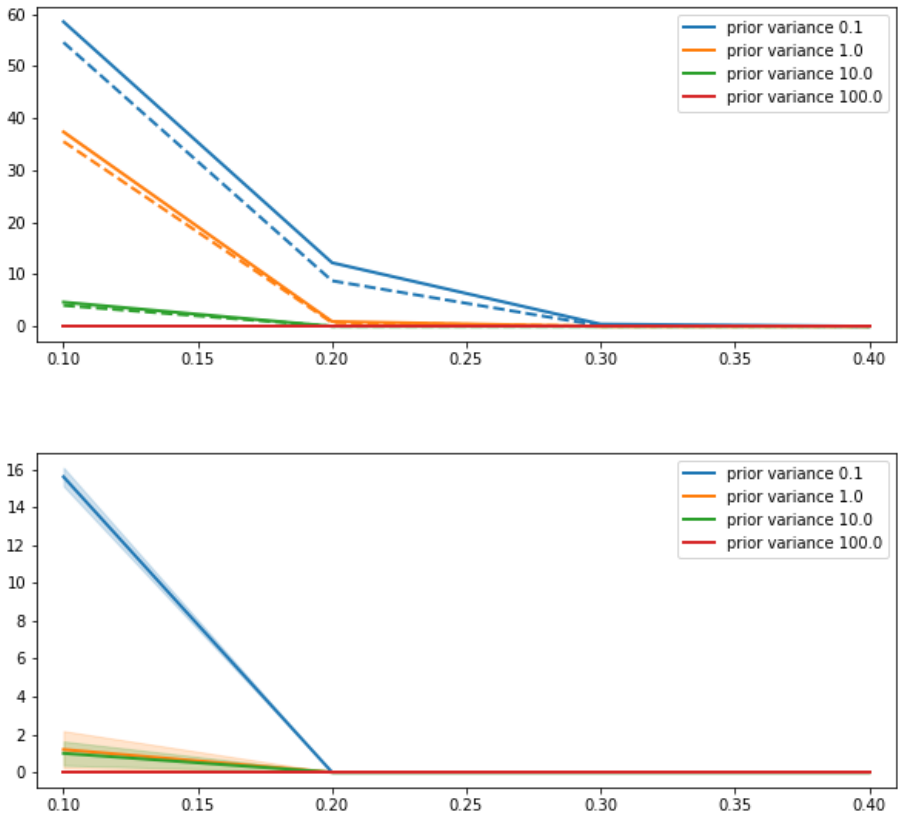


Figure 4.4: Values of $\overline{AGA}(\gamma)$ for Mlogreg model on MNIST data set with 5,000 train and 1,000 test images. Solid line shows average over 5 random seeds, shaded area depicts one standard deviation. **Top**: Results for MAP predictions. Dotted lines show values of $\underline{AGA}(\gamma)$. **Bottom**: Results for HMC mean predictions.

4.5 Experimental Results

4.5.1 Multinomial Logistic Regression / Neural Network with Zero Hidden Layers

The comparisons of AGA on MNIST can be found in Figure 4.4, while the ones on FashionMNIST are depicted in Figure 4.5. Further experiment outcomes such as training and test accuracy are summarised for both data sets in Table 4.2.

As we can see, the adversarial robustness of the HMC mean predictor is lower than the one of the MAP predictor for this simple multinomial logistic regression model. Its $\overline{AGA}$ score is lower for all prior variances, all values of γ and on both data sets, as long as it is not 0 for both inference types (e.g. for prior variance 100).

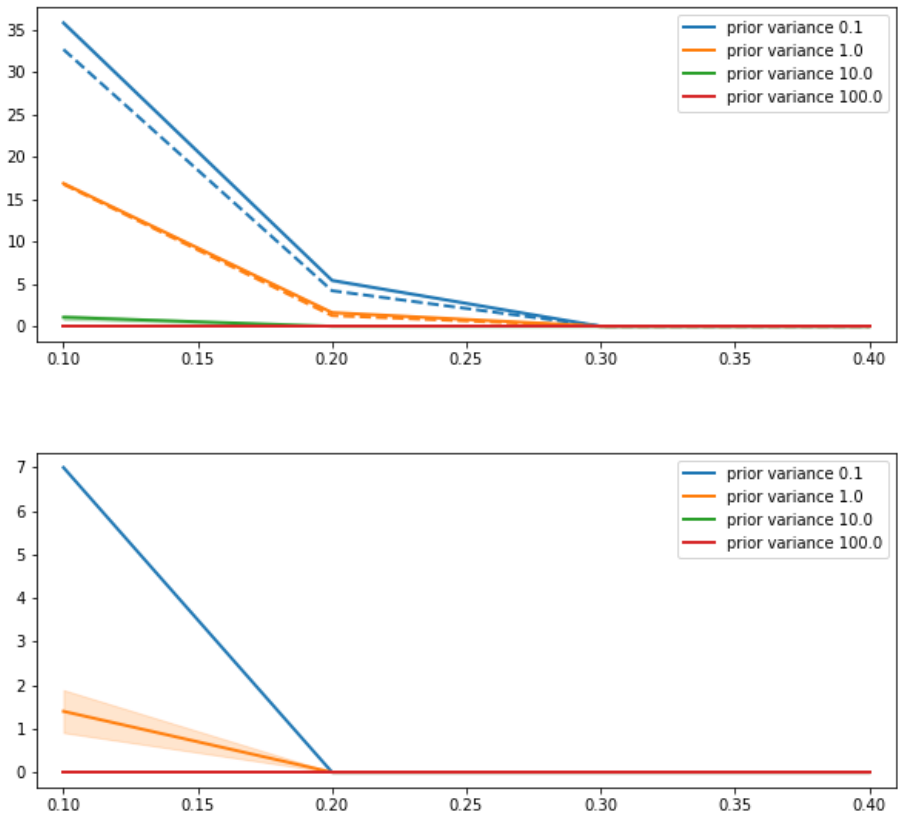


Figure 4.5: Values of $\overline{AGA}(\gamma)$ for Mlogreg model on FashionMNIST data set with 5,000 train and 1,000 test images. Solid line shows average over 5 random seeds, shaded area depicts one standard deviation. **Top**: Results for MAP predictions. Dotted lines show values of $\underline{AGA}(\gamma)$. **Bottom**: Results for HMC mean predictions.

Furthermore, since the lower bound on the AGA estimate that we conceive for the MAP predictor using IBP ($\underline{AGA}$), is consistently higher than the upper bound $\overline{AGA}$ of the HMC mean predictor, we can for the Mlogreg model indeed conclude that $AGA^{\mathcal{T}_m}_{\infty}(\hat{f}^{(n),MAP}, \cdot) \geq AGA^{\mathcal{T}_m}_{\infty}(\hat{f}^{(n),HMC}, \cdot)$, for both $(\mathcal{D}_n, \mathcal{T}_m) =$ MNIST and $(\mathcal{D}_n, \mathcal{T}_m) =$ FashionMNIST.

Lastly, with respect to the second research question, we can establish that also for the HMC mean predictor, stronger regularisation through more restrictive prior distributions does not only help (clean) test set generalisation / prevent overfitting, but also increases adversarial test set accuracy. In Table 4.2, we do see a big gap of $> 10\%$ between train and clean test accuracy for a prior variance of 100, which is reduced to $< 5\%$ for a prior variance of 0.1, reflecting the reduction in overfitting

	prior	MNIST		FashionMNIST	
Metric	variance	MAP	HMC	MAP	HMC
Train acc.	0.1	87.5 (<0.1)	94.2 (0.1)	80.1 (0.1)	89.2 (<0.1)
	1	92.5 (<0.1)	97.7 (<0.1)	86.9 (0.1)	94.1 (0.1)
	10	96.7 (0.1)	99.7 (<0.1)	91.6 (0.1)	97.9 (0.1)
	100	98.4 (0.1)	100.0 (0.0)	90.0 (0.3)	99.5 (<0.1)
Test acc.	0.1	87.7 (0.1)	90.7 (0.2)	78.3 (0.2)	83.2 (0.1)
	1	90.2 (0.0)	90.0 (0.2)	82.3 (0.1)	82.4 (0.3)
	10	90.2 (0.1)	88.7 (0.5)	81.7 (0.6)	80.1 (0.7)
	100	85.6 (1.0)	86.8 (0.6)	77.2 (0.5)	77.8 (0.3)
$\text{Avg}_i\ \mathbb{V}_j[\theta_i^{(j)}]$	0.1	-	0.09 (0.0)	-	0.1 (0.0)
	1	-	0.8 (0.0)	-	0.8 (0.0)
	10	-	5.5 (0.1)	-	4.6 (0.1)
	100	-	12.4 (0.4)	-	9.0 (0.3)

Table 4.2: Further key metrics of predictive functions of Mlogreg model on MNIST and FashionMNIST data sets. Standard deviations over 5 random seeds shown in brackets.

achieved through the more restrictive prior [6]. This reduction in generalisation gap can even more starkly be observed w.r.t. the adversarial test accuracy in Figures 4.4 and 4.5, indicating that a smaller prior variance increases adversarial robustness for the priors investigated. However, the effect is not as strong for the HMC mean predictor as for the MAP predictor, as the total range of $\overline{AGA}$ values for the MAP predictor is about $4\times$ higher at $\gamma = 0.1$ on both data sets (roughly 60% vs. 15% on MNIST and 30% vs. 7.5% on FMNIST).

4.5.2 Neural Network with Three Hidden Layers

The comparisons of AGA on MNIST can be found in Figure 4.6, while the ones on FashionMNIST are depicted in Figure 4.7. Further experiment outcomes such as training and test accuracy are summarised for both data sets in Table 4.3. Generally, we observe the same phenomenon as for the linear Mlogreg model: the adversarial robustness of the MAP prediction function seems higher than the one of the HMC

[6] Remember also that as $\alpha^2 \to \infty$, MAP actually converges towards MLE (see Section 4.4.3), which is known to be prone to overfitting

mean prediction function. In immediate comparison of identical prior variance α^2, perturbation γ and data set, $\overline{AGA}^{MAP}$ is higher than $\overline{AGA}^{HMC}$. However, the picture is not as clear for the deeper model. To begin with, the MAP training did not converge for a prior variance of 0.1 for DNN3 as training collapsed to a constant predictor (the L_2-penalisation becomes so strong that the loss function in Eq. (4.6) is minimised for 0-valued parameters θ, corresponding to a constant predictor). Furthermore, we did not succeed to obtain a samples from a satisfactory approximation of $p(\theta|\mathcal{D}_n)$ using HMC for a prior variance of 100 [7]. As a result, we can only directly compare the values for prior variances of 1 and 10. To make things even worse, the training and test accuracies for the MAP prediction function are not satisfactory with a prior variance of 1, since the increased amount of parameters makes an L_2-regularisation parameter of $\frac{1}{2}$ resulting from such a prior variance too strong to allow for a better solution for DNN3 (we tried increasing number of epochs and varying hyperparameter settings to no avail). Thus the only strong indicator for the difference in robustness are the results for a prior variance of 10, which is however only observed for MNIST this time while for FashionMNIST the robustness seems to be the same.

At least, with respect to the second research question, we can again confirm the results from Mlogreg also for the DNN3 model: higher prior variance can lead to a significant decrease in adversarial robustness for the HMC mean prediction function. Also, the effect the prior can have on the posterior robustness for the HMC mean prediction function is in a similar range (roughly 15% for MNIST and 7.5% for FMNIST at $\gamma = 0.1$) for the DNN3 model as for the Mlogreg model, while for the MAP prediction function it is much lower for the DNN3 model in comparison to what was observed for the Mlogreg model in Section 4.5.1.

[7] Despite conducting hyperparameter grid search over stepsize $\times$ trajectory length $\times$ Mass $= \{0.001, 0.002, 0.003, 0.004, 0.005\} \times \{10, 20, 30, 40\} \times \{0.1, 0.3, 0.5, 1.0\}$ (not combining small Mass with large stepsize and trajectory length nor large stepsize with large trajectory length though, as this lead to worse results quickly in our experience), the average variance of obtained posterior samples for a prior variance of 100 was always significantly lower than when a prior variance of 10 was used, as were train and test accuracies

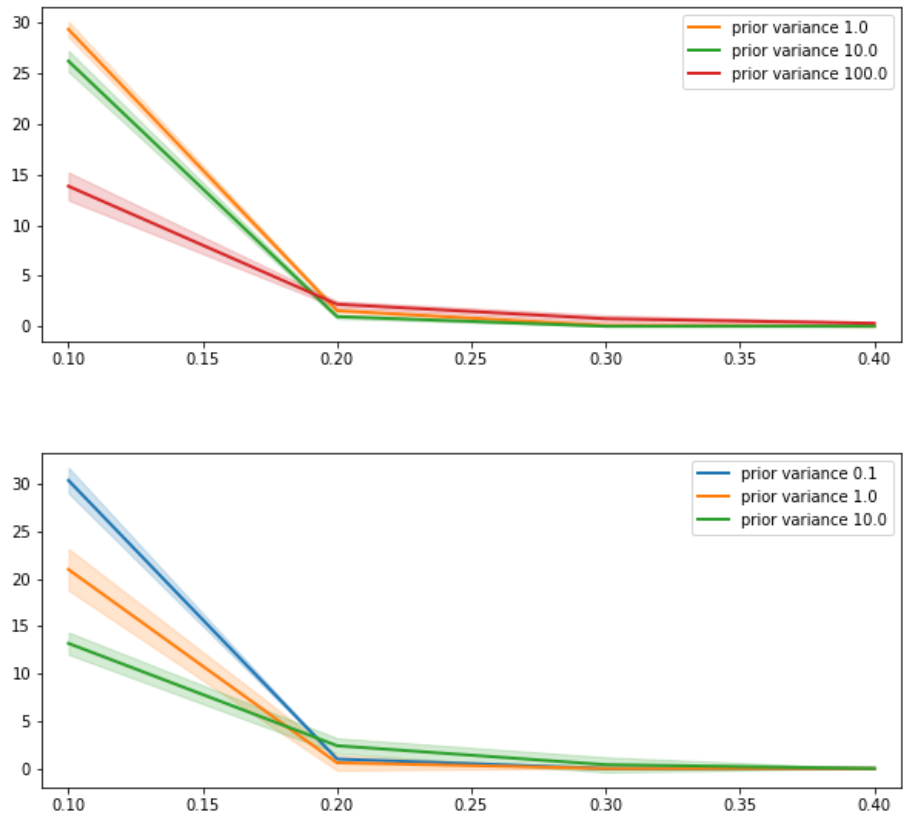


Figure 4.6: Values of $\overline{AGA}(\gamma)$ for DNN3 model on MNIST data set with 5,000 train and 1,000 test images. Solid line shows average over 5 random seeds, shaded area depicts one standard deviation. **Top**: Results for MAP predictions. **Bottom**: Results for HMC mean predictions.

4.6 Discussion

We have empirically investigated whether being more Bayesian helps in increasing the adversarial robustness of a neural network model by comparing the adversarial robustness of the mean prediction function of neural networks with HMC approximated posteriors to identical networks with MAP approximated posteriors on two different data sets, for two different models, and for four different prior distributions. Results for GPCs in Chapter 3 as well as existing literature favour this hypothesis, yet a principled analysis has been outstanding.

Our findings interestingly indicate that in fact the opposite is true. Empirically, facing state-of-the-art adversarial attacks, the HMC mean prediction function was

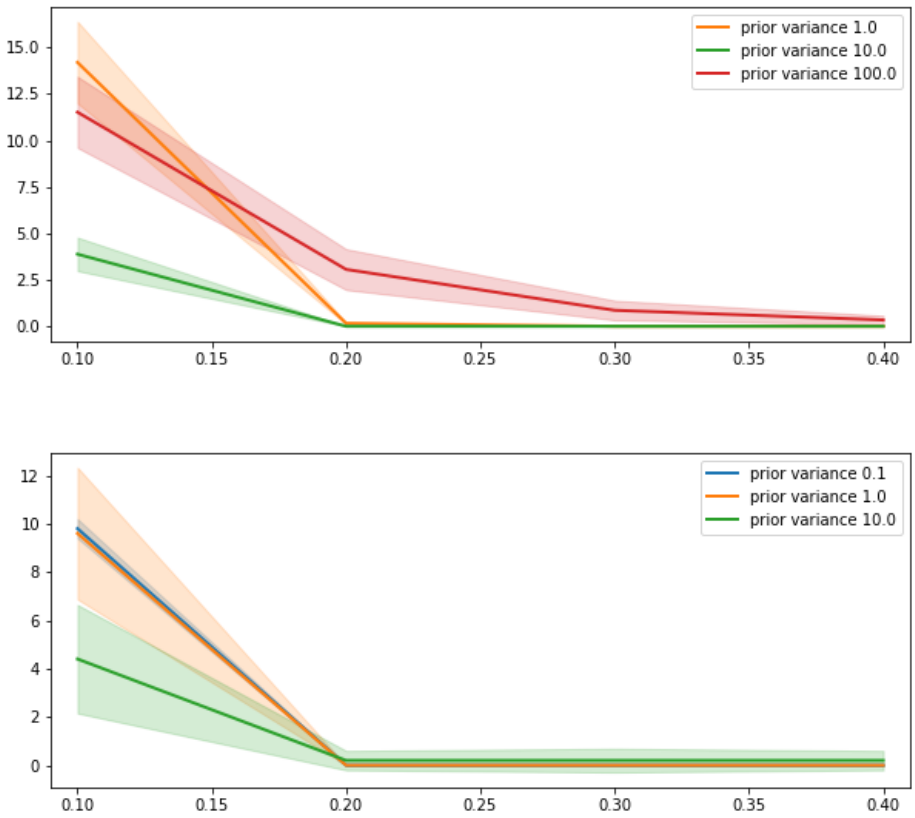


Figure 4.7: Values of $\overline{AGA}(\gamma)$ for DNN3 model on FashionMNIST data set with 5,000 train and 1,000 test images. Solid line shows average over 5 random seeds, shaded area depicts one standard deviation. **Top**: Results for MAP predictions. **Bottom**: Results for HMC mean predictions.

found to be less robust than the MAP prediction function. This was especially true for the linear Bayesian model under inspection (0-hidden layer NN), where the difference in robustness was most explicit and additionally confirmed by interval bound propagation verification methods. However, it was also true for the DNN (whenever direct comparison was possible). The seeming contradiction to Carbone et al. (2020) can be explained by the fact that they use one-step FGSM attacks starting from test points directly rather than PGD-20 attacks starting from uniform unit-ball perturbations, which, especially in light of their theoretical findings about 0 value gradients at data points, makes the attack they used particularly ineffective for well-approximated BNNs. The difference in findings compared to what we observed for GPCs in Chapter 3 (where better posterior approximation led to better

	prior	MNIST		FashionMNIST	
Metric	variance	MAP	HMC	MAP	HMC
Train acc.	0.1	-	94.4 (<0.1)	-	87.0 (0.2)
	1	85.3 (0.3)	99.5 (0.1)	74.4 (2.1)	97.0 (0.2)
	10	99.5 (0.1)	100.0 (<0.1)	92.7 (0.2)	99.8 (0.1)
	100	99.8 (0.1)	-	92.0 (0.3)	-
Test acc.	0.1	-	92.4 (0.2)	-	82.3 (0.5)
	1	85.3 (0.6)	95.0 (0.3)	72.6 (1.9)	85.0 (0.6)
	10	95.2 (0.3)	93.0 (0.6)	83.3 (0.7)	83.2 (0.3)
	100	90.3 (1.0)	-	81.6 (0.6)	-
$\text{Avg}_i\ \mathbb{V}_j[\theta_i^{(j)}]$	0.1	-	0.1 (0.0)	-	0.1 (0.0)
	1	-	0.9 (0.0)	-	0.9 (0.0)
	10	-	3.8 (<0.1)	-	5.0 (0.1)

Table 4.3: Further key metrics of predictive functions of DNN3 model on MNIST and FashionMNIST data sets. Standard deviations over 5 random seeds shown in brackets. For a prior variance of 0.1, we did not succeed to obtain a MAP solution for DNN3, as training collapsed to a constant predictor. For a prior variance of 100, we did not succeed to obtain a samples from a satisfactory approximation of $p(\theta|\mathcal{D}_n)$ using HMC.

adversarial robustness) can be explained by two things. Firstly, the connection between GPs and BNNs only holds in the limit case of infinite-width layers, which in practice can never be achieved with BNNs and in particular did not apply for the small scale BNNs we analysed in this chapter. Secondly, while in this chapter we compared two extreme ends of the quality of posterior approximations (MAP vs. HMC), in Chapter 3 we compared two posterior approximations which lie in between those two on the spectrum of the quality of posterior approximations. It could thus be that for GPCs and / or BNNs, a better posterior approximation only leads to better adversarial robustness until a certain point, beyond which it actually degrades adversarial robustness.

Also, previous results from ensemble methods or randomised smoothing methods indicating better robustness for better posterior approximations ultimately come from distributions that are much stronger concentrated around the MAP solution than the ones obtained by full HMC based Bayesian inference (e.g. randomised

smoothing corresponds to a full distribution only over the intercept in the first layer and MAP on all other parameters, and ensemble methods use only samples with very high posterior probability that correspond to local maxima of MAP optimisation). We thus conjecture that among the adversarial robustness of all possible parameter configurations, MAP solutions perform reasonable well, while a significant share of configurations in the typical set of the posterior distribution might exhibit shocking adversarial robustness. Thus, for the high-dimensional problems that we deal with where MAP and typical set potentially deviate significantly (see Betancourt (2017) for a discussion of mode/MAP vs. typical set in high-dimensional problems), the resulting Monte Carlo estimate of the mean prediction coming from draws of the typical set might suffer from such bad adversarial robustness. We conclude that marginalising over well approximated posterior distributions whose probability mass is not concentrated around MAP solutions is potentially harmful for BNNs, and thus posterior approximations that are based on MAP or are close to the MAP approximation should be preferred if adversarial robustness is of concern (and both achieve comparable test / validation accuracies). However, more experiments on a wider set of NN architectures (including CNNs or RNNs), prior distributions, and data sets will also be needed to solidify this conjecture. In particular, it would be interesting to see if our results also hold for more complex data sets and NN architectures. Unfortunately, exact inference using HMC is to date not feasible for such problems but we hope that advances in the field might make them possible in the future. Lastly, memory limitations prevented us from conducting experiments with a higher number of samples for the Monte Carlo estimate of the mean prediction to inquire whether a more accurate mean estimation would improve robustness, but we hope that advances in hardware will make such an analysis equally possible in the future.

Remark 5 (Recent literature findings)**.** *After the original submission of this thesis, two important works appeared that corroborate our findings and hypotheses w.r.t. the adversarial robustness of HMC approximated BNNs in comparison to the robustness of their MAP approximated counterparts.*

In the first, Izmailov et al. (2021b) run extensive experiments using 'hundreds of Tensor processing unit (TPU) devices' to obtain HMC approximations for modern NN architectures on CIFAR-10 (ResNet-20-FRN, a 20 hidden-layer ResNet architecture (He et al., 2016) and CNN-LSTM, a CNN with long short-term memory as used in Wenzel et al. (2020)). One of their findings is that while their (clean) test accuracy is higher, HMC approximated BNN mean predictions respond worse to a number of covariate shifts [8] which they expose the test data to when compared to the corresponding MAP approximated predictions. The types of covariate shifts they inspect are systematic corruptions such as a change in brightness, an added Gaussian blur or a change in saturation [9]. In particular, they find that the stronger the corruption intensity, the relatively worse was the performance of the HMC mean predictor in comparison to the MAP predictor. Since adversarial attacks can be seen as a form of worst-case covariate shift (inside the perturbation budget), we see our results in line with these results by Izmailov et al. (2021b) and conjecture that given their results, we would see the same qualitative behaviour w.r.t. adversarial robustness also for CIFAR-10 and bigger and more complex architectures than DNN3 if we had the computational resources to conduct approximate inference using HMC for such problems.

Motivated by these first findings, Izmailov et al. (2021a) subsequently published a follow-up work in which they provide further experiments that test the robustness of HMC approximated BNNs to covariate shift, which again they test by adding corruptions similar to the ones used in Izmailov et al. (2021b) to test data. The datasets they use this time are both CIFAR-10 and MNIST, and they use a small CNN with 2 convolutional layers followed by 3 fully-connected layers. On MNIST they additionally use a 2 hidden-layer fully connected feedforward network with 256 neurons in each of the hidden layers, i.e. a DNN which is not too dissimilar from our DNN3. Again, they find that on average, the HMC mean prediction is less

[8] A covariate shift is defined as a change in the input generating distribution $p(x)$ between train and test data, i.e. $p_{train}(x) \neq p_{test}(x)$

[9] The full list is: brightness, contrast, defocus blur, elastic, fog, frost, frosted glass blur, Gaussian blur, Gaussian noise, shot noise, impulse noise, pixelate, saturate, spatter, speckle noise, zoom blur

robust to covariate shift than the MAP prediction.

Interestingly, they also give a theoretical explanation as to why this is the case. Namely, they show that the presence of linear dependencies in the training data (e.g. in the MNIST training data, the 'dead' pixels that remain black for all training points mean that the training data can be projected into an affine subspace, i.e. that there are such linear dependencies) results in a lack of posterior contraction in the first hidden layer if i.i.d. Gaussian priors on parameters are used, meaning that the resulting posterior mean prediction of the model will be affected by changes to those linear dependencies introduced by covariate shift. On the other hand, the L_2-penalisation term in the MAP approximation (see Eq. (4.6)) means that the corresponding parameter values will be shrunken to zero, which in turn makes the MAP predictions unaffected by such covariate shifts changing the linear dependencies observed in the training data. To avoid this negative impact of covariate shift to HMC approximated BNNs, Izmailov et al. (2021a) propose to replace the prior $p(W^1) = \mathcal{N}(W^1|0, \alpha^2 I)$ with the empirical covariance prior $p(W^1) = \mathcal{N}(W^1|0, \alpha^2\Sigma + \epsilon I)$ for some small constant $\epsilon > 0$ and $\Sigma := \frac{1}{n-1}\sum_{i=1}^{n} x^{(i)} {x^{(i)}}^T$ (i.e. the empirical covariance matrix), which helps to make the HMC mean predictor more robust to covariance shift. These findings give us hope that while the adversarial robustness of HMC approximated BNNs with Gaussian i.i.d. priors seems to be lower than the robustness of their MAP approximated counterparts, this drawback of HMC approximated BNNs can be potentially mitigated through the use of different priors. Finding the right priors for this (e.g. building on the results of Izmailov et al. (2021a)) is thus an important avenue for future research.

5

Using Bayesian Optimisation to Improve Query Efficiency of Adversarial Black-box Attacks

5.1 Introduction

So far, we have been mostly investigating worst-case adversarial robustness, that is, either robustness with guarantees (i.e. no attack could ever find an adversarial example) or robustness against the strongest white-box attacks (i.e. for any attack in full knowledge of the model it is hard or even impossible to find an adversarial example). However, concerning real-world applications, we might find ourselves much more often in the black-box scenario, that is in a scenario where the adversarial attack does not a-priori have information about the attacked model (see Section 2.2). Understanding how vulnerable a model is in this scenario can thus be more relevant from a practical viewpoint. In particular, it can be important to understand how many queries a black-box attack needs to find adversarial examples. Depending on the context, if this number is reasonably high, that might be sufficient to assume safety in practice since repeated query might be easily detected and flagged or simply be too costly for an attacker.

Thus, in this chapter we focus on the robustness of models against black-box

attacks aiming to contribute to its better understanding by proposing a novel, query efficient black-box attack.

In particular, we propose a query efficient black-box attack that iteratively optimises over both the adversarial perturbation and the effective dimensionality of the latent search space. Our main contributions are summarised as follows:

- We introduce a novel gradient-free black-box attack method, BayesOpt attack, which uses Bayesian optimisation with Gaussian process surrogate models to find the effective adversarial example and is capable of dealing with high-dimensional image inputs.
- We proposes a Bayesian technique, which learns the optimal degree of search dimensionality reduction by harnessing our probabilistic surrogate and information from query data. This technique can be incorporated naturally into our attack procedure, leading to efficient optimisation over both adversarial perturbation and the latent search space dimension.
- We empirically demonstrate that under the L_∞ constraint, our proposed attack method can achieve comparable success rate with about *2 to 5 times* fewer model queries in comparison to current state-of-the-art query-efficient black box attack methods.

5.2 Related Work

Most of the existing adversarial attacks (Madry et al., 2017; Moosavi-Dezfooli et al., 2016; Kurakin et al., 2016; Gu and Rigazio, 2014; Goodfellow et al., 2014; Chen et al., 2018; Carlini and Wagner, 2017b) focus on white-box settings, where the attacker can get full access to the target model and has complete knowledge of the architecture, weights and gradients to generate successful adversarial examples. However, real-world systems are usually attacked in a black-box setting, where one has no knowledge about the model and can only observe the input-output correspondences by querying the model. Here we give a brief overview of various existing black-box attacks.

Surrogate Model. One class of black-box attacks uses data acquired from querying the target black-box model to train a surrogate model (Papernot et al., 2017), which mimics the classification behaviour of the target model. The adversary can then employ any white-box method to attack the fully observable surrogate model and apply the successful adversarial example to the target model. Such approaches rely on the transferability assumption that adversarial examples which are effective to the surrogate model are also very likely to conceive the target model given their similar classification performance on the same data (Szegedy et al., 2013). Moreover, to provide training data for the substitute model, the adversary either requires information on the target model's training set, which is highly unrealistic in real-world applications, or needs to build a synthetic training set by querying the target model (Papernot et al., 2017), which implies a large number of model evaluations and becomes hardly feasible for large models or complex data sets (Brendel et al., 2017).

Gradient Estimation. An alternative to training a substitute model is to estimate the gradient via finite differences and use the estimated gradient information to produce attacks. However, the naive coordinate-wise gradient estimation requires excessive queries to the target model (2 queries per coordinate per descent/attack step) and thus is not feasible for attacking models with high-dimensional inputs (e.g. classifiers on ImageNet). Chen et al. (2017) overcome this limitation by using stochastic coordinate gradient descent and selecting the batch of coordinates via importance sampling, introducing the state-of-the-art zeroth-order attack ZOO, which achieves comparable attack success rate and perturbation costs as many white-box attacks. Although ZOO makes it computationally tractable to perform black-box attacks on high-dimensional image data, it still requires millions of queries to generate a successful adversarial example, making it impracticable for attacking real-world systems where each model query can be expensive and the query budget limited. Improving on ZOO, AutoZOOM (Tu et al., 2018) significantly enhances the query efficiency by using random vectors to estimate the full gradient and

adjusting the estimation parameter adaptively to trade off query efficiency vs. input perturbation cost. More importantly, AutoZOOM demonstrates the benefits of employing dimension reduction techniques in accelerating attacks (i.e. searching for adversarial perturbation in a low-dimensional latent space and projecting it back to the high-dimensional input space). Parallel work by Ilyas et al. (2018) estimate the gradient through a modified version of a natural evolution strategy which can be viewed as a finite-difference method over a random Gaussian basis. The estimated gradient is then used with projected gradient descent, a white-box attack method, to generate adversarial examples.

Gradient-free Optimisation. As discussed, gradient estimation approaches in general need an excessive number of queries to achieve successful attacks. Moreover, their dependence on the gradient information makes them less effective against defences which manipulate the gradients (Athalye et al., 2018; Brendel et al., 2017; Guo et al., 2017). To address this, gradient-free methods have been developed. One recent example, which has demonstrated state-of-the-art query efficiency, is GenAttack (Alzantot et al., 2018). GenAttack uses genetic algorithms to iteratively evolve a population of candidate adversarial examples. Besides having an annealing scheme for mutation rate and range, GenAttack also adopts dimensional reduction techniques, similar to AutoZOOM, to improve the query efficiency. In parallel to GenAttack, Brendel et al. (2017) introduce a decision-based attack, Boundary Attack, which only requires access to the final model decision. Boundary Attack starts from a huge adversarial perturbation and then iteratively reduces the perturbation through a random walk along the decision boundary. However, Boundary Attack takes about 72 times more queries than GenAttack to fool an undefended ImageNet model (Alzantot et al., 2018). Another recent approach introduced in Moon et al. (2019) reduces the search space to a discrete domain and subsequently uses combinatorial optimisation to find successful attacks, however mostly focusing on the less challenging setting of untargeted attacks.

Finally, the prior works that use Bayesian optimisation for adversarial attacks (Suya et al., 2017; Zhao et al., 2019) only investigate the use of simple Gaussian processes as the surrogate model and are thereby limited in their performance. The method proposed in Suya et al. (2017) deals with the untargetted attack setting and only demonstrates the effectiveness of Bayesian optimisation in comparison to random search on a low-dimensional ($d = 57$) email attack task. BO-ADMM proposed in Zhao et al. (2019) works on image data but it applies Bayesian optimisation directly on the search space of image dimension to minimise a joint objective of attack loss and distortion loss. Despite its query efficiency, BO-ADMM leads to poor-quality adversarial examples of large distortion loss.

5.3 Preliminaries

5.3.1 Problem Definition

We focus on the black-box attack setting, where the adversary has no knowledge about the network architecture, weights, gradient or training data of the target model prediction function $\hat{f}^{(n)}$, and can only query the target model with an input x to observe its prediction scores on all C classes (i.e. $\hat{f}^{(n)} : \mathbb{R}^d \longrightarrow [0,1]^C$) (Tu et al., 2018; Alzantot et al., 2018). Moreover, we aim to perform *targeted* attacks, which is more challenging than untargeted attacks, subject to a constraint on the maximum change to any of the coordinates (i.e., a L_∞-norm constraint) (Warde-Farley and Goodfellow, 2016; Alzantot et al., 2018). That is, given a valid input x^* of class y^* and a target $\bar{t} \neq y^*$, we aim to find an adversarial input x^{adv}, which is close to x^* according to the $L_\infty-$norm, such that $\arg\max_{i\in\{1,\dots,C\}} \hat{f}^{(n)}(x^{adv})_i = \bar{t}$.
In our approach, we follow the convention to optimise over the perturbation $\boldsymbol{\delta}$ instead of the adversarial example x^{adv} directly (Chen et al., 2017; Alzantot et al., 2018; Tu et al., 2018). Therefore, our problem can be formulated as (see Eq. (2.58):

$$\text{Find } \boldsymbol{\delta} \text{ with } \underset{i\in\{1,\dots,C\}}{\arg\max}\, \hat{f}^{(n)}(x^* + \boldsymbol{\delta})_i = \bar{t} \quad \text{s.t.} \quad \|\boldsymbol{\delta}\|_\infty \leq \gamma \tag{5.1}$$

5.3.2 Bayesian Optimisation

Bayesian optimisation (BayesOpt) is a query-efficient approach to tackle global optimisation problems (Brochu et al., 2010). It is particularly useful when the objective function is a black-box and is costly to evaluate. There are two key components in BO: a probabilistic surrogate model, such as a GP or a BNN which models the unknown objective function, and an acquisition function $\alpha(\cdot)$ operating on the surrogate model, which is maximised to select the location of the next evaluation of the objective function.

In its default formulation, Bayesian optimisation starts by generating a small initial data set $\mathcal{D}_0$ consisting of observations of the unknown objective function (e.g. by drawing 10 random samples of input points and evaluating the objective function at those points). Subsequently, it iterates back and forth between updating the surrogate model to take into account the most recently observed data and maximising the acquisition function $\alpha(\cdot)$ to select the next query point, until a pre-specified termination criterion (e.g. maximum number of iterations reached) is met. This is summarised in Algorithm 2 for a maximisation problem.

The acquisition function typically has a form such that maximising it when choosing the next query location is effectively trading off exploitation and exploration. For example, in this work, we use the upper confidence bound (UCB) acquisition function (for an overview of acquisition functions see Brochu et al. (2010)):

$$\alpha^{UCB}(x) = \mu_y(x) + \beta_t \sqrt{\Sigma_y(x)}, \tag{5.2}$$

where μ_y is the predictive mean and Σ_y is the predictive variance of the probabilistic surrogate model. β_t is the parameter that trades off exploration (locations with high predictive variance, i.e. uncertainty) and exploitation (locations with high predictive mean, i.e. likely high objective function values). If $\beta_t = \infty$, α^{UCB} corresponds to pure exploration and always proposes locations that have the highest predictive uncertainty as next query points, leading to near-random search like behaviour. If $\beta_t = 0$, α^{UCB} corresponds to pure exploitation and will only propose locations that have the high predictive mean values as next query points, which for

GP surrogate models will be often points fairly near already observed data points. Under mild assumptions (e.g. on kernel and objective function smoothness), with high probability, global convergence of the BayesOpt algorithm with α^{UCB} can be shown for specific, finite $\beta_t > 0$ (Srinivas et al., 2009).

Algorithm 2 Bayesian Optimisation Algorithm

1: **Input:** A black-box function $y(x)$, initial observation data $\mathcal{D}_0$, iteration budget T
2: **Output:** The best recommendation for location x^* of global maximum of y
3: **for** $t = 1, \ldots, T$ **do**
4: Select $x^{(t)} = \arg\max \alpha(x|\mathcal{D}_{t-1})$
5: $y^{(t)} = y(x^{(t)})$ and $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup (x^{(t)}, y^{(t)})$
6: Update the surrogate model with $\mathcal{D}_t$
7: **end for**
8: $x^* \leftarrow \arg\max_{\{x^{(t)}\}_{1,\ldots,T}} y(x^{(t)})$

5.4 BayesOpt Attack

5.4.1 BayesOpt attack objective

It has been shown that reducing the dimensionality of the search space in Eq. (5.1) increases query efficiency significantly (Chen et al., 2017; Tu et al., 2018; Alzantot et al., 2018). Due to our focus on the small query regime where our surrogate model needs to be trained with a very small number of observation data, we adopt the previously suggested dimensionality reduction technique, bilinear resizing, to reduce the challenging problem of optimising over the high-dimensional input space of $x^* \in \mathbb{R}^d$ to one over a relatively low-dimensional input space, setting $x^{adv} = x^* + g(\boldsymbol{\delta})$ where $\boldsymbol{\delta} \in \mathbb{R}^{d^r}$ with $d^r < d$ and $g : \mathbb{R}^{d^r} \longrightarrow \mathbb{R}^d$ being the bilinear resizing decoder[1].

Furthermore, we follow the approach of using a smoothing version of the CE loss (see Eq. (2.59)) as objective function to find a $\boldsymbol{\delta}$ fulfilling Eq. (5.1), which has been found to be beneficial in previous work (Chen et al., 2017; Tu et al., 2018; Alzantot et al., 2018). Together with the dimensionality reduction, this leads to

[1]Note that our method is not dependent on a particular choice of decoder, i.e. it could also be used together with an autoencoder- or PCA-based dimensionality reduction technique.

the following black-box objective problem for our Bayesian optimisation:

$$\boldsymbol{\delta}^* = \arg\max_{\boldsymbol{\delta}} y(\boldsymbol{\delta}) \quad \text{s.t.} \quad \boldsymbol{\delta} \in [-\gamma, \gamma]^{d^r} \tag{5.3}$$

$$\text{where} \quad y(\boldsymbol{\delta}) = \left[\log \hat{f}^{(n)}(x^* + g(\boldsymbol{\delta}))_{\bar{t}} - \log \sum_{j \neq \bar{t}}^{C} \hat{f}^{(n)}(x^* + g(\boldsymbol{\delta}))_j \right]$$

5.4.2 GP-based BayesOpt Attack

We first use BayesOpt with a standard GP as the surrogate to solve for the black-box attack objective in the reduced input dimension in Eq. (5.3). The GP encodes our prior assumptions on the objective y :

$$y \sim \mathcal{GP}(0, k(\boldsymbol{\delta}, \boldsymbol{\delta}')) \tag{5.4}$$

Since we normalise the objective function value using a zero-mean prior seems adequate. Based on the observation data $\mathcal{D}_{t-1} = \{\boldsymbol{\delta}^{(i)}, y^{(i)}\}_{i=1}^{t-1} = \{\boldsymbol{\delta}^{1:t-1}, \boldsymbol{y}^{1:t-1}\}$, the optimal GP hyper-parameters $\boldsymbol{\rho}^*$ such as the length scales and variance of the kernel function k are learnt using type-II-maximum likelihood, i.e. by maximising the marginal likelihood $p(\mathcal{D}_{t-1}|\boldsymbol{\rho})$ which has analytic form (see Eq. (2.45)[2]:

$$p(\mathcal{D}_{t-1}|\boldsymbol{\rho}) = (2\pi)^{-\frac{t-1}{2}} |\mathbf{K}_{1:t-1} + \sigma^2 I|^{-\frac{1}{2}} \mathrm{e}^{-\frac{1}{2}\boldsymbol{y}^{1:t-1}{}^T [\mathbf{K}_{1:t-1} + \sigma^2 I]^{-1} \boldsymbol{y}^{1:t-1}}, \tag{5.5}$$

where $\mathbf{K}_{1:t-1} = K_{\boldsymbol{\rho}}(\boldsymbol{\delta}^{1:t-1}, \boldsymbol{\delta}^{1:t-1})$ is the $t-1 \times t-1$ matrix with pairwise covariances $k_{\boldsymbol{\rho}}(\boldsymbol{\delta}^{(l)}, \boldsymbol{\delta}^{(m)})$, $l, m \leq t-1$ as entries, $|\cdot|$ is the determinant and σ^2 is the assumed noise variance.

The predictive posterior distribution for $y^{(t)}$ at a test point $\delta^{(t)}$ conditioned on $\mathcal{D}_{t-1}$ then has the form (see Section 2.1.3):

$$p(y^{(t)}|\boldsymbol{\delta}^{(t)}, \mathcal{D}_{t-1}) = \mathcal{N}(y^{(t)}|\mu_y(\cdot), k_y(\cdot,\cdot)) \tag{5.6}$$

where

$$\mu_y(\boldsymbol{\delta}^{(t)}) = k(\boldsymbol{\delta}^{(t)}, \boldsymbol{\delta}^{1:t-1}) \boldsymbol{K}_{1:t-1}^{-1} \boldsymbol{y}^{1:t-1}, \tag{5.7}$$

$$k_y(\boldsymbol{\delta}^{(t)}, \boldsymbol{\delta}^{(t)}) = k(\boldsymbol{\delta}^{(t)}, \boldsymbol{\delta}^{(t)}) - K(\boldsymbol{\delta}^{(t)}, \boldsymbol{\delta}^{1:t-1}) \boldsymbol{K}_{1:t-1}^{-1} K(\boldsymbol{\delta}^{1:t-1}, \boldsymbol{\delta}^{(t)}). \tag{5.8}$$

[2]Note that to facilitate the maximisation using gradient methods, we maximise $\log p(\mathcal{D}_{t-1}|\boldsymbol{\rho})$ to be precise.

Based on the predictive posterior distribution, we construct the acquisition function $\alpha^{UCB}(\boldsymbol{\delta}|\mathcal{D}_{t-1})$ to help select the next query point $\boldsymbol{\delta}^{(t)}$. As described, the acquisition function can be considered as a utility measure which balances the exploration and exploitation by giving higher utility to input regions where the functional value is high (high μ_y) and where the model is very uncertain (high k_y). The approach of using BayesOpt with a GP surrogate to attack the target model is summarised in Algorithm 3.

Algorithm 3 BayesOpt Attack

1: **Input:** A black-box function y, initial observation data $\mathcal{D}_0$, iteration budget T, a decoder $g(\cdot)$
2: **Output:** The best recommended adversarial example $x^{adv} = x^* + g(\boldsymbol{\delta}^*)$
3: **for** $t = 1, \ldots, T$ **do**
4: Select $\boldsymbol{\delta}^{(t)} = \arg\max \alpha_t(\boldsymbol{\delta}|\mathcal{D}_{t-1})$
5: $y^{(t)} = y(\boldsymbol{\delta}^{(t)})$ and $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup (\boldsymbol{\delta}^{(t)}, y^{(t)})$
6: Update the surrogate model with $\mathcal{D}_t$
7: **end for**
8: $\boldsymbol{\delta}^* \leftarrow \arg\max_{\{\boldsymbol{\delta}^{(t)}\}_{1,\ldots,T}} y(\boldsymbol{\delta}^{(t)})$

5.4.3 Additive GP-based BayesOpt Attack

Although techniques such as bilinear resizing are able to reduce the input dimension from the original image size (e.g. $d = 3,072$ for CIFAR10 image) to a much lower dimension (e.g. $d' = 192$), the reduced search space for the adversarial attack is still considered very high-dimensional for GP-based BayesOpt (which is usually applied to problems with $d' \leq 20$).

There are two challenges in using BayesOpt for high-dimensional problems. The first is the curse of dimensionality in modelling the objective function. When the unknown function is high-dimensional, estimating it with non-parametric regression becomes very difficult because it is impossible to densely fill the input space with finite numbers of sample points, even if the sample size is very large (Györfi et al., 2006). The second challenge is the computational difficulty in optimising the acquisition function. The computation cost needed for optimising the acquisition function to within a desired accuracy grows exponentially with dimension (Kandasamy et al., 2015).

We adopt the additive-GP model (Duvenaud et al., 2011; Kandasamy et al., 2015) to deal with the above-mentioned challenges associated with using GPs for our purposes. The key assumption we make is that the objective can be decomposed into a sum of low-dimensional composite functions:

$$y(\delta) = \sum_{j=1}^{M} y_j(\delta_{A_j}) \tag{5.9}$$

where $\boldsymbol{\delta}_{A_j}$ denotes disjoint low-dimensional subspaces, $\boldsymbol{\delta} = \cup_{j=1}^{M} \boldsymbol{\delta}_{A_j}$ and $\boldsymbol{\delta}_{A_j} \cap \boldsymbol{\delta}_{A_i} = \emptyset$ for all $j \neq i$. If we impose a GP prior for each y_j, the prior for the overall objective y is also a GP: $y \sim \mathcal{GP}(0, k(\boldsymbol{\delta}, \boldsymbol{\delta}'))$ where $k(\boldsymbol{\delta}, \boldsymbol{\delta}') = \sum_{j=1}^{M} k_j \left(\boldsymbol{\delta}_{A_j}, \boldsymbol{\delta}'_{A_j}\right)$. The predictive posterior distribution for each subspace $p(y_j^{(t)} | \boldsymbol{\delta}_{A_j}^{(t)}, \mathcal{D}_{t-1})$ is again Gaussian with:

$$\begin{aligned}
\mu_{y_j}(\boldsymbol{\delta}_{A_j}^{(t)}) &= \boldsymbol{k}_j(\boldsymbol{\delta}_{A_j}^{(t)}, \boldsymbol{\delta}_{A_j}^{1:t-1}) \boldsymbol{K}_{1:t-1}^{-1} \mathbf{y}^{1:t-1}, \\
k_{y_j}(\boldsymbol{\delta}_{A_j}^{(t)}, \boldsymbol{\delta}_{A_j}^{(t)}) &= k_j(\boldsymbol{\delta}_{A_j}^{(t)}, \boldsymbol{\delta}_{A_j}^{(t)}) - K_j(\boldsymbol{\delta}_{A_j}^{(t)}, \boldsymbol{\delta}_{A_j}^{1:t-1}) \mathbf{K}_{1:t-1}^{-1} K_j(\boldsymbol{\delta}_{A_j}^{1:t-1}, \boldsymbol{\delta}_{A_j}^{(t)}).
\end{aligned}$$

In our case, the exact decomposition (i.e. which input dimension belongs to which low-dimensional subspace) is unknown but we can infer it together with other GP hyperparameters by maximising marginal likelihood (Kandasamy et al., 2015). The maximisation is done using a greedy approach choosing the best of 20 random decompositions, as we found this approach to work better than alternative approaches and exhaustive evaluation of the marginal likelihood for all possible decompositions is computationally intractable (see Section 5.5.5). Note that for the additive GP, the acquisition function is formulated based on $p(y_j^{(t)} | \boldsymbol{\delta}_{A_j}^{(t)}, \mathcal{D}_{t-1})$ for each subspace and is optimised in each low-dimensional subspace, thus leading to a much more efficient optimisation task. The optimal perturbations in all the subspaces $\{\boldsymbol{\delta}_{A_j}^*\}_{j=1}^{M}$ are then combined to give the next query point $\boldsymbol{\delta}^{(t)}$.

5.4.4 Learning The Optimal d^r

Generating the successful adversarial example $x^{adv} \in \mathbb{R}^d$ by searching perturbations in a reduced dimension $\boldsymbol{\delta} \in \mathbb{R}^{d^r}$ has become a popular practice that leads to significant improvement in query efficiency (Chen et al., 2017; Tu et al., 2018; Alzantot et al., 2018). However, what the optimal d^r is and how this can be decided

efficiently has not been investigated in previous work (Chen et al., 2017; Tu et al., 2018; Alzantot et al., 2018). As we show empirically in Section 5.5.2, setting d^r arbitrarily can lead to suboptimal attack performance in terms of query efficiency, attack success rate as well as perturbation costs while finding a good d^r by trial and error or progressive increasing is computationally expensive and highly inefficient because the optimal d^r varies with the attack input x^*. An effective way of learning d^r is thus very important for adversarial attacks. In this section, we propose a principled method, which is neatly compatible with our Bayesian attack technique, to learn the optimal d^r from the query information.

The optimal d^r should be the one that both takes into consideration our prior knowledge on the discrete d^r choices and at the same time best explains the observed query data. Given that our BayesOpt attack uses a probabilistic surrogate (i.e. GP) to model the unknown relation between the attack objective score y and the adversarial perturbation $\boldsymbol{\delta}$, this naturally translates to the criterion of maximising the posterior for d^r:

$$p(d_j^r|\mathcal{D}_{t-1}) = \frac{p(\mathcal{D}_{t-1}|d_j^r)p(d_j^r)}{p(\mathcal{D}_{t-1})}, \tag{5.10}$$

where $p(\mathcal{D}_{t-1}|d_j^r)$ is the marginal likelihood (evidence) of adopting a specific d_j^r, $p(d_j^r)$ is our prior over the possible d^r values and $p(\mathcal{D}_{t-1}) = \sum_j p(\mathcal{D}_{t-1}|d_j^r)p(d_j^r)$ is the normalising constant. Since different d_j^r correspond to different kernel functions on the original data space $\mathbb{R}^d$ [3], we can match different d_j^r to different model choices. The problem of choosing d^r thus becomes a Bayesian model selection task (see also Section 2.1.3). Our prior assumption is that we do not prefer one d^r over another, i.e. we use a uniform prior with $p(d_j^r) = p(d_i^r)$ for $j \neq i$ and thus we can select d^r by comparing their evidence (MacKay, 2003):

$$\frac{p(d_i^r|\mathcal{D}_{t-1})}{p(d_j^r|\mathcal{D}_{t-1})} = \frac{p(\mathcal{D}_{t-1}|d_i^r)p(d_i^r)}{p(\mathcal{D}_{t-1}|d_j^r)p(d_j^r)} = \frac{p(\mathcal{D}_{t-1}|d_i^r)}{p(\mathcal{D}_{t-1}|d_j^r)}. \tag{5.11}$$

[3] If k_j is a kernel function on $\mathbb{R}^{d_j^r}$, then $\tilde{k_j}(x, x') := k_j(g_j^{-1}(x), g_j^{-1}(x'))$ is a kernel function on $\mathbb{R}^d$, as it inherits symmetry and positive definiteness from k_j (see also Shawe-Taylor et al. (2004), Ch. 3). Yet, in general, $\tilde{k_j}(x, x') \neq \tilde{k_i}(x, x')$ for $i \neq j$ due to the different degrees of compression associated to the decoders $g_j : \mathbb{R}^{d_j^r} \to \mathbb{R}^d$ and $g_i : \mathbb{R}^{d_i^r} \to \mathbb{R}^d$

The exact computation of the evidence term requires marginalisation over model hyper-parameters, which is intractable. We approximate the integral with point estimates:

$$p(\mathcal{D}_{t-1}|d_j^r) = \int p(\mathcal{D}_{t-1}|\boldsymbol{\rho}, d_j^r)p(\boldsymbol{\rho}|d_j^r)d\boldsymbol{\rho} \approx p(\mathcal{D}_{t-1}|\boldsymbol{\rho}^*, d_j^r), \tag{5.12}$$

where $\boldsymbol{\rho}^* = \arg\max_{\boldsymbol{\rho}} p(\mathcal{D}_{t-1}|\boldsymbol{\rho}, d_j^r)$, i.e. with the marginal likelihood of our GP model after type-II maximum likelihood over hyperparameters. The implementation of this procedure is summarised in Algorithm 4. First, we project the same perturbation query data in the original image dimension $\mathcal{D}_{t-1}^d = \{g(\boldsymbol{\delta}^{(i)}), y^{(i)}\}_{i=1}^{t-1}$ to different latent spaces to get corresponding low-dimensional training data sets for separate GP models. The GP model that corresponds to d_j^r is then trained on $\mathcal{D}_{t-1}^{d_j^r} = \{g^{-1}\left(g(\boldsymbol{\delta}^{(i)})\right), y^{(i)}\}_{i=1}^{t-1}$ where $g^{-1}\left(g(\boldsymbol{\delta}^{(i)})\right) \in \mathbb{R}^{d_j^r}$ for each d_j^r. Finally we select the optimal d^r by comparing the marginal likelihood $p(\mathcal{D}_{t-1}^{d_j^r}|\boldsymbol{\rho}^*, d_j^r)$ of each GP surrogate.

Most importantly, this approach to selecting d^r does not require any extra target model queries, as it is entirely done using already queried/observed data. However, we would like to highlight that beyond this, the use of the probabilistic surrogate model in our BayesOpt attack approach enables us to naturally use the Bayesian model selection technique to learn the optimal d^r which benefits from the optimal trade-off between data-fit and model complexity. As we show empirically in Section 5.5.4, automating and incorporating the learning of d^r into our BayesOpt attack in such a way ultimately leads to higher attack success rates and query efficiency. Other adversarial attack methods could in theory also use our proposed approach for d^r selection, but it would require the additional efforts of constructing probabilistic models to obtain $p(\mathcal{D}_{t-1}|d_j^r)$.

5.5 Experimental Results

5.5.1 Experimental Set Up

We empirically compare the performance of our BayesOpt attacks against state-of-the-art black-box methods on 3 benchmark image data sets: MNIST, CIFAR10

Algorithm 4 Bayesian selection of d^r

Input: A decoder $g(\cdot)$, observation data $\mathcal{D}_{t-1}^d = \{g(\boldsymbol{\delta}^{(i)}), y^{(i)}\}_{i=1}^{t-1}$ where $g(\boldsymbol{\delta}^{(i)}) \in \mathbb{R}^d$ and a set of possible $d^r : \{d_j^r\}_{j=1}^N$
Output: The optimal reduced dimension d^{r*} and the corresponding GP model
for $j = 1, \ldots, N$ **do**
  $\mathcal{D}_{t-1}^{d_j^r} = \{g^{-1}\left(g(\boldsymbol{\delta}^{(i)})\right), y^{(i)}\}_{i=1}^{t-1}$ where $g^{-1}\left(g(\boldsymbol{\delta}^{(i)})\right) \in \mathbb{R}^{d_j^r}$
  Fit a GP model to $\mathcal{D}_{t-1}^{d_j^r}$ and computes its maximum marginal likelihood $p(\mathcal{D}_{t-1}^d | \boldsymbol{\rho}^*, d_j^r)$
end for
Select $d^{r*} = \arg\max_{d_j^r \in \{d_j^r\}_{j=1}^N} p(\mathcal{D}_{t-1}^d | \boldsymbol{\rho}^*, d_j^r)$ and its correspond GP model

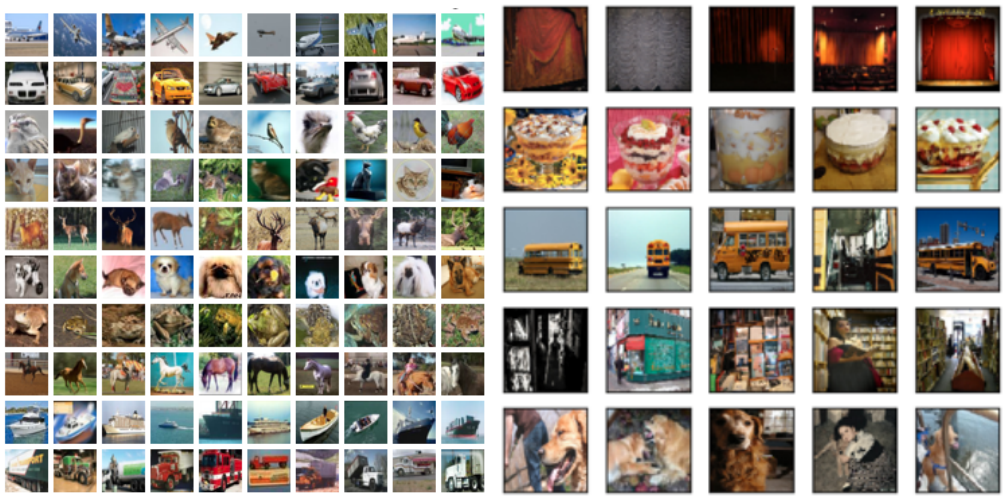


Figure 5.1: Left: Sample images from CIFAR10 data set. Shown are 10 random samples for each of the 10 classes. **Right:** Sample images from ImageNet data set. Shown are 5 random samples for 5 random classes.

and ImageNet.

MNIST (LeCun, 1998) is known from the previous chapters. CIFAR10 (Krizhevsky et al., 2009) consists of 60,000 colour images in 10 classes, split into 50,000 training images and 10,000 test images. Figure 5.1 shows 10 random samples from each class. The images have a dimensionality of 32×32 pixels in each of the 3 RGB channels, i.e. a total of $d = 3,072$. ImageNet (Russakovsky et al., 2015) consists of 1,000,000 colour images in 1,000 classes, split into 800,000 training images and 200,000 test images. Figure 5.1 shows 5 random samples from 5 random classes of the data set. The images have a dimensionality of 96×96 pixels in each of the 3 RGB channels, i.e. a total of $d = 27,648$.

The methods we compare against are ZOO (Chen et al., 2017), AutoZOOM (Tu et al., 2018) and GenAttack (Alzantot et al., 2018). We denote the BayesOpt attack with standard GP surrogate as GP-BO, its variant that learns the d^r automatically is GP-BO-auto-d^r as well as the attack with additive GP surrogate as ADDGP-BO. For all the BayesOpt methods, we use GP-UCB as the acquisition function[4] and update the GP hyperperameters $\boldsymbol{\rho}$ every 5 BayesOpt iterations. We relearn the optimal d^r for GP-BO-auto-d^r and the search space decomposition for ADDGP-BO every 40 iterations. For ADDGP-BO, we decompose the search spaces into $M = 12$ sub-spaces of equal dimensions for MNIST and CIFAR10 and into $M = 27$ sub-spaces for ImageNet. For all GP surrogates, we use the Matern-5/2 kernel in our work.

Remark 6 (A note on kernel selection)**.** *The reason we chose the Matern-5/2 kernel is that after experimenting with all kernels presented in Section 2.1.3, we found that this gave us the best results in terms of attack success rate and query efficiency on average. A noteworthy alternative consisted in the linear kernel, which we tried using inspired by the work of Moon et al. (2019). They use a Taylor series expansion of the loss function around the attacked test point and argue that the small magnitude of the perturbation budget makes terms of order ≥ 2 disappear, thereby imposing a locally linear behaviour on the target model. This lets them achieve strong empirical results with their accordingly chosen attack and optimisation method. Among all kernels, the linear kernel should capture such a locally linear behaviour best, and indeed we found that for a large portion of attack instances, the linear kernel did indeed outperform the other kernels (including the Matern-5/2 kernel) by a small but significant margin. However, on the remaining attack instances, the linear kernel needed up to $8\times$ more queries to find successful attacks, which lead to overall worse results for the linear kernel when compared to the Matern-5/2 kernel. These findings indicate that while the local linearity assumption of Moon et al. (2019) is often accurate, it is not always the case in practice.*

[4]The parameter that trades off exploration and exploitation is set to $\beta_t = 2$.

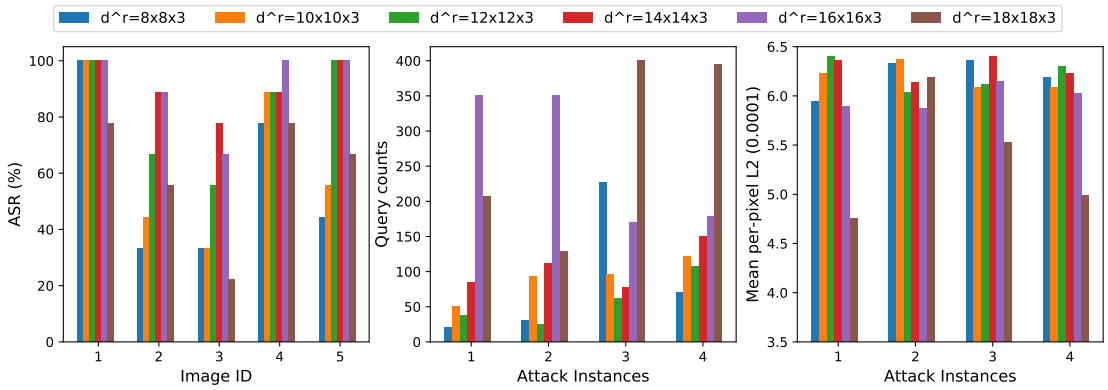


Figure 5.2: Performance of BayesOpt attack with GP surrogate in different reduced dimension d^r. Each color represents a particular d^r. **Left:** Attack success rates (ASR) for attacking all the other 9 classes on each of the 5 original images. The d^r that leads to the highest ASR varies with the original images. **Middle:** Number of queries needed to attack the same attack instances (i.e. the same original image and the same attack target) by using different d^r. **Right:** Effect of the choice of d^r on the L_2 distances between the resulting adversarial examples and the original image (for the same 4 attack instances).

The target models that we attack have the same architectures and parameters as those used in AutoZOOM and GenAttack. They are neural network classification models with test accuracies of 99.5% for MNIST, 80% for CIFAR10 and 78% for ImageNet. Following the experiment design in Tu et al. (2018), we randomly select 50 correctly classified test images from each test data set. We then perform targeted attacks on these images. For MNIST and CIFAR10, each selected image is attacked 9 times, targeting all but its true class, giving a total of 450 attack instances. For ImageNet, we also select 50 correctly classified images from the test set but perform only one random targeted attack for each image for computational reasons. We set $\gamma = 0.3$ for attacking MNIST and $\gamma = 0.05$ for CIFAR10 and ImageNet, which are used in Alzantot et al. (2018). We use the recommended parameter settings and their open sourced implementations for performing all competing attack methods (Chen et al., 2017; Tu et al., 2018; Alzantot et al., 2018).

5.5.2 Effect of d^r

We first empirically investigate the effect of the reduced dimensionality d^r of the latent space in which we search for the adversarial perturbation. We experiment with the GP-based BayesOpt attacks for the CIFAR10 classification model using

reduced dimensions of $d_r = \{6 \times 6 \times 3, 8 \times 8 \times 3, 10 \times 10 \times 3, 12 \times 12 \times 3, 14 \times 14 \times 3, 16 \times 16 \times 3, 18 \times 18 \times 3\}$ and perform targeted attacks on 5 target images with each image being attacked 9 times, leading to 45 attack instances. We first investigate the attack success rate (ASR) achieved at different d^r for all the 5 images. The results on ASR out of the 9 targeted attacks for each of the 5 images, which are indicated by Image ID 1 to 5, are shown in the left subplot of Figure 5.2. It's evident that the d^r which leads to highest attack success rate varies for different original images x^*.

We then examine how the d^r affect the query efficiency and attack quality (average L_2 distance of the adversarial image from the original image) for the attack instances (e.g. cause the model to misclassify an airplane image as a cat) that GP-based BayesOpt can attack successfully at all d^r. We present the results on 4 attack instances of an airplane image in the middle and right subplots of Figure 5.2. We can see that even for the same original image and attack instances, varying d^r can impact query efficiency and L_2 norm of the successful adversarial perturbation significantly. For example, $d^r = 8 \times 8 \times 3$ is most query efficient for attack instance 1 and 4 but is outperformed by other dimensions in attack instance 2 and 3. Therefore, the importance of d^r and the difficulty of finding the optimal d^r for a specific target/image emphasises the need for a method for learning it automatically from the data. In Section 5.5.5 we also include some further results on MNIST and CIFAR10 that support the need for it. In the next sections, we show that our d^r learner does indeed lead to more superior attack performance.

5.5.3 Performance Under a Fixed Query Budget

In this experiment, we limit the total query number to $1,000$ to make a successful attack on MNIST and CIFAR10 (slightly above the median query counts needed for GenAttack (Alzantot et al., 2018)) and $2,000$ for ImageNet. For GP-BO and ADDGP-BO, we adopt the reduced search dimension recommended by AutoZOOM, $d^r = 14 \times 14 \times 3$ for CIFAR10 and $d^r = 14 \times 14 \times 1$ for MNIST. For GP-BO-auto-d^r we learn the reduced dimension d^r in a range between $6 \times 6 \times 1$ and $28 \times 28 \times 1$ for MNIST, and between $6 \times 6 \times 3$ and $32 \times 32 \times 3$ for CIFAR10.

Table 5.1: Attack results on 50 random MNIST images by using $d^r = 14 \times 14 \times 1$ under a query budget of $1,000$. Q denotes the query count. ASR denotes attack success rate. The standard deviations are in parentheses.

Attack method	*ASR*	*Q (Max, Median, Mean)*	Avg. L_2 pert. (per pixel)
ADDGP-BO	97%	$829, 46, 95(6)$	$6.81 \times 10^{-3}(3.12 \times 10^{-5})$
GP-BO-auto-d^r	98%	$833, 75, 121(6)$	$6.89 \times 10^{-3}(5.55 \times 10^{-5})$
GP-BO	82%	$899, 42, 77(5)$	$7.15 \times 10^{-3}(4.17 \times 10^{-5})$
GenAttack	92%	$986, 146, 217(9)$	$6.66 \times 10^{-3}(2.81 \times 10^{-5})$
AutoZOOM	99%	$998, 229, 265(9)$	$5.05 \times 10^{-3}(7.71 \times 10^{-5})$
ZOO	1%	$256, 128, 128(64)$	$2.78 \times 10^{-3}(3.53 \times 10^{-5})$

Table 5.2: Attack results on 50 randomly selected CIFAR10 images with $d^r = 14 \times 14 \times 3$ under a query budget of $1,000$. Q denotes the query count. ASR denotes attack success rate.The standard deviation are in parentheses.

Attack method	*ASR*	*Q (Max, Median, Mean)*	Avg. L_2 pert. (per pixel)
ADDGP-BO	87%	$885,\ 154,\ 222(10)$	$5.87 \times 10^{-4}(3.22 \times 10^{-6})$
GP-BO-auto-d^r	87%	$891, 159, 234(10)$	$5.96 \times 10^{-4}(5.00 \times 10^{-6})$
GP-BO	72%	$899, 141, 190(8)$	$5.78 \times 10^{-4}(4.67 \times 10^{-6})$
GenAttack	68%	$991, 246, 329(14)$	$7.38 \times 10^{-4}(4.14 \times 10^{-6})$
AutoZOOM	38%	$896, 102, 154(11)$	$9.97 \times 10^{-4}(1.53 \times 10^{-5})$
ZOO	4%	$768,\ 256, 369(57)$	$1.77 \times 10^{-4}(1.16 \times 10^{-5})$

For ImageNet, due to the higher dimensionality, we perform BayesOpt attacks with ADDGP-BO and GP-BO on a reduced dimension of $d^r = 48 \times 48 \times 3$. As for the GP-BO-auto-$d^r$, we allow the algorithm to automatically learn the reduced dimension d^r in the range between $6 \times 6 \times 3$ to $60 \times 60 \times 3$. GenAttack is performed with the same upsampling as our methods. As for AutoZOOM and ZOO, we use the optimal d^r settings recommended in Tu et al. (2018) and Chen et al. (2017) for ImageNet.

For BayesOpt attack, each iteration requires 1 query to the objective function so we limit its maximum iteration number to $1,000$. AutoZOOM comprises two stages in its attack algorithm. The first exploration stage aims to find the successful adversarial example. Once a successful attack is found, it switches to the fine-

Table 5.3: Attack results on 50 randomly ImageNet images with random target label under a query budget of $2,000$. Q denotes the query count. ASR denotes attack success rate. The standard deviations are in parentheses. ZOO fails to make any successful attack within this budget.

Attack method	*ASR*	*Q (Max, Median, Mean)*	Avg. L_2 pert. (per pixel)
ADDGP-BO	60%	$1985, 1247, 1206(90)$	$1.74 \times 10^{-4}(1.89 \times 10^{-6})$
GP-BO-auto-d^r	32%	$1999, 922, 938(128)$	$1.96 \times 10^{-4}(8.07 \times 10^{-6})$
GP-BO	16%	$1947, 1113, 1232(162)$	$1.62 \times 10^{-4}(7.94 \times 10^{-6})$
GenAttack	12%	$1971, 1354, 1266(259)$	$2.03 \times 10^{-4}(9.81 \times 10^{-6})$
AutoZOOM	2%	$1451, 1451, 1451(0.00)$	$1.21 \times 10^{-5}(0.00)$

tuning stage to reduce the perturbation cost (e.g. L_2-norm) of the successful attack. We report its performance regarding the attack success rate and the L_2-norm of adversarial perturbations after a budget of $1,000$ queries. Moreover, AutoZOOM uses the L_2-norm to measure the perturbation costs but our method and GenAttack limit the search space via the $L_\infty-$norm. We observe that the adversarial examples found by AutoZOOM incur much higher perturbation costs in terms of $L_\infty-$norm than GenAttack and our method. Therefore, we only consider the final adversarial examples whose L_∞ distances from the original images lie within $[-\gamma, \gamma]$ as successful attacks.

The results on MNIST in Table 5.1 show that all our BayesOpt attacks can achieve a comparable attack success rate with a much lower query count than GenAttack (Alzantot et al., 2018) and AutoZOOM (Tu et al., 2018). Specifically, the median query count of ADDGP-BO is 68% less than of GenAttack and 80% less than of AutoZOOM while that of GP-BO-auto-d^r is 49% less than of GenAttack and 67% less than of AutoZOOM.

As for results on attacking on CIFAR10, Table 5.2 shows that all our BayesOpt attacks can achieve significantly higher success rates but again at remarkably lower query counts than the existing black-box approaches. For example, ADDGP-BO can achieve 18% higher success rate while using 37% fewer queries in terms of the median query count compared to GenAttack. In addition, our approaches also lead to better quality adversarial examples which are closer to the original image

than the benchmark methods as reflected by the lower average L_2 perturbation (20.5% less). Some examples of the adversarial examples found by our method can be found in Figure 5.3.

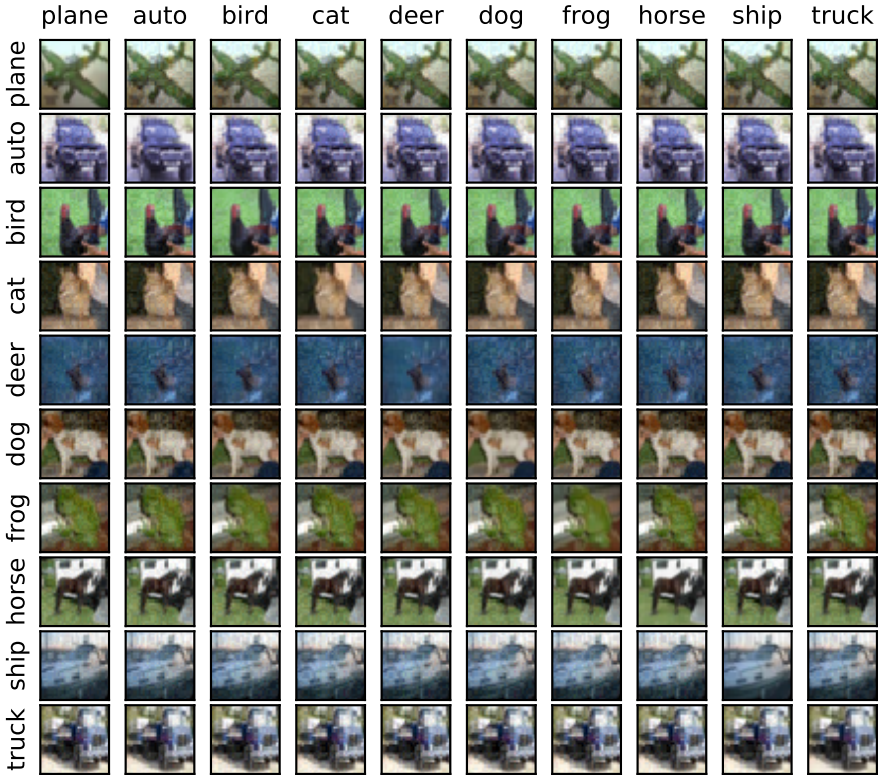


Figure 5.3: CIFAR10 adversarial examples generated by our BayesOpt attack. True labels and target labels correspond to the rows and columns.

Lastly, this set of experiments also demonstrates the effectiveness of our Bayesian model selection approach for learning d^r, as GP-BO-auto-d^r leads to 15% increase in attack success rate compared to GP-BO while maintaining competitiveness in query efficiency and L_2 distance.

Similarly, the results on ImageNet in Table 5.3 shows that all our BayesOpt attacks can achieve higher attack success rates than the competing methods for the given query budget of $2,000$. ADDGP-BO is clearly the best performing method in this very high-dimensional setting, achieving 5 times higher success rate than the best competing method (GenAttack) while again obtaining successful adversarial perturbations with lower average L_2 perturbation (by 14 %) than GenAttack. This

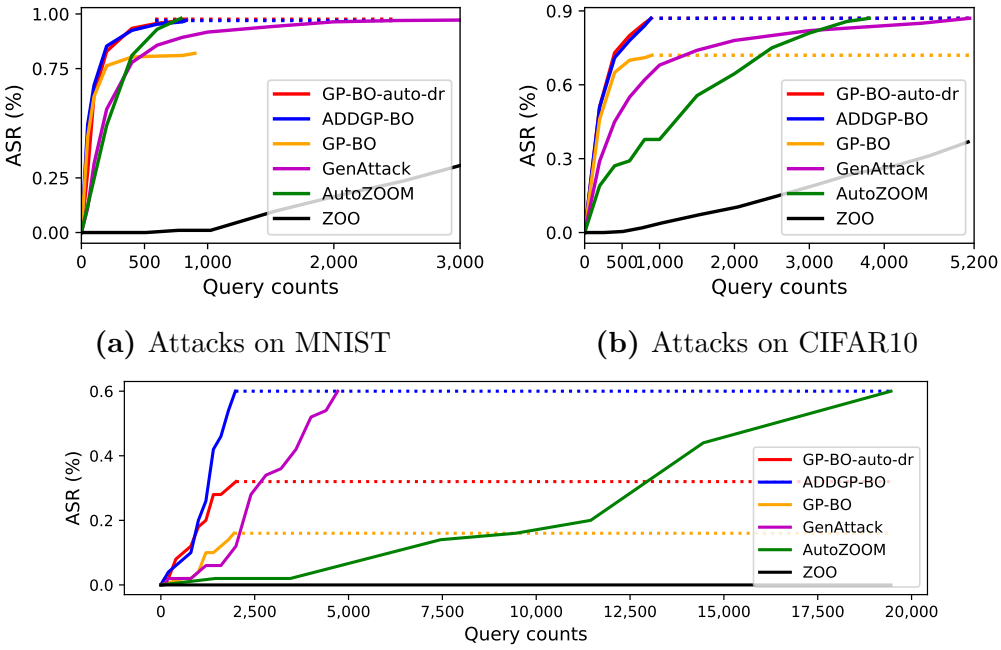
confirms our hypothesis that the additive-GP surrogate together with decomposition learning is very effective for high-dimensional optimisation.

Lastly, this set of experiments also shows that gradient-estimation-based methods are in general query inefficient, which is in line with the finding in GenAttack (Alzantot et al., 2018). AutoZOOM, despite being one of the most query efficient gradient estimation methods, achieves significantly lower success rates and produces adversarial examples that have the highest average L_2 distance within the respective query budget for all three data sets except MNIST. ZOO achieves even lower success rates.

5.5.4 Query Efficiency Comparison

We finish by comparing the query efficiency, measuring the change in the attack success rate over query counts for all the methods. We limit the query budget of our BayesOpt attacks to $1,000$ but let the competing methods to continue running until they achieve the same attack success rate as our best BayesOpt attacks or exceed a much higher query limit ($2,000$ queries for MNIST, $5,000$ queries for CIFAR10 and $14,000$ queries for ImageNet).

As shown in Figure 5.4, BayesOpt attacks converge much faster to the high attack success rates than the other methods. Specifically, GP-BO-auto-d^r takes less than 833 queries to achieve the success rate of 98% for MNIST, which is 24% of the queries required by GenAttack ($2,500$). As for CIFAR10, both ADDGP-BO and GP-BO-auto-d^r take around 890 queries to achieve a success rate of 87%, which is 23% of the query count needed by AutoZOOM and 17% of that needed by GenAttack. One point to note is that AutoZOOM appears to be slightly more query efficient than GenAttack in this set of experiments. In the case of ImageNet, all our BayesOpt attacks again enjoy faster successes, with ADDGP-BO taking only $1,985$ queries to achieve 60% ASR. Meanwhile, GenAttack, needs a budget of $4,711$ queries, 2.4 times that of ADDGP-BO, to achieve the same ASR. AutoZOOM needs $19,451$ queries which is 9.8 times higher than for ADDGP-BO.



(a) Attacks on MNIST **(b)** Attacks on CIFAR10

(c) Attacks on ImageNet

Figure 5.4: Query efficiency of BayesOpt Attacks. The plots show the attack success rate (ASR) of different methods up to certain query counts. The best BayesOpt attacks (i.e. GP-BO-auto-d^r (red) and ADDGP-BO (blue)) can achieve an ASR of 98% on MNIST(a) and 87% on CIFAR10 (b) within a budget of 1000 queries. To achieve the same success rates, GenAttack (purple) takes 3500 for MNIST and 5141 for CIFAR10, and AutoZOOM (green) takes 800 for MNIST and 3880 for CIFAR10. For ImageNet (c), the best BayesOpt attack is ADDGP-BO (blue), which achieves an ASR of 60% with 1985 queries. To achieve the same success rates, GenAttack (purple) takes a budget of 4711 queries and AutoZOOM (green) takes 19451 queries. ZOO fails to make any attack on ImageNet within the given budget.

5.5.5 Additional Experiments on MNIST and CIFAR10

Further experiments on the effect of d^r. We have already seen in Section 5.5.2 that the optimal reduced dimensionality can depend on both image and target. In Tables 5.4 and 5.5 we present some more findings across a series of random test points of MNIST and CIFAR10 that further underline this and also show that it is true for other black-box attacks such as AutoZOOM and GenAttack as well.

Objective Value over BayesOpt Iterations. In Figure 5.5, we illustrate how different BayesOpt variants compare in optimising the objective in Eq. (5.3) via the case of attacking a CIFAR10 image of label 9 (class truck) on the other

Table 5.4: Summary of attack results on 7 MNIST images. Q denotes the query count. ASR denotes attack success rate. The standard deviations are in parentheses.

Attack	d_r	*ASR*	*Q (Max, Med., Mean)*	$10^3\times$ avg. L_2 pert.
GP-BO	$14 \times 14 \times 1$	92%	899, 53, 119(28)	7.01(0.11)
ADDGP-BO	$14 \times 14 \times 1$	98%	584, 34, 67(13)	6.50(0.08)
AutoZOOM	$14 \times 14 \times 1$	97%	892, 186, 247(27)	5.10(0.19)
GenAttack	$14 \times 14 \times 1$	97%	976, 184, 239(22)	5.56(0.09)
GP-BO	$16 \times 16 \times 1$	97%	400, 50, 80(12)	7.08(0.11)
ADDGP-BO	$16 \times 16 \times 1$	100%	540, 42, 72(12)	6.49(0.09)
AutoZOOM	$16 \times 16 \times 1$	98%	812, 251, 295(28)	5.98(0.20)
GenAttack	$16 \times 16 \times 1$	97%	928, 202, 237(19)	5.61(0.09)
GP-BO	$28 \times 28 \times 1$	98%	430, 201, 225(11)	8.71(0.19)
ADDGP-BO	$28 \times 28 \times 1$	100%	666, 57, 100(15)	9.71(0.11)
AutoZOOM	$28 \times 28 \times 1$	98%	860, 185, 244(27)	10.1(0.16)
GenAttack	$28 \times 28 \times 1$	83%	976, 365, 399(33)	6.26(0.07)

9 target labels with original label 9. We plot the value of objective function (Eq. (5.3)), which is equal to the negative of attack loss, against the BayesOpt iterations/query counts. We can see that our additive GP surrogate (ADDGP-BO) as well as the Bayesian learning of optimal d^r (GP-BO-auto-d^r) lead to faster convergence and thus higher attack success rate for this instance than plain GP based Bayesian optimisation (GP-BO).

Effectiveness of Decomposition Learning. To learn the decomposition for the additive-GP surrogate in ADDGP-BO attack, we follow the approach proposed in Kandasamy et al. (2015) to treat the decomposition as an additional hyperparameter and learn the optimal decomposition by maximising marginal likelihood. However, exhaustive search over all $M!d!/(d_s!^M)$[5] possible decompositions is expensive. We adopt a computationally cheap alternative by randomly selecting 20 decompositions and choosing the one with the largest marginal likelihood. The method

[5] M is the number of subspaces and $d_s = |A_j|$ is the dimension of each subspace

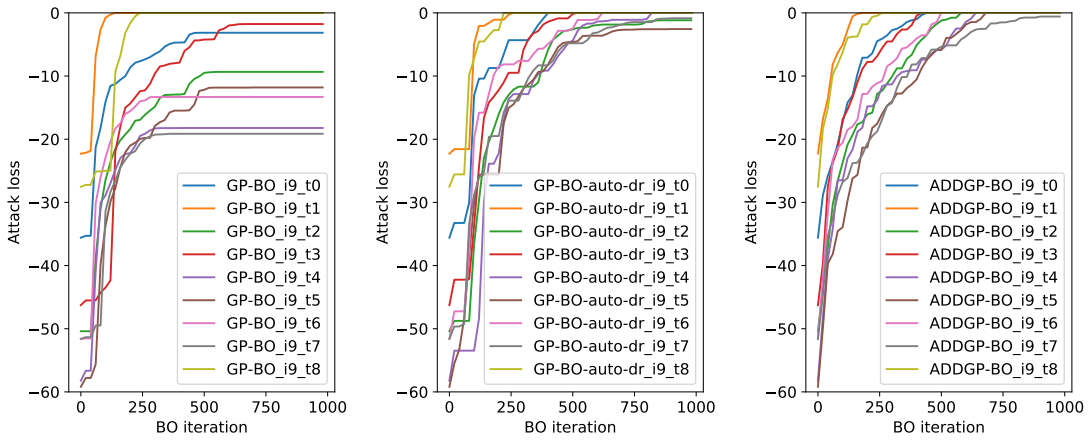
Table 5.5: Summary of attack results on 27 CIFAR10 images. Q denotes the query count. *ASR* denotes attack success rate. The standard deviations are in parentheses.

Attack	d_r	*ASR*	*Q (Max, Med., Mean)*	$10^4\times$ avg. L_2 pert.
GP-BO	$8 \times 8 \times 3$	52%	$314, 48, 70(13)$	5.78(0.14)
ADDGP-BO	$8 \times 8 \times 3$	75%	$899, 140, 234(33)$	5.55(0.08)
AutoZOOM	$8 \times 8 \times 3$	56%	$382, 126, 139(17)$	9.58(0.27)
GenAttack	$8 \times 8 \times 3$	67%	$671, 236, 286(32)$	7.40(0.13)
GP-BO	$14 \times 14 \times 3$	81%	$899, 142, 192(11)$	5.74(0.07)
ADDGP-BO	$14 \times 14 \times 3$	90%	$885, 141, 213(15)$	5.79(0.05)
AutoZOOM	$14 \times 14 \times 3$	42%	$376, 95, 133(12)$	9.82(0.20)
GenAttack	$14 \times 14 \times 3$	75%	$971, 251, 321(21)$	7.35(0.07)
GP-BO	$16 \times 16 \times 3$	83%	$785, 280, 292(22)$	5.34(0.14)
ADDGP-BO	$16 \times 16 \times 3$	87%	$878, 168, 229(29)$	5.76(0.09)
AutoZOOM	$16 \times 16 \times 3$	3%	$298, 170, 170(90)$	7.50(0.14)
GenAttack	$16 \times 16 \times 3$	79%	$986, 281, 339(36)$	7.36(0.16)

decomposition learning procedure is repeated every 40 BO iterations and we use $M = 12$ for CIFAR10. We denote the method as ADDGP-BO-LD in this section but as ADDGP-BO in the rest of the paper.

We also experiment with another alternative way to learn the decomposition (ADDGP-BO-FD), which is similar to importance sampling in ZOO. We group the pixels/dimensions together if the magnitude of average change in their pixel values over the past 5 BO iterations are close (i.e. pixels that are subject to the largest adversarial perturbations are grouped together). Again, we divide the dimensions into disjoint 12 groups as above to ensure fair comparison.

We compare these two types of decomposition learning methods using 20 randomly selected CIFAR10 images, each of which is attacked on 9 other classes except its original class and a query budget of $1,000$. The results are shown in Table 5.6. It is evident that learning decomposition by maximising marginal likelihood (ADDGP-BO-LD) can achieve higher attack success rate than pixel-value-change-based decomposition learning (ADDGP-BO-FD) given the query budget.



(a) GP-BO **(b)** GP-BO-auto-d^r **(c)** ADDGP-BO

Figure 5.5: Attack objective value against BayesOpt iterations (query count) for using various BayesOpt methods to attack one CIFAR10 image of class label 9 (denoted by i). Curves of different colours correspond to the 9 different target labels (denoted by t). Convergence to 0 objective value indicates successful attack. ADDGP-BO and GP-BO-auto-d^r enjoy faster convergence and thus higher attack success rates than simple GP-BO.

Table 5.6: Summary of attack results on CIFAR10 under a query budget of $1,000$ for different decomposition learning method. $d_r = 14 \times 14 \times 3$ which is further decomposed into 12 subspaces of $d_s = 49$. Q denotes the query count. *ASR* denotes attack success rate. The standard deviations are in parentheses.

Attack method	*ASR*	*Q (Max, Median, Mean)*	Average L_2 perturbation
ADDGP-BO-LD	94%	$885, 134, 212(16)$	$5.78 \times 10^{-4}(5.32 \times 10^{-6})$
ADDGP-BO-FD	87%	$899, 143, 196(14)$	$5.05 \times 10^{-4}(4.52 \times 10^{-6})$

5.6 Discussion

In this chapter, we introduced BayesOpt attack, a new black-box adversarial attack which leverages Bayesian optimisation to find successful adversarial perturbations with high query efficiency. We also suggest two ways in which to improve BayesOpt attack. Firstly, by adopting an additive surrogate structure to ease the optimisation challenge over the typically high-dimensional task. Secondly, by taking full advantage of our probabilistic surrogate model and the available query data to learn the optimal degree of dimension reduction for the search space via Bayesian model selection. In comparison to several existing black-box attack methods, the resulting BayesOpt attacks can achieve high success rates with 2-5 times fewer queries while still

producing adversarial examples that are closer to the original image (in terms of average L_2 distance). We believe our BayesOpt attacks can be a competitive alternative for accessing model robustness against black-box attacks.

One limitation of our attack methods is that due to the poor scalability of GPs (see Section 2.1.3), the attack algorithms are computationally more expensive than most existing alternatives especially as the query number increases. Thus, from an attacking point of view, our BayesOpt attacks are optimal for the setting where the cost of evaluating the target model, being it the monetary costs, computational costs or the risk of being detected, is much higher than the computational cost of the attack algorithm itself and thus query efficiency is highly prioritised. On the other hand, replacing the GP surrogates with more scalable models such as sparse GPs or Bayesian neural networks can be an interesting future research direction. Another interesting future research direction would be to conduct an analysis that is similar to Chapter 4 for black-box attacks using BayesOpt attack and investigate how robust exactly or well approximated Bayesian models are against black-box attacks in comparison to the deterministic benchmark models analysed in this chapter.

6 Conclusions and Future Work

6.1 Conclusions

We set out to shed more light on the adversarial robustness of Bayesian machine learning models, as we find this to be a necessity for their deployment in safety critical applications. We did so in three parts.

In the first part, we focussed on Gaussian process classification (GPC) models. With our approach, we made it possible to assess the safety of a GPC model in terms of its adversarial robustness on the basis of formal guarantees that can be used for safety verification, comparison of robustness profiles between different GPC models as well as interpretability analyses. In experiments, we found that using formal (worst-case) bounds as derived by us can indeed be necessary when assessing adversarial safety, as an empirical analysis based on attacks can lead to a false sense of security. When comparing adversarial robustness profiles, we interestingly found evidence that a better GPC posterior approximation correlates with higher robustness. Lastly, we found interpretability analyses based on our guarantees to be beneficial and complementary to state-of-the-art interpretability methods when non-linearities are present in data and model.

In the second part, we turned to Bayesian neural networks (BNNs). Contrary to previous findings and hypotheses, we showed that ensembling and averaging/marginalising does *not* automatically lead to higher adversarial robustness but can in fact even lead to worse adversarial robustness, especially if it is done on the basis of a high-quality posterior approximation such as obtained by HMC. Furthermore, we found in our experiments, that a large prior variance leading to a larger posterior variance can hurt adversarial robustness in particular. Our findings in this part show that caution is necessary when deploying BNNs in safety critical applications and that deriving reliable adversarial robustness guarantees for BNNs, such as those developed in Wicker et al. (2020) and Cardelli et al. (2019a) remain crucial to obtain safety.

In the third part, we branched out to all (Bayesian and non-Bayesian) machine learning models for contributions w.r.t. a better understanding of their adversarial robustness on a more practical level. Specifically, we improved the power of *practical* adversarial robustness assessments by means of developing a more query efficient black-box attack which needs 2-5 times fewer queries of the target model than previous methods to find adversarial examples. Since in many realistic attack scenarios, attackers might be restricted to use limited query black-box attacks, the resulting lower robustness against black-box attacks of target models established by our approach can be more important than the true (worst-case or white-box attack) robustness in practice. Intriguingly, the attack we propose to achieve this reduction in target model queries rests upon a Bayesian machine learning model itself, as our attack is using GP based Bayesian optimisation to find adversarial examples.

Altogether, we hope to have contributed to a better understanding of the adversarial robustness of Bayesian machine learning models with these three parts of our thesis. The ultimate goal of machine learning models that can support humans in safety critical tasks without any reservations is still distant, but we believe the gap to be closing.

6.2 Future Work

Naturally, there are a lot of questions and problems around the adversarial robustness of Bayesian machine learning models that we could not answer in this thesis, and that have to be left to future work as a consequence. Some of this future work could be seen as direct extensions to our work. At the same time, there are also new directions in relation to our work that we would find interesting to investigate.

6.2.1 Extensions of our Work

Firstly, there are a few extensions to the branch-and-bound approach we proposed in Chapter 3. In particular, at the moment, one of the main limitations of our method is its computational complexity w.r.t. the input dimensionality, as it limits its applications to $10 - 20$ dimensions for binary problems. While GPC models might in general not be suitable for very high-dimensional problems due to the curse of dimensionality, making our approach scale to at least a few hundred dimensions would be desirable. Sticking to the branch-and-bound framework proposed, this could potentially be achieved by further work on tighter outer bounds (as for example in Section 3.8), or by tighter inner bounds. One particular idea for tighter inner bounds, whose effect on convergence would be interesting to investigate, is to use the GPFS attack to propose a further candidate point for the best inner bound value of a given sub-region.

Secondly, the investigation of the robustness of well approximated BNNs should be extended to different data sets (e.g. CIFAR10 or ImageNet) and bigger models. Unfortunately, this is still problematic for HMC methods. For example, we already found it difficult to get finite time convergence for the three hidden layer neural network we used, and for CIFAR10 the best reported (clean) test accuracy we found for an HMC approximated model was 43.4% (Cobb and Jalaian, 2020). Then, despite the fact that Grosse et al. (2018) found that incorporating the predictive uncertainty allows to circumvent uncertainty based detection mechanisms, it would be interesting to see if this indeed holds also for the models and data sets used in

our experiments. Furthermore, as we have seen, the prior distribution can have a big effect on the adversarial robustness of a BNN. It would be interesting to see if certain prior distributions can indeed increase adversarial robustness (e.g. by leading to a posterior distribution which is the equivalent of randomised smoothing). We see the work on functional priors by Sun et al. (2019) as a particularly promising line of work to build such an analysis upon.

Lastly, we see three directions along which our BayesOpt black-box attack introduced in Chapter 5 could be further improved. Firstly, combining the automated dimensionality reduction method with the additive GP prior for the surrogate model could enhance the performance of the latter, as we have seen in Chapter 5 that the dimensionality reduction method does significantly improve the performance of the vanilla GP BayesOpt attack. Secondly, using sparse GP approaches or even BNN models with good posterior approximations could lead to a better scalability of our attack, e.g. for problems such as ImageNet where $> 1,000$ queries are typically needed to find an adversarial example. Lastly, it would be interesting to use BayesOpt attack to investigate how robust well approximated Bayesian models are against black-box attacks in comparison to the deterministic benchmark models we attacked in Chapter 5.

6.2.2 New Directions

Firstly, it would also be interesting to see how adversarial robustness affects other Bayesian models (e.g. those mentioned in Chapter 2) and to see if one can derive some sort of general adversarial robustness results for all Bayesian models.
This leads also to the second new direction that we would find interesting to explore, which is to explore the adversarial robustness of (Bayesian) graph neural networks (Zhang et al., 2019; Sun et al., 2018) and to see if using graph kernels (Ru et al., 2020) can lead to more efficient black-box attacks by means of Bayesian optimisation for graph neural networks as well.

Finally, another interesting direction that we would like to investigate further in a more broad sense would be the links between adversarial robustness and interpretability. The fundamental problem about adversarial examples is that the misclassification caused by them does not resonate with human perception (e.g. a misclassification of a perturbed dog image that to humans still looks like a dog). As Ilyas et al. (2019) demonstrate, building a classifier using only those misclassified adversarial examples does ultimately lead to good test set accuracy on held out unperturbed data, meaning that the data does contain some exploitable generalisable patterns that are however not aligned with human perception. It could be interesting to see if aligning model behaviour with human perception can increase adversarial robustness, and vice-versa. Potentially, this would also include an analysis of robustness against adversarial attacks whose perturbation budgets are restricted in terms of metrics that are more aligned with human perceptions than the commonly used L_p-norms (e.g. rotations, translations, shrinking, embedding, etc.).

Bibliography

Mahdieh Abbasi, Arezoo Rajabi, Christian Gagné, and Rakesh B Bobba. Toward adversarial robustness by diversity in an ensemble of specialized deep neural networks. In *Canadian Conference on Artificial Intelligence*, pages 1–14. Springer, 2020.

Moustafa Alzantot, Yash Sharma, Supriyo Chakraborty, and Mani Srivastava. Genattack: Practical black-box attacks with gradient-free optimization. *arXiv preprint arXiv:1805.11090*, 2018.

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

V Balakrishnan, S Boyd, and S Balemi. Branch and bound algorithm for computing the minimum stability degree of parameter-dependent linear systems. *International Journal of Robust and Nonlinear Control*, 1(4):295–317, 1991.

Andrew Barron, Mark J Schervish, and Larry Wasserman. The consistency of posterior distributions in nonparametric problems. *The Annals of Statistics*, 27(2):536–561, 1999.

Michael Betancourt. A conceptual introduction to Hamiltonian Monte Carlo. *arXiv preprint arXiv:1701.02434*, 2017.

Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer, Cambridge, 2006.

Arno Blaas, Andrea Patane, Luca Laurenti, Luca Cardelli, Marta Kwiatkowska, and Stephen Roberts. Adversarial robustness guarantees for classification with Gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pages 3372–3382. PMLR, 2020.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

Ilija Bogunovic, Jonathan Scarlett, Stefanie Jegelka, and Volkan Cevher. Adversarially robust optimization with Gaussian processes. In *Advances in Neural Information Processing Systems*, pages 5760–5770, 2018.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

Léon Bottou. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer, 2012.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv preprint arXiv:1712.04248*, 2017.

Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

Ginevra Carbone, Matthew Wicker, Luca Laurenti, Andrea Patane, Luca Bortolussi, and Guido Sanguinetti. Robustness of Bayesian neural networks to gradient-based attacks. *arXiv preprint arXiv:2002.04359*, 2020.

Luca Cardelli, Marta Kwiatkowska, Luca Laurenti, Nicola Paoletti, Andrea Patane, and Matthew Wicker. Statistical guarantees for the robustness of Bayesian neural networks. *arXiv preprint arXiv:1903.01980*, 2019a.

Luca Cardelli, Marta Kwiatkowska, Luca Laurenti, and Andrea Patane. Robustness guarantees for Bayesian inference with Gaussian processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7759–7768, 2019b.

Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14. ACM, 2017a.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, 2017b.

Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. Hidden voice commands. In *25th Security Symposium (Security 16)*, pages 513–530, 2016.

Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 15–26. ACM, 2017.

Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. Ead: elastic-net attacks to deep neural networks via adversarial examples. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

Hugh A Chipman, Edward I George, Robert E McCulloch, et al. Bart: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266–298, 2010.

Adam D Cobb and Brian Jalaian. Scaling Hamiltonian Monte Carlo inference for Bayesian neural networks with symmetric splitting. *arXiv preprint arXiv:2010.06772*, 2020.

Adam D Cobb, Atılım Güneş Baydin, Andrew Markham, and Stephen J Roberts. Introducing an explicit symplectic integration scheme for Riemannian manifold Hamiltonian Monte Carlo. *arXiv preprint arXiv:1910.06243*, 2019.

Jeremy M Cohen, Elan Rosenfeld, and J Zico Kolter. Certified adversarial robustness via randomized smoothing. *arXiv preprint arXiv:1902.02918*, 2019.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. *arXiv preprint arXiv:2003.01690*, 2020.

Francesco Croce, Maksym Andriushchenko, and Matthias Hein. Provable robustness of ReLU networks via maximization of linear regions. In *the 22nd International Conference on Artificial Intelligence and Statistics*, pages 2057–2066. PMLR, 2019.

M de Laplace. Oeuvres complètes, vol. 7. *Paris: reedite par Gauthier-Villars*, 1886.

John S Denker and Yann Lecun. Transforming neural-net output levels to probability distributions. In *Advances in neural information processing systems*, pages 853–859, 1991.

Dimitrios I Diochnos, Saeed Mahloujifar, and Mohammad Mahmoody. Adversarial risk and robustness: General definitions and implications for the uniform distribution. *arXiv preprint arXiv:1810.12272*, 2018.

Joseph L Doob. Application of the theory of martingales. *Le calcul des probabilites et ses applications*, pages 23–27, 1949.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

David K Duvenaud, Hannes Nickisch, and Carl E Rasmussen. Additive Gaussian processes. In *Advances in neural information processing systems*, pages 226–234, 2011.

Krishnamurthy Dj Dvijotham, Jamie Hayes, Borja Balle, Zico Kolter, Chongli Qin, Andras Gyorgy, Kai Xiao, Sven Gowal, and Pushmeet Kohli. A framework for robustness certification of smoothed classifiers using f-divergences. In *International Conference on Learning Representations*, 2019.

Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017.

Samuel G Finlayson, John D Bowers, Joichi Ito, Jonathan L Zittrain, Andrew L Beam, and Isaac S Kohane. Adversarial attacks on medical machine learning. *Science*, 363 (6433):1287–1289, 2019.

Adrià Garriga-Alonso, Carl Edward Rasmussen, and Laurence Aitchison. Deep convolutional networks as shallow Gaussian processes. *arXiv preprint arXiv:1808.05587*, 2018.

Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. CRC press, 2013.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*, volume 1. MIT press Cambridge, 2016.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*, 2018.

Kathrin Grosse, David Pfaff, Michael T Smith, and Michael Backes. The limitations of model uncertainty in adversarial settings. *arXiv preprint arXiv:1812.02606*, 2018.

Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*, 2014.

Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens Van Der Maaten. Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117*, 2017.

László Györfi, Michael Kohler, Adam Krzyzak, and Harro Walk. *A distribution-free theory of nonparametric regression*. Springer Science & Business Media, 2006.

W Keith Hastings. Monte Carlo sampling methods using Markov chains and their applications. 1970.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

JB Heaton, NG Polson, and Jan Hendrik Witte. Deep learning for finance: deep portfolios. *Applied Stochastic Models in Business and Industry*, 33(1):3–12, 2017.

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

Daniel Hernández-Lobato, José M Hernández-Lobato, and Pierre Dupont. Robust multi-class Gaussian process classification. In *Advances in neural information processing systems*, pages 280–288, 2011.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *IEEE transactions on Neural Networks*, 13(2):415–425, 2002.

Zilong Hu, Jinshan Tang, Ziming Wang, Kai Zhang, Ling Zhang, and Qingling Sun. Deep learning for image-based cancer detection and diagnosis- a survey. *Pattern Recognition*, 83:134–149, 2018.

Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. *arXiv preprint arXiv:1804.08598*, 2018.

Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. *arXiv preprint arXiv:1905.02175*, 2019.

Pavel Izmailov, Patrick Nicholson, Sanae Lotfi, and Andrew Gordon Wilson. Dangers of Bayesian model averaging under covariate shift. *arXiv preprint arXiv:2106.11905*, 2021a.

Pavel Izmailov, Sharad Vikram, Matthew D Hoffman, and Andrew Gordon Wilson. What are Bayesian neural network posteriors really like? *arXiv preprint arXiv:2104.14421*, 2021b.

Kirthevasan Kandasamy, Jeff Schneider, and Barnabás Póczos. High dimensional Bayesian optimisation and bandits via additive models. In *International Conference on Machine Learning*, pages 295–304, 2015.

Hyun-Chul Kim and Zoubin Ghahramani. Bayesian Gaussian process classification with the EM-EP algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):1948–1959, 2006.

Hyun-Chul Kim and Zoubin Ghahramani. Outlier robust Gaussian process classification. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 896–905. Springer, 2008.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

Anders Krogh and John A Hertz. A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957, 1992.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

Malte Kuss, Carl Edward Rasmussen, and Ralf Herbrich. Assessing approximate inference for binary Gaussian process classification. *Journal of machine learning research*, 6(10), 2005.

Lambda. ICML 2020 abstract submissions by primary subject, 2020. URL https://twitter.com/LambdaAPI/status/1225260075205349376/photo/2.

Yann LeCun. The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.

Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as Gaussian processes. *arXiv preprint arXiv:1711.00165*, 2017.

Xuanqing Liu, Yao Li, Chongruo Wu, and Cho-Jui Hsieh. Adv-bnn: Improved adversarial defense through robust Bayesian neural network. *arXiv preprint arXiv:1810.01279*, 2018a.

Yi Liu, Jie Ling, Zhusong Liu, Jian Shen, and Chongzhi Gao. Finger vein secure biometric template generation based on deep learning. *Soft Computing*, 22(7): 2257–2265, 2018b.

David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

David JC MacKay. A practical Bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Rowan McAllister, Yarin Gal, Alex Kendall, Mark Van Der Wilk, Amar Shah, Roberto Cipolla, and Adrian Vivian Weller. Concrete problems for autonomous vehicle safety: Advantages of Bayesian deep learning. International Joint Conferences on Artificial Intelligence, Inc., 2017.

Thomas P Minka. Expectation propagation for approximate Bayesian inference. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 362–369. Morgan Kaufmann Publishers Inc., 2001.

Aryan Mobiny, Aditi Singh, and Hien Van Nguyen. Risk-aware machine learning classifier for skin lesion diagnosis. *Journal of clinical medicine*, 8(8):1241, 2019.

Seungyong Moon, Gaon An, and Hyun Oh Song. Parsimonious black-box adversarial attacks via efficient combinatorial optimization. *arXiv preprint arXiv:1905.06635*, 2019.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016.

Radford M Neal. *Bayesian Learning For Neural Networks*. PhD thesis, Citeseer, 1995.

Radford M Neal et al. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11):2, 2011.

Christopher Nemeth and Paul Fearnhead. Stochastic gradient Markov chain Monte Carlo. *Journal of the American Statistical Association*, pages 1–18, 2020.

Arnold Neumaier. Complete search in continuous global optimization and constraint satisfaction. *Acta numerica*, 13:271–369, 2004.

Andre T Nguyen and Edward Raff. Adversarial attacks, regression, and numerical stability regularization. *arXiv preprint arXiv:1812.02885*, 2018.

NHTSA. Tesla crash preliminary report. Technical report, National Highway Traffic Safety Administration, Jan. 2017.

Hannes Nickisch and Carl Edward Rasmussen. Approximations for binary Gaussian process classification. *Journal of Machine Learning Research*, 9(Oct):2035–2078, 2008.

Roman Novak, Lechao Xiao, Jaehoon Lee, Yasaman Bahri, Greg Yang, Jiri Hron, Daniel A Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Bayesian deep convolutional networks with many channels are Gaussian processes. *arXiv preprint arXiv:1810.05148*, 2018.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519. ACM, 2017.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037, 2019.

Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018.

Rafael Pinot, Florian Yger, Cédric Gouy-Pailler, and Jamal Atif. A unified view on differential privacy and robustness to adversarial examples. *arXiv preprint arXiv:1906.07982*, 2019.

Carl Edward Rasmussen and Christopher Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should I trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144. ACM, 2016.

Irina Rish et al. An empirical study of the naive Bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46, 2001.

J Ben Rosen and Panos M Pardalos. Global minimization of large-scale constrained concave quadratic problems by separable programming. *Mathematical Programming*, 34 (2):163–174, 1986.

Binxin Ru, Adam Cobb, Arno Blaas, and Yarin Gal. Bayesopt adversarial attack. In *International Conference on Learning Representations*, 2019.

Binxin Ru, Xingchen Wan, Xiaowen Dong, and Michael Osborne. Neural architecture search using Bayesian optimisation with Weisfeiler-Lehman kernel. *arXiv preprint arXiv:2006.07556*, 2020.

Wenjie Ruan, Xiaowei Huang, and Marta Kwiatkowska. Reachability analysis of deep neural networks with provable guarantees. *arXiv preprint arXiv:1805.02242*, 2018.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Lawrence K Saul, Tommi Jaakkola, and Michael I Jordan. Mean field theory for sigmoid belief networks. *Journal of artificial intelligence research*, 4:61–76, 1996.

John Shawe-Taylor, Nello Cristianini, et al. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Lewis Smith and Yarin Gal. Understanding measures of uncertainty for adversarial example detection. *arXiv preprint arXiv:1803.08533*, 2018.

Michael Thomas Smith, Kathrin Grosse, Michael Backes, and Mauricio A Alvarez. Adversarial vulnerability bounds for Gaussian process classification. *arXiv preprint arXiv:1909.08864*, 2019.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

Lichao Sun, Yingtong Dou, Carl Yang, Ji Wang, Philip S Yu, and Bo Li. Adversarial attack and defense on graph data: A survey. *arXiv preprint arXiv:1812.10528*, 2018.

Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger Grosse. Functional variational Bayesian neural networks. *arXiv preprint arXiv:1903.05779*, 2019.

Fnu Suya, Yuan Tian, David Evans, and Paolo Papotti. Query-limited black-box attacks to classifiers. *NIPS Workshop*, 2017.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Tom Taulli. Google and Microsoft announce game-changing developments in AI, 2019. URL https://www.forbes.com/sites/tomtaulli/2019/05/12/the-weeks-important-ai-announcements-from-google-and-microsoft/?sh=6cf16d5e6db9.

Yee Whye Teh. Dirichlet process., 2010.

Tencent Keen Security Lab. Experimental security research of Tesla autopilot, 2019. URL https://keenlab.tencent.com/en/whitepapers/Experimental_Security_Research_of_Tesla_Autopilot.pdf.

Michalis Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *Artificial Intelligence and Statistics*, pages 567–574, 2009.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*, 2018.

Chun-Chen Tu, Paishun Ting, Pin-Yu Chen, Sijia Liu, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh, and Shin-Ming Cheng. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. *arXiv preprint arXiv:1805.11770*, 2018.

Alfredo Vellido. The importance of interpretability and visualization in machine learning for applications in medicine and health care. *Neural computing and applications*, pages 1–15, 2019.

Bao Wang, Zuoqiang Shi, and Stanley Osher. Resnets ensemble via the Feynman-Kac formalism to improve natural and robust accuracies. In *Advances in Neural Information Processing Systems*, pages 1657–1667, 2019.

David Warde-Farley and Ian Goodfellow. 11 adversarial perturbations of deep neural networks. *Perturbations, Optimization, and Statistics*, 311, 2016.

Florian Wenzel, Kevin Roth, Bastiaan S Veeling, Jakub Świątkowski, Linh Tran, Stephan Mandt, Jasper Snoek, Tim Salimans, Rodolphe Jenatton, and Sebastian Nowozin. How good is the bayes posterior in deep neural networks really? *arXiv preprint arXiv:2002.02405*, 2020.

Matthew Wicker, Luca Laurenti, Andrea Patane, and Marta Kwiatkowska. Probabilistic safety for Bayesian neural networks. *arXiv preprint arXiv:2004.10281*, 2020.

Matthew Wicker, Luca Laurenti, Andrea Patane, Zhuotong Chen, Zheng Zhang, and Marta Kwiatkowska. Bayesian inference with certifiable adversarial robustness. In *International Conference on Artificial Intelligence and Statistics*, pages 2431–2439. PMLR, 2021.

Christopher KI Williams and David Barber. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12): 1342–1351, 1998.

Andrew Gordon Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. *arXiv preprint arXiv:2002.08791*, 2020.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Han Xu, Yao Ma, Hao-Chen Liu, Debayan Deb, Hui Liu, Ji-Liang Tang, and Anil K Jain. Adversarial attacks and defenses in images, graphs and text: A review. *International Journal of Automation and Computing*, 17(2):151–178, 2020.

Yuzhe Yang, Guo Zhang, Dina Katabi, and Zhi Xu. Me-net: Towards effective adversarial robustness with matrix estimation. *arXiv preprint arXiv:1905.11971*, 2019.

Matthew Yuan, Matthew Wicker, and Luca Laurenti. Gradient-free adversarial attacks for Bayesian neural networks. *arXiv preprint arXiv:2012.12640*, 2020.

Yingxue Zhang, Soumyasundar Pal, Mark Coates, and Deniz Ustebay. Bayesian graph convolutional neural networks for semi-supervised classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5829–5836, 2019.

Pu Zhao, Sijia Liu, Pin-Yu Chen, Nghia Hoang, Kaidi Xu, Bhavya Kailkhura, and Xue Lin. On the design of black-box adversarial examples by leveraging gradient-free optimization and operator splitting method. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 121–130, 2019.

Roland S Zimmermann. Comment on" Adv-bnn: Improved adversarial defense through robust Bayesian neural network". *arXiv preprint arXiv:1907.00895*, 2019.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.