University of Brasília
Institute of Exact Sciences
Department of Statistics

Master’s Dissertation

Multiobjective Bayesian Optimization To Enhance Computational Efficiency In Neural Network Models

by

João Gabriel Rodrigues Reis

Brasília, 31 june of 2024

Multiobjective Bayesian Optimization To Enhance Computational Efficiency In Neural Network Models

by

João Gabriel Rodrigues Reis

Dissertation submitted to the Department of Statistics at the University of Brasília in fulfilment of the requirements for obtaining the Master Degree in Statistics.

Advisor: Prof. Dr. Guilherme Souza Rodrigues

Brasília, June 2024

Approved by:

Simplicity is the ultimate sophistication.

(Leonardo da Vinci)

To my mother.

Acknowledgments

The conclusion of this dissertation is not just an individual achievement, but the result of the support and collaboration of many people and institutions, to whom I express my deepest gratitude.

Firstly, I thank my advisor, Prof. Dr. Guilherme Souza Rodrigues, for his precise guidance, patience, and continuous encouragement throughout this project. His expertise and dedication were fundamental to the development of this work.

To my family, for their love, understanding, and unwavering support. You were my source of inspiration and motivation during the most challenging moments.

To my friends, especially Aitcheou Gauthier and Rodrigo Barroso, for always being willing to help when I needed it most. Your encouragement and moments of relaxation were essential for maintaining my balance during this journey. I also thank other friends who, directly or indirectly, contributed with their presence and support.

To the University of Brasília, for the excellent academic environment and for providing the necessary resources for the realization of this research.

Finally, I thank all the professors and staff of the institution, whose dedication and commitment to education are inspiring. And to all who, directly or indirectly, contributed to the completion of this work, my sincere thanks.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Resumo Expandido

Otimização Bayesiana Multiobjetivo para Aperfeiçoar a Eficiência Computacional em Modelos de Redes Neurais

Modelos de aprendizado de máquina, especialmente Redes Neurais Artificiais (RNAs), tornaram-se ferramentas indispensáveis em diversas áreas devido à sua capacidade de aprender com dados e fazer previsões ou tomar decisões. A grande flexibilidade dos modelos de RNA torna a escolha dos hiperpâmetros crucial para a obtenção do desempenho ideal. Entretanto, encontrar essa configuração pode ser uma tarefa desafiadora e computacionalmente intensiva.

O ajuste de hiperparâmetros (*hyperparameter tuning*) é crucial para otimizar o desempenho de modelos de aprendizado de máquina. Diversas técnicas são empregadas para essa finalidade, sendo a Otimização Bayesiana (BO) uma das mais prevalentes. Contudo, essa abordagem tradicionalmente foca na maximização da precisão dos modelos, o que frequentemente resulta em modelos desnecessariamente complexos. Esse processo muitas vezes ignora o princípio da parcimônia, também conhecido como a navalha de Occam, que sugere a preferência por soluções mais simples quando desempenhos similares são possíveis.

Na aplicação do princípio de parcimônia em modelos estatísticos clássicos, foram desenvolvidas várias métricas, como o *AIC* (Critério de Informação de Akaike) e o *BIC* (Critério de Informação Bayesiano). Essas métricas avaliam não apenas a precisão do modelo, mas também

o número de parâmetros, buscando um equilíbrio entre complexidade e desempenho. No entanto, sua aplicabilidade é limitada em redes neurais artificiais (RNAs) devido à complexidade destes modelos. As RNAs frequentemente não possuem um conjunto único de pesos ótimos devido à sua alta capacidade de parametrização e às múltiplas soluções locais encontradas durante o treinamento. Essa característica torna desafiador determinar o número efetivo de parâmetros ou os graus de liberdade de uma RNA, o que é crucial para a aplicação do AIC e do BIC, comprometendo a validade dessas métricas para avaliar sua parcimônia. Uma abordagem adotada para contornar essa limitação foi utilizar o custo total de treinamento e avaliação da RNA como um indicativo de sua parcimônia, visando identificar o modelo mais eficiente, ou seja, que tenham alto poder preditivo sem comprometer excessivamente os recursos computacionais.

Normalmente os algoritmos de BO focam em um único objetivo (predições mais acuradas), o que pode resultar em soluções com alto consumo de recursos. Alternativamente, a *Otimização Bayesiana Multi Objetivo* (MOBO) é uma generalização do BO que lida com múltiplos objetivos conflitantes, permitindo uma tunagem de hiperparâmetros que equilibra a precisão do modelo e o custo computacional. Este estudo investiga a eficácia da MOBO na redução dos custos computacionais totais, mantendo ao mesmo tempo a alta acurácia dos modelos, através de simulações que comparam o desempenho da MOBO com métodos tradicionais de BO e busca aleatória.

A BO é uma técnica bastante popular para a tunagem de hiperparâmetros, pois é capaz de encontrar boas configurações com poucas avaliações da função objetivo. Isso é especialmente útil quando a avaliação da função é computacionalmente cara. A BO utiliza processos gaussianos para modelar a função objetivo, permitindo uma estimativa precisa das regiões promissoras no espaço de hiperparâmetros. Através do uso de funções de aquisição, a BO equilibra automaticamente a exploração de novas áreas do espaço de hiperparâmetros e a exploração de áreas já conhecidas que parecem promissoras. Isso ajuda a guiar a busca de maneira inteligente.

Utilizando processos gaussianos e funções de aquisição adaptadas, a MOBO pode iden-

tificar soluções que oferecem um compromisso eficiente entre diferentes métricas de desempenho. Isso é particularmente vantajoso em cenários onde é necessário minimizar o consumo de recursos.

Utilizando o HPOBench, uma plataforma que proporciona uma ampla gama de benchmarks específicos para otimização de hiperparâmetros, este estudo avalia a eficácia da tunagem de hiperparâmetros ao considerar simultaneamente duas funções-objetivo: acurácia e custo. Por meio de simulações, o desempenho de diversas implementações MOBO, BO e métodos de busca aleatória foram comparados. Os resultados obtidos demonstraram que o MOBO foi capaz de gerar modelos significativamente mais eficientes, reduzindo bastante o custo computacional sem sacrificar a precisão.

Abstract

The optimization of hyperparameters is a crucial step in enhancing the performance of machine learning models, particularly Artificial Neural Networks (ANNs). This dissertation explores the application of Multi-Objective Bayesian Optimization (MOBO) to improve computational efficiency in neural network predictions. Traditional Bayesian Optimization (BO) focuses on a single objective, often resulting in resource-intensive solutions. MOBO, however, addresses multiple conflicting objectives, allowing for a balanced trade-off between model accuracy and computational cost.

In this study, we conducted simulations using a benchmark framework HPOBench (Schneider et al., 2021) to compare the performance of MOBO with traditional BO and random search algorithms. The results demonstrate that MOBO significantly reduces total computational cost while maintaining high model accuracy.

Keywords: MOBO, Bayesian Optimization, Multi-Objective Optimization, Hyperparameter Tuning, Neural Networks

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Machine learning models have proven to be highly valuable and versatile tools in a wide array of disciplines and industries. Their ability to learn from data and make predictions or decisions has made them indispensable in addressing complex problems and driving innovation. One prominent characteristic shared by these models is their inherent flexibility, which necessitates researchers to define a set of parameters known as hyperparameters prior to model training.

Selecting the best hyperparameters ensures that the model is better equipped to solve a particular problem. In the context of Artificial Neural Networks (ANN), some hyperparameters that can be defined by the researcher include the number of layers, the number of neurons in each layer, the learning rate, activation functions, among others. Unfortunately, “there is no simple and clear-cut method for determination of these parameters. Guidelines are either heuristic or based on simulations derived from limited experiments. Therefore, the design of an ANN is more of an art than a science” (Zhang, Eddy Patuwo, and Y. Hu, 1998).

Nonetheless, there are many approaches for searching the best combinations of hyperparameters within a space defined by the researcher. This process is known as hyperparameter tuning, and it involves employing various techniques to optimize model performance. Indeed, Bayesian optimization is one of the most renowned algorithms for hyperparameter tuning. It operates by iteratively training the model and assessing an objective function, typically the loss function, over the so-called validation set. This makes hyperparameter optimization a challenging task since it frequently demands a significant amount of time for both training and evaluation. Therefore, the efficiency of hyperparameter tuning lies in obtaining the most accurate model by training as few models as possible. This is crucial since evaluating all combinations of hyperparameters grows exponentially, making it impractical due to the limited amount of time and computational resources available.

A fundamental principle of the scientific method, known as Occam's Razor or principle of parsimony, states that the simplest and most concise explanation for a phenomenon or problem tends to be the most likely and preferable among various hypotheses. In statistical context, this principle is fundamental for model selection, where the chosen model is the simplest capable of explaining the relationship between variables. For this purpose, several measures have been developed to compare different models. Akaike (1974), proposes the AIC method as "a mathematical formulation of the principle of parsimony in the field of model construction", which take into account the number of parameters and the precision of the model, balancing the two things. However, these measures do not work properly for neural networks. According to Curry and Morgan (2006), the absence of unique best-fitting weights in ANNs raises concerns when using these metrics, as it is unclear how one should compute *degrees of freedom* or "the effective number of parameters" (Moody, 1992).

Therefore, selecting more parsimonious neural network models presents a substantial challenge, extending beyond penalization based on the number of network parameters. Some hyperparameters, such as the activation function, greatly contribute to complexity and cannot be directly quantified by simply counting parameters. Thus, to apply Occam's razor principle, other model characteristics beyond the number of neurons will be taken into account. The focus shifts towards models that are more efficient in terms of training time and predicting time. Among the numerous implications of achieving these more efficient models, it is worth highlighting the reduction in both energy consumption and memory usage. This can have significant impacts as the use of ANNs advances.

Despite the fact that classical metrics for goodness of fit do not work well, there are several techniques applied during the training step that help to obtain efficient models for ANNs. Regularization, feature selection and early stopping are some example of methods to avoid overfitting and, in a way, may contribute to obtaining more parsimonious models. However, none of these techniques directly address the issue of training and evaluation time.

A simple approach to obtain simpler models is known as *knowledge distillation* (Hinton and Dean, 2015). This technique involves first training a complex model, known as the teacher model, and then attempting to replicate its behavior or predictions in a smaller model called the student model. The student model is designed to have reduced computational and memory requirements while still maintaining the essential knowledge acquired from the teacher model. Knowledge distillation is an interesting technique for achieving parsimonious models. However, it does require first training an intermediary model (teacher model).

Another aproach to simplify models is called *Quantization* (Jacob et al., 2017), a technique that reduces the precision or bit-width of the weights and activations in a neural network, aiming to make the model more efficient for deployment in resource-constrained environments.

Optimizers, as they do not penalize overly complex solutions, tend to produce models that are not necessarily efficient. The main goal of this study is to evaluate the impact of introducing additional objective functions, such as training and evaluation costs, using Multi-Objective Bayesian Optimization (MOBO), specifically in the context of hyperparameter tuning. MOBO is an extension of BO designed to handle scenarios with multiple conflicting objectives or criteria. This idea was first explored by Snoek, Larochelle, and Adams (2012), who utilized a Gaussian Process variant of multi-task learning to fine-tune the "expected improvement per second" within the Bayesian Optimization (BO) framework, aiming to finding good settings of hyperparameters of Neural Networks as quickly as possible, prioritizing points that are not only likely to yield good results but also can be evaluated quickly.

Using a similar idea, this study investigates the effectiveness of Multi-Objective Bayesian Optimization (MOBO) for hyperparameter tuning using the total cost of the model (train cost + validation cost + test cost) as the second objective. The concept of incorporating multi-objective optimization has previously been explored by esteemed authors, such as Horn and Bischl (2016), who suggest a multi-objective optimization of hyperparameters, taking into account false negative rate and false positive rate in classification models. Similarly, Braga et al. (2006) propose a regularization by minimizing both squared error and the norm of the weight vectors.

The following chapters of the present study are organized as follows: Chapter 2 provides a background on hyperparameter tuning, presenting the main techniques used in this work. Chapter 3 describes the proposed research methodology for ANN search models with more parsimonious architecture and some results of simulations with different algorithms. Chapter 4 provides a brief discussion and future work.

To ensure the reproducibility of this research and to provide access to the code and data used throughout the study, all relevant materials have been made available in a public GitHub repository. The repository includes the scripts for the simulations, the datasets, and the analysis tools developed. Interested readers can access the repository at `https://github.com/j-gabriel-reis/projeto-mestrado`, where they will find detailed instructions for reproducing the experiments and further exploring the methods presented in this dissertation.

Chapter 2

Background

2.1 Single-Objective Optimization

Optimization is a key area of applied mathematics with broad applications across various fields such as economics, computing and logistics. In single-objective optimization, the primary aim is to identify the “best solution” that optimizes a specific criterion, often subject to certain constraints. This involves minimizing or maximizing a single objective function to find the most efficient or cost-effective solution. A practical example can be seen in the automobile industry, where engineers strive to design a car that minimizes fuel consumption by strategically selecting and balancing variables such as engine efficiency, aerodynamics, car weight, and tire resistance. The design must also comply with safety standards, cost limitations and performance requirements. Through a meticulous process of design parameter selection, simulation and modeling, prototype testing and iterative improvements, manufacturers can optimize a car’s design to reduce fuel consumption effectively.

As shown by Takahashi (2004), the optimization problem can be expressed as

$$\boldsymbol{\theta}^{opt} = \text{arg min}_{\boldsymbol{\theta}} f(\boldsymbol{\theta})$$

subject to

$$\begin{cases} g_i(\boldsymbol{\theta}) \leq 0, & i = 1, \ldots, r \\ h_j(\boldsymbol{\theta}) = 0, & j = 1, \ldots, p \end{cases}$$

given that $\boldsymbol{\theta} \in \mathbb{R}^p$, $f : \mathbb{R}^p \to \mathbb{R}^1$, $g : \mathbb{R}^p \to \mathbb{R}^r$, and $h : \mathbb{R}^p \to \mathbb{R}^q$. Where $g = (g_1, \ldots, g_r)$ and $h = (h_1, \ldots, h_p)$.

In this notation $f(\boldsymbol{\theta})$ is the objective function that will be minimized, $g_i(\boldsymbol{\theta})$ and $h_i(\boldsymbol{\theta})$ are the inequality and equality constraints, respectively.

Since no single global optimization technique exists, the selection of an appropriate method depends on the characteristics of the functions $f(\boldsymbol{\theta})$, $g(\boldsymbol{\theta})$, and $h(\boldsymbol{\theta})$, specifically whether they are linear or non-linear, continuous or discontinuous, and differentiable or non-differentiable.

In particular scenarios where $f(\boldsymbol{\theta})$ is too expensive to evaluate and does not have a closed form, is noisy, and derivative information is unavailable, Bayesian optimization is particularly useful. This technique builds a probabilistic model of the objective function and iteratively updates the model as new data points are evaluated, making it efficient for optimizing under uncertainty and reducing the number of function evaluations needed to find an optimum. Due to these particularities, Bayesian optimization is one of the most widespread techniques for the hyperparameter tuning in Neural Networks, where the search for an optimal configuration can be extremely complex and time-consuming.

2.1.1 Gaussian Process

This section explains Gaussian Processes (GPs), which are crucial for Bayesian Optimization. GP is a valuable probabilistic approach that treats functions as random variables, making it a highly suitable choice for a prior distribution in Bayesian Optimization. As Bayesian models, they not only provide a point estimate but also specify the entire posterior predictive distribution.

They are particularly useful in regression tasks, where the goal is to predict a continuous target variable. They provide not only a predicted mean value at each input point but also a measure of uncertainty in the prediction, represented by the variance. Quantifying this uncertainty is extremely valuable in situations where it is essential to make informed decisions based on reliable predictions.

A GP is a collection of random variables, where any finite subset of these variables has a joint Gaussian distribution. At its core, a GP defines functions over an infinite number of points in the input space. These functions are described by a covariance function, also known as a kernel function, which influences their shape and behavior. The associated distribution of the random variables is characterized by a mean function, referred to as μ, and a covariance function, referred to as $\mathbf{K}$. The model assumes that,

$$f(\Theta) \sim N(\boldsymbol{\mu}(\Theta), \mathbf{K}(\Theta, \Theta)),$$

where $\Theta = [\theta_1, \cdots, \theta_N]^T$ is a vector of N observations of θ, $\boldsymbol{\mu}(\Theta) = [\mu(\theta_1), \ldots, \mu(\theta_N)]^T$ is the prior mean vector and

$$\mathbf{K}(\Theta, \Theta) = \begin{bmatrix} k(\theta_1, \theta_1) & \cdots & k(\theta_1, \theta_N) \\ \vdots & \ddots & \vdots \\ k(\theta_N, \theta_1) & \cdots & k\theta_N, \theta_N) \end{bmatrix}$$

is the prior covariance matrix. For simplicity, it is usually assumed that $\mu(\theta) = \mathbf{0}$, especially when the kernel k effectively captures the data structure. Common kernel functions include the linear kernel for capturing linear relationships, given by

$$k(\theta_i, \theta_j) = \theta_i \theta_j, \quad i, j = 1, \ldots, N$$

and the Radial Basis Function (RBF) kernel, which models smooth functions and can be expressed as

$$k(\theta_i, \theta_j) = \exp\left(\frac{-||\theta_i - \theta_j||^2}{2 * \ell^2}\right).$$

Here, ℓ is a hyperparameter that controls the width or smoothness of the kernel. It determines how quickly the similarity (or correlation) between the data points decreases as they move away from each other. A smaller ℓ value results in a more wiggly or flexible kernel, while a larger ℓ value results in a smoother kernel. When θ_i and θ_j are close together, the Euclidean distance $||\theta_i - \theta_j||^2$ tends to zero and k is close to 1, showing that they strongly affect each other. But when data points are far apart, k gets close to 0, which means they barely affect each other.

One of the key properties of the multivariate normal distribution is that the conditional distribution of a subset of its components, given the other components, is also a multivariate normal distribution. Thus, for a set of new observations $\Theta^* = [\boldsymbol{\theta}_{N+1}, \ldots, \boldsymbol{\theta}_{N+q}]^T$, the predictions $\mathbf{f}^* = [f(\boldsymbol{\theta}_{N+1}), \ldots, f(\boldsymbol{\theta}_{N+q})]^T$ can be obtained according to a known Gaussian distribution. For new observations, by the properties of Gaussian processes, the jointly distribution follows,

$$\begin{bmatrix}\mathbf{f}\\ \mathbf{f}^*\end{bmatrix} \sim N\left(\begin{bmatrix}\boldsymbol{\mu}(\Theta)\\ \boldsymbol{\mu}(\Theta^*)\end{bmatrix}, \begin{bmatrix}\mathbf{K}(\Theta,\Theta) & \mathbf{K}(\Theta,\Theta^*)\\ \mathbf{K}(\Theta^*,\Theta) & \mathbf{K}(\Theta^*,\Theta^*)\end{bmatrix}\right),$$

where $\mathbf{f} = [f(\boldsymbol{\theta}_1), \ldots, f(\boldsymbol{\theta}_N)]^T$.

From this, the conditional distribution of $\mathbf{f}^*$ given the previous observations is derived. The posterior distribution is also a Gaussian distribution and is given by

$$\mathbf{f}^*|\Theta^*, \mathbf{f}, \Theta \sim N\left(\mathbf{m}^*, C^*\right),$$

where the posterior mean is expressed as

$$\mathbf{m}^* = \boldsymbol{\mu}(\Theta^*) + \mathbf{K}(\Theta^*,\Theta)\mathbf{K}(\Theta,\Theta)^{-1}(\mathbf{f} - \boldsymbol{\mu}(\Theta)),$$

and the posterior covariance is denoted by

$$C^* = \mathbf{K}(\Theta^*, \Theta^*) - \mathbf{K}(\Theta^*, \Theta)\mathbf{K}(\Theta, \Theta)^{-1}\mathbf{K}(\Theta^*, \Theta)^T.$$

In practice, GPs start with a prior distribution over functions, encoding initial beliefs about the underlying data. As new data is observed, the GP is updated to provide a posterior distribution that blends these prior beliefs with the observed information. This Bayesian approach allows GPs to adapt to the data and provide probabilistic predictions, making them suitable for various applications.

The choice of kernel functions, in a Gaussian process (GP) regression or modeling task is a critical decision that influences the model's ability to capture and predict the underlying patterns in the data. Different covariance functions capture different types of behavior and structures in the data.

2.1.2 Bayesian Optimization

This section presents a brief introduction to Bayesian optimization techniques. For further reading, see Brochu, Cora, and De Freitas (2010).

Bayesian Optimization is an advanced and remarkably efficient optimization technique utilized to discover extreme values of objective functions that require substantial resources for evaluation. This approach is particularly useful when the objective function does not have a readily available mathematical expression but can be observed through sampling. One of its key strengths lies in its capacity to integrate prior assumptions about the problem, guiding the next sampling step and striking a balance between exploration and exploitation within the search space.

Define $f(\theta)$ as an objective function and $\mathbf{D}_N = \{(\theta_1, f_1), \cdots, (\theta_i, f_i), \cdots, (\theta_N, f_N)\}$, where θ_i, $i = 1, \ldots, N$, represents evaluation points, and f_i represents the observed values. The process starts with a Gaussian Process (GP) as a probabilistic model for the objective function

$f(\theta)$. This means that it is assumed that for any set of points $\theta_1, ..., \theta_n$, $(f(\theta_1), ..., f(\theta_n))$ approximately follows a n-variate Gaussian Distribution. Bayes' Theorem is applied to update the probabilistic model when a new observation f_{N+1}, for example, is observed, serving as a mathematical formula to revise beliefs (probability distribution) about a model (f in this case) in response to new evidence (θ_{N+1}). It is expressed as

$$P(f(\theta)|\mathbf{D}_{N+1}) \propto P(f(\theta_{N+1})|\mathbf{D}_N, \theta_{N+1})P(f(\theta)|\mathbf{D}_N).$$

The posterior distribution of $f(\theta)$ after incorporating observations $\mathbf{D}_{N+1}$ is denoted as $P(f(\theta)|\mathbf{D}_{N+1})$. Another way to perceive this stage is as the process of approximating the objective function through the utilization of a surrogate function, commonly known as a response surface. The likelihood $P(f(\theta_{N+1})|\mathbf{D}_N, \theta_{N+1})$ measures how well the observed data align with the model's predictions . In this framework, $P(f(\theta)|\mathbf{D_N})$, representing the posterior distribution at point $\mathbf{D}_N$, serves as the prior in the subsequent iteration. This means that the posterior distribution obtained after incorporating the observations $\mathbf{D}_N$ becomes the new prior when considering new observations. This iterative process continues throughout each stage of Bayesian optimization, where the current posterior is continuously updated and used as the prior for the next iteration.

With the updated posterior distribution, we can make predictions about the objective function at any point in the search space. This allows us to calculate acquisition strategies to determine where to evaluate the function next. The choice involves an inherent balance between exploration, which occurs in regions where the objective function's behavior is highly uncertain, and exploitation, which involves testing values of θ where the objective function is anticipated to perform well. This optimization method possesses an advantageous characteristic, as it strives to minimize the total number of evaluations of the objective function.

A loop with N iterations of the Bayesian optimization process is available in algorithm 1.

Algorithm 1 Bayesian Optimization

```
1: Initialize the surrogate model with a prior distribution $P(f(\theta))$
2: Specify the acquisition function (e.g., Expected Improvement, Probability of Improvement)
3: Specify the number of iterations $J$
4: Initialize the dataset $D$ with initial observations $(\theta, f(\theta))$
5: for $j = 1$ to $J$ do
6:     Optimize the acquisition function to find the next evaluation point $\theta_j$
7:     Evaluate the objective function at the selected point $f(\theta_j)$
8:     Append $(\theta_j, f(\theta_j))$ to $D$
9:     Update the surrogate model with the new observation
10:    Update the acquisition function with the surrogate model
11: end for
12: Return the best-found solution $\theta_{\text{best}}$ or the optimal hyperparameters
```

Acquisition Function

An acquisition function in the context of Bayesian optimization acts as the compass guiding the search for optimal solutions in complex, high-dimensional spaces.

A popular acquisition function is the *Expected Improvement* (EI). It quantifies the potential improvement in the objective function at a given point compared to the current best result. It is like asking, "At which point is the greatest reduction in the loss function expected, relative to the current best result"? EI guides the search toward promising areas while acknowledging the uncertainty inherent in any optimization process and is given by

$$\text{EI}(\theta^*) = E[\max(\mathbf{m}^* - \mathbf{f}_{\text{best}}, 0)].$$

q-Expected Improvement (qEI) is a variation of Expected Improvement (EI), designed for scenarios where q points can be evaluated simultaneously. It employs a Monte Carlo approach to compute the expected improvement by sampling the joint posterior distribution over a set of points, evaluating the improvement over the current best value for each sample, maximizing this improvement across the evaluated points, and averaging the improvements over all samples. This method promotes the exploration of new areas in the search space and utilizes the possibility of parallel evaluations to accelerate the optimization process. This makes qEI ideal for applications where function evaluation is expensive and can be performed in parallel, enhancing computational efficiency and optimizing the search for the global optimum.

Another approach is the *Probability of Improvement* (PI), which calculates the likelihood that the objective function will exceed a certain threshold at a specific point. It encourages exploration by favoring points with a high probability of improvement.

$$\text{PI}(\theta^*) = P(\mathbf{m}^* \geq \mathbf{f}_{best}).$$

The "Upper Confidence Bound" (UCB) method aims for a balance by selecting points where the upper confidence bound on the objective function is maximized. It introduces a parameter that allows you to adjust the exploration-exploitation trade-off to suit your needs.

$$UCB(\theta^*) = \mathbf{m}^* + \beta\sigma(\theta^*),$$

where $\mathbf{m}^*$ is the mean predicted by surrogate model at θ^*, $\sigma(\theta^*)$ is the standard deviation and β is a hyperparameter that controls the balance between exploration and exploitation.

Ultimately, the choice of the right acquisition function depends on the nature of the problem and your optimization goals. Bayesian optimization algorithms rely on these functions to iteratively steer the search towards optimal solutions in a data-efficient manner, making them invaluable tools in complex optimization tasks.

Figure 2.1 shows an example of four Bayesian optimization loops.

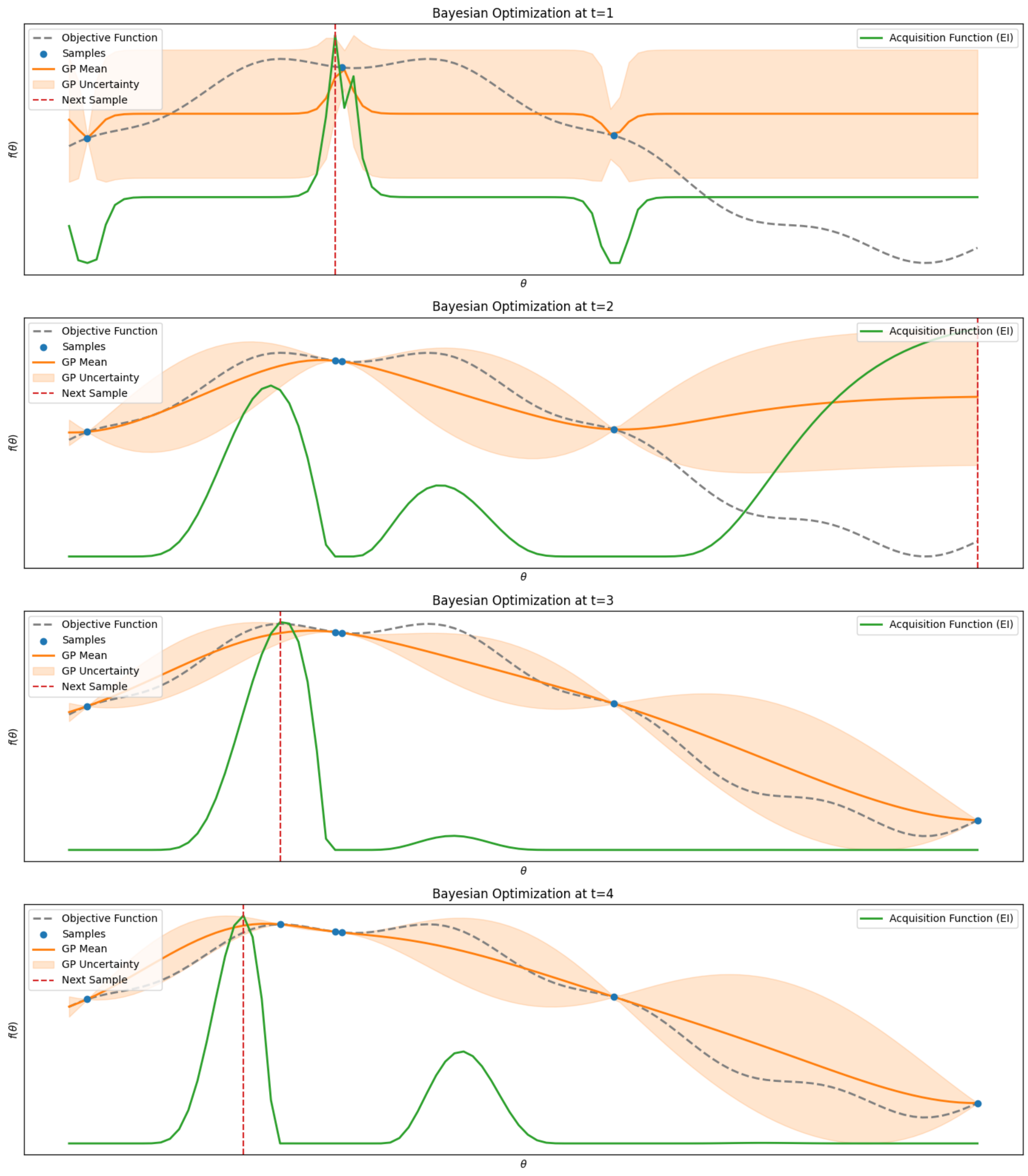


Figure 2.1: **Bayesian Optimization Example**. This example demonstrates the application of Bayesian optimization. The diagrams present a Gaussian process (GP) model approximating the objective function across four iterations where the objective function values are sampled. Additionally, the diagrams include the acquisition function depicted in the plots below. The acquisition values are elevated in regions where the GP forecasts a high objective value (exploitation) and in areas of significant predictive uncertainty (exploration). These characteristics lead to prioritizing sampling in these regions.

2.2 Multi-Objective Optimization

A multi-objective optimization problem involves finding solutions that simultaneously satisfy multiple performance criteria, which are often in conflict with each other. Unlike single-objective optimization, which seeks to maximize or minimize a single function, multi-objective optimization requires balancing two or more objective functions.

Since in problems of this nature there tends to be no optimal solution that simultaneously minimizes all objective functions, the main challenge in this type of optimization is that improving one objective may lead to the deterioration of another, creating a scenario of compromise or trade-off. Therefore, the ideal solution is not unique but rather a set of optimal solutions known as Pareto-optimal or efficient solutions. These solutions represent different degrees of trade-offs between the objectives, without any solution being clearly superior in all respects.

For a clearer illustration, refer to the previously mentioned example from the automotive field. Here, the objective is not only to minimize fuel consumption, but also to reduce the cost of the vehicle. These two goals frequently conflict, as materials and technologies that lower fuel consumption often lead to higher manufacturing costs. Consequently, unlike single-objective scenarios, there is no single optimal solution but a range of options. These solutions are considered non-dominated and collectively form the set of Pareto optimal solutions.

The Pareto solutions provide several alternatives that may converge into a single final solution, but an additional step is required where a decision-maker will choose the definitive solution.

Let $\boldsymbol{\theta} \in \mathbb{R}^p$ be the vector of design parameters for this problem. This vector is restricted to a specific subset $F \subseteq \mathbb{R}^p$, which defines the conditions or constraints that $\boldsymbol{\theta}$ must satisfy. The objective functions of the problem are represented by the vector $\mathbf{f}(\boldsymbol{\theta}) = (f_1(\boldsymbol{\theta}), \dots, f_m(\boldsymbol{\theta}))$, where $\mathbf{f} : \mathbb{R}^p \to \mathbb{R}^m$. The multi-objective problem can be formally described as

$$\theta^{opt} = \text{arg min}_{\boldsymbol{\theta}} \mathbf{f}(\boldsymbol{\theta}),$$

$$\text{subject to}\{\boldsymbol{\theta} \in F\},$$

where $\boldsymbol{\theta}^{opt}$ is the set of Pareto-optimal solutions for the problem, where no solution can be improved without worsening at least one of the other objective functions. Formally, $\boldsymbol{\theta}^{opt}$ can be defined as

$$\boldsymbol{\theta}^{opt} = \{\boldsymbol{\theta} \in F \mid \nexists\tilde{\boldsymbol{\theta}} \in F \text{ such that } \mathrm{f}(\tilde{\boldsymbol{\theta}}) \preceq \mathrm{f}(\boldsymbol{\theta}) \text{ and } \mathrm{f}(\tilde{\boldsymbol{\theta}}) \neq \mathrm{f}(\boldsymbol{\theta})\},$$

$\mathrm{f}(\tilde{\boldsymbol{\theta}} \preceq \mathrm{f}(\boldsymbol{\theta})$ means that $f_i(\tilde{\boldsymbol{\theta}}) <= f_i(\boldsymbol{\theta}),$ for $i = 1, \ldots, m.$ Thus, $\boldsymbol{\theta}$ is a Pareto optimal solution if there is no other solution $\boldsymbol{\theta}_1 \in F$ such that $f(\boldsymbol{\theta}_1) \preceq f(\boldsymbol{\theta})$ and $f(\boldsymbol{\theta}_1) \neq f(\boldsymbol{\theta})$. In other words, $\boldsymbol{\theta}$ is not *dominated* by another feasible solution.

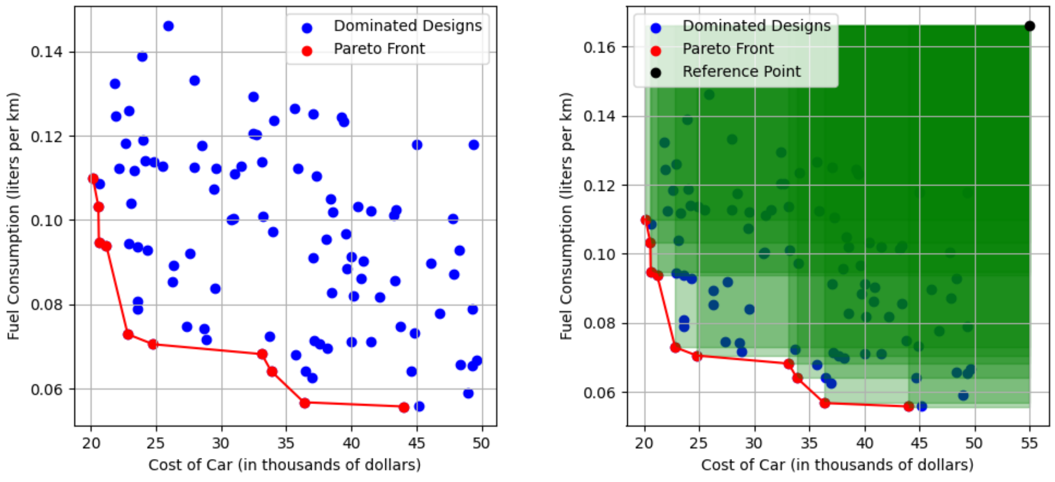
There are several metrics to measure the quality of a multi-objective optimization. One of the most comum metric is *Hypervolume* (HV) (Zitzler and Künzli, 2004).

HV evaluates the volume of the objective space that is dominated by a set of solutions. It is a comprehensive indicator because it accounts for both convergence and diversity of the solutions. Figure 2.2 presents a visualization of the Pareto front and hypervolume in an example.

To calculate the hypervolume, is necessary first to choose a reference point $\mathbf{f}_{ref}$, called the negative utopia point, which is dominated by all solutions in the objective space. Then, for each solution s in the set of solutions S, determine the hypervolume of the region dominated by s and bounded by $\mathbf{f}_{ref}$. The total hypervolume is the sum of the volumes of the regions dominated by all solutions in S. The hypervolume $\text{HV}(S, \mathbf{f}_{ref})$ is expressed as

$$\text{HV}(S, \mathbf{f}_{ref}) = \int_{\mathbb{R}^m} I(f(\boldsymbol{\theta}) \leq \mathbf{f}_{ref})\, d\boldsymbol{\theta},$$

where I is the indicator function that is 1 if $\mathbf{f}(x) \leq \mathbf{f}_{ref}$ and 0. An example of the measure can be viewed in figure 2.2b.



(a) Pareto Front (b) Hypervolume

Figure 2.2: **Multi-Objective Example**. In (a) the plot shows the Pareto Optimal solutions of example given in section 2.2. Image (b) illustrates the area corresponding of hypervolume metric.

2.2.1 Multi-Objective Bayesian Optimization (MOBO)

Multi-objective Bayesian optimization, often abbreviated as MOBO, is a powerful technique used in optimization problems where there are multiple conflicting objectives to be considered simultaneously. It extends the principles of traditional Bayesian optimization to tackle these complex scenarios. The primary objective is to discover and learn the Pareto front, which represents the set of optimal trade-offs.

In the context of MOBO optimization, the desire is to optimize a vector of objective functions $\mathbf{f}(\Theta) = [f_1(\boldsymbol{\theta}), \ldots, f_M(\boldsymbol{\theta})]^T \in \mathbb{R}^m$. Typically, there is no single solution $\boldsymbol{\theta}^{opt}$ that can simultaneously optimize all M objectives while satisfying I constraints of the form $g_i(\boldsymbol{\theta}) \geq 0$.

The optimization process is an extension of the univariate case. However, the necessity of dealing with multiple objectives requires additional techniches to balance conflicting objectives. It is common in MOBO to train individual surrogate models for each objective, and adaptations are made to acquisition functions to accommodate multiple models. An Example of such acquisition functions is the Expected Hypervolume Improvement (EHVI), that measures the expected improvement in the region of non-dominated solutions when adding a new point, helping to identify solutions that enhance the balance between multiple objectives.

A surrogate model will be used capture the relationships between the input parameters and each objective, assuming

$$f_i(\Theta) \sim N(\boldsymbol{\mu}_i(\Theta), \mathbf{K}_i(\Theta, \Theta)),$$

where $\Theta = [\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_N]^T$ is the vector of N samples and $\boldsymbol{\mu}_i(\theta)$ is the mean vector of the function i and $K_i(\Theta, \Theta)$ is the covariance matrix of the process.

Analogous to BO, the process starts with an initial set of solutions and the surrogate model. An acquisition function guides the selection of new input points to evaluate. These evaluations are used to update the surrogate models and identify solutions that dominate others. As the iterations continue, MOBO aims to populate the Pareto front with a diverse set of solutions that represent the best trade-offs between the conflicting objectives. This diversity ensures that decision-makers have a range of options to choose from, depending on their priorities and preferences.

Acquisition Functions

As in the BO scenario, acquisition functions are required in MOBO to efficiently explore the search space and balance the trade-off between exploration and exploitation. These functions use a probabilistic model to predict the objective values and their associated uncertainties, guiding the selection of candidate points for the next evaluation.

Expected Hypervolume Improvement (EHVI) (Zitzler et al., 2003) is a prominent acquisition function that evaluates the expected improvement in the hypervolume of the objective space dominated by a set of solutions. At a candidate point θ^* EHVI can be expressed as

$$\text{EHVI}(\theta^*) = \mathbb{E}\left[\max\left(0, HV(S \cup \{\mathbf{f}(\theta^*)\}, f_{\text{ref}}) - HV(S, f_{ref})\right)\right]$$

where $\text{HV}(S, f_{\text{ref}})$ is the hypervolume of the current Pareto front S, and $\mathbf{f}(\theta^*)$ is the objective vector corresponding to the candidate point θ^*. The HV improvement is calculated using Box Decomposition, where the space not dominated by the Pareto frontier is decomposed and partitioned into a set of axis-aligned hyperrectangles. EHVI evaluates the difference in HV before and after adding the new point. To compute it, a surrogate model (for instance, a Gaussian process) is used to approximate the distribution of objective values at θ^*, followed by numerical integration to determine the expected value.

The q-Expected Hypervolume Improvement (qEHVI) (Daulton, Balandat, and Bakshy, 2020) extends EHVI to consider a batch of q candidate points simultaneously, rather than a single point. In this case, the acquisition function is the joint hypervolume improvement of the multiple points. This batch approach is advantageous in scenarios where multiple evaluations can be performed in parallel, such as in high-performance computing environments. More formally, qEHVI evaluates the expected improvement in hypervolume when a set of q candidate points $\theta_1^*, \theta_2^*, \ldots, \theta_q^*$ is added to the existing Pareto front. It can be expressed as:

$$\text{qEHVI}(\theta_1^*, \theta_2^*, \ldots, \theta_q^*) = \mathbb{E}\left[\max(0, \text{HV}(P \cup \{f(\theta_1^*, \theta_2^*, \ldots, \theta_q^*)\}) - \text{HV}(P))\right],$$

BoTorch (Balandat et al., 2020) is a framework offering a range of algorithm implementations for Multi-Objective Bayesian Optimization MOBO, such as EHVI and qEHVI.

MOBO Loop

The following outlines the Multi-Objective Bayesian Optimization (MOBO) algorithm. This algorithm is designed to efficiently optimize multiple objectives by using surrogate models and acquisition functions. The process involves initializing surrogate models with prior distributions, specifying an acquisition function, and iteratively updating the models based on new observations. The algorithm aims to find the best solutions or optimal hyperparameters through a series of iterations.

Algorithm 2 Multi-Objective Bayesian Optimization

```
 1: Specify the acquisition function (e.g., EHVI, qEHVI).
 2: Specify the number of iterations J.
 3: Specify batch-size $q$ (when necessary).
 4:
 5: for $i = 1, ..., m$ do
 6:     Initialize the surrogate model with a prior distribution $P_i(f(\theta))$ for each objective.
 7: end for
 8: for $j$ = 1, 2, 3, ..., J do
 9:     Optimize the acquisition function to find the next evaluation point.
10:     Evaluate the objective function at the selected point.
11:     Append $(\theta_{n+j}, f_{n+j})$ to $D$.
12:     Update the surrogate model with the new observation.
13:     Update the acquisition function with the surrogate model.
14: end for
15:
16: Return the best-found solution or the optimal hyperparameters.
```

2.3 Comparison Between Tuning Methods

As models continue to advance in complexity and scale, the pursuit of better hyperparameter optimization methods has gained paramount importance. The quest for optimal hyperparameters is an ongoing challenge in machine learning and deep learning, given the profound impact they have on the performance of these models.

To effectively develop, improve, understand and compare methods, it is imperative to have access to benchmark problems that are not only realistic but also efficient and available for an extended duration. HPOBench (Schneider et al., 2021) is a valuable resource, providing a comprehensive collection of reproducible benchmarks accessible through a unified API. The framework enables researchers and practitioners to easily access and work with a diverse set of benchmark problems for the evaluation and comparison of hyperparameter optimization methods. Its API simplifies the process of accessing and utilizing these benchmarks, contributing to more consistent and rigorous research in the domain of hyperparameter optimization.

HPOBench offers a diverse collection of over one hundred containerized multi-fidelity hyperparameter optimization (HPO) benchmark problems. These benchmarks are accessible in both raw and tabular versions and support multi-objective optimization scenarios.

To ensure the utility and relevance of these benchmarks, researchers have conducted a comprehensive large-scale study. This study involved the evaluation of more than ten optimization methods for all benchmark problems. The aim was to validate the effectiveness and practicality of these benchmarks as they relate to real-world HPO challenges.

A benchmark comprises a parameterized algorithm, a dataset for training, and a loss function to evaluate the performance of these algorithms. The benchmark emphasizes reproducibility, flexibility and efficiency.

HPOBench addresses the challenge of reproducibility through the implementation of containerization for benchmarks and their associated dependencies, using Singularity containers. This strategy effectively isolates each benchmark from both the other benchmarks and the host system, streamlining the management of complex software requirements. These containerized benchmarks are then stored in a GitLab registry, guaranteeing their long-term utility without need for additional installation procedures. While this method enhances reproducibility by maintaining benchmarks as containers, users should be mindful of variables such as seed replicates and hardware discrepancies when comparing optimization results.

The benchmarks consist of two groups: 22 preexisting multifidelity benchmarks from seven distinct model families, sourced from existing multifidelity literature, and 88 new benchmarks from five different model families, created to significantly enhance the flexibility of HPOBench.

HPO benchmarks have a significant drawback in that they evaluate a costly function, making it expensive to empirically compare optimization algorithms. This limitation hinders their use for the interactive development of new methods. To address this concern, in addition to raw benchmarks, the framework offer two well-established benchmark categories that mitigate this problem: Tabular Benchmark and surrogate Benchmark.

2.4 Model Compression

2.4.1 Knowledge Distillation

Knowledge Distillation (Hinton and Dean, 2015), is a framework for condensing complex models or ensembles into a single compact model. Effective ANN representations are often associated with deeper architectures characterized by a high number of parameters. However, this approach can lead to slower model predictions and increased memory allocation. To deal with these important limitations, knowledge distillation proposes transferring the knowledge gained by the complex model, often referred to as the teacher model, to a smaller and more efficient model, known as the student model.

The student model learns by optimizing following loss function:

$$\ell_{KD} = \alpha \ell(\mathbf{y}, \hat{y}_S) + \beta \ell(\hat{\mathbf{y}}_\mathbf{T}, \hat{\mathbf{y}}_\mathbf{S}),$$

where ℓ_{KD} is the total loss function of the student model, α and β are weights that control the relative importance of the two components of the loss, $\ell(\mathbf{y}, \hat{y}_S)$ is the traditional loss function calculated based on the predictions of the student model and the true labels and $\hat{\mathbf{y}}_\mathbf{S}$).

2.4.2 Quantization

Another strategy for obtaining more efficient models is Quantization (Jacob et al., 2017), a process of reducing the precision of the number of bits used to represent the weights and activations of a model. It involves representing numerical values with a lower bit precision than the original floating-point representation. This is done to reduce the memory footprint and computational requirements of the model, making it more efficient for deployment on resource-constrained devices.

The process of quantization reduce the precision of a model's weights and activations, typically expressed as 32-bit floating-point numbers, to representations with lower precision, such as 8-bit integers. Initially, a model is trained using conventional techniques, and its weights and activations are represented with high precision. Following performance evaluation, the desired quantization precision, such as 8 bits, is chosen and the weights and activations are converted to this new precision. Methods such as truncation, rounding, or more advanced techniques like clustering are employed in this process. The quantized model may undergo retraining or fine-tuning to offset potential performance losses due to reduced precision. Subsequently, the quantized model is assessed to ensure that the reduction in precision has not significantly compromised performance compared to the original model.

Quantization is particularly relevant in scenarios where memory and computational resources are limited, such as edge devices, mobile phones, and IoT (Internet of Things) devices. It allows for faster inference and reduced model size, making it more practical to deploy deep learning models in real-world applications with constraints.

The success of quantization may depend on the nature of the task and the model architecture, and in some cases, it may be combined with additional strategies such as pruning to further optimize the final model.

Chapter 3

Method and Results

The fundamental goal of this study is to conduct an evaluation of the consequences resulting from the integration of a secondary objective function (cost) into an artificial neural network training process. This additional cost function is directly linked to training and prediction time, with our primary focus being on investigating how this integration can contribute to the development of parsimonious models that are intrinsically more efficient and time-economic.

More efficient models offer several significant benefits, such as reduced computational costs achieved by requiring less power and resources, thereby lowering hardware and energy expenses. With the increasing use of machine learning models, this reduction in resource consumption translates to a smaller environmental footprint, and as the use of AI intensifies, these savings can become even greater, amplifying both economic and environmental benefits.

Additionally, efficient models can lead to faster processing times, enabling quicker training and prediction phases. This speed not only enhances the performance of the applications utilizing these models but also improves user experience by providing rapid results. Moreover, the efficiency of these models allows for better scalability, enabling them to handle larger datasets and more complex problems.

In this work, to achieve more efficient models, we aim to minimize both the standardized loss $L(\boldsymbol{\theta})$ and cost $C(\boldsymbol{\theta})$ of the models. Therefore, the objective function is given by

$$\mathbf{f}(\boldsymbol{\theta}) = (L(\boldsymbol{\theta}), \quad C(\boldsymbol{\theta})),$$

where $L(\boldsymbol{\theta})$ is the complement of accuracy $(1-accuracy)$. In addition to MOBO, univariate BO optimization processes were adjusted by weighting cost and accuracy in a single loss function to evaluate their competence in finding more efficient solutions. The loss is expressed as

$$l(\boldsymbol{\theta}) = \alpha L_{\text{std}}(\boldsymbol{\theta}) - \beta C_{\text{std}}(\boldsymbol{\theta}),$$

where $L_{\text{std}}(\boldsymbol{\theta})$ and $C_{\text{std}}(\boldsymbol{\theta})$ are the standardized transformations of the observed values of loss and cost, respectively, based on initial observed values $\mathbf{D}$. The acquisition function used to minimize l was $qEVI[\alpha, \beta]$, where $\alpha, \beta = (1 - \alpha)$ are the weights associated with accuracy and cost, respectively.

To benchmark the results of optimization techniques, the HPOBench framework is utilized. The selected benchmark for this study was developed by researchers at HPOBench and is specifically tailored for MLPs. They subjected this benchmark to evaluation using eight publicly available datasets accessible in the OpenML AutoML benchmark. Table 3.1 presents the available datasets.

Table 3.1: OpenML Task IDs used from the AutoML benchmark for MLP, including tid (Task ID), #obs (Number of Observations), and #feat (Number of Features)

name	tid	#obs	#feat
blood-transf	10101	748	4
vehicle	53	846	18
Australian	146818	690	14
car	146821	1728	6
phoneme	9952	5404	5
segment	146822	2310	19
credit-g	31	1000	20
kc1	3917	2109	22

This benchmark encompasses five crucial hyperparameters: the model's depth and width, batch size, L2 regularization, and the initial learning rate for the Adam optimizer. It is offered in both raw and tabular formats. In the tabular version, hyperparameters are discretized into 10 bins, and we assessed the resulting grid, comprising 1.000 configurations for each of 30 distinct architectures, thus totaling 30.000 configurations. Table 3.2 show the hyperparameters configurations of the MLP models of HPOBench.

Table 3.2: Hyperparameters of MLP Benchmark

Name	Type	Log-Scale	Range
alpha	float	yes	$[1.0 \times 10^{-8}, 1.0]$
batch_size	int	yes	$[4, 256]$
depth	int	no	$[1, 3]$
learning_rate_init	float	yes	$[1.0 \times 10^{-5}, 1.0]$
width	int	yes	$[16, 1024]$
epochs	int	no	$[3, 243]$
subsample	float	no	$[0.1, 1]$

A notable feature of this benchmark is its capacity to return values from a lookup table, providing recorded function values of a raw HPO benchmark without the need for actual evaluations. Additionally, this benchmark stands out as it evaluates four distinct metrics (accuracy, balanced accuracy, precision, and F1 score) while recording cost times of the model, enabling multi-objective optimization in a more cost-effective manner.

Using the benchmark is possible to obtain 30.000 points pre-evaluated avoiding the necessity to train the models. Figures 3.1 and 3.2 show, using the segment dataset, the path taken by six different algorithms over 25 and 80 iterations, respectively, in a particular execution: BO (qEI with different weightings), MOBO (qEHVI and qNEHVI), and random search (Sobol), all of them with batch size $q = 4$. BO was designed using qEHVI as the acquisition function and three different loss functions, each with different weightings of loss and cost. The configuration $qEI[1, 0]$ assumes $\alpha = 1$ and $\beta = 0$, representing the classic BO approach that considers only the loss objective. Analogously, $qEI[0.7, 0.3]$ assigns 70% importance to the loss and 30% to the cost. Finally, $qEI[0.5, 0.5]$ treats loss and cost as equally important. In grey, is possible to see all the 30.000 points obtained through the benchmark. Both figures show that in methods which consider cost, the iterations tend to move toward regions closer to the origin, where the models are less expensive. One important detail is that optimizations are only available to evaluate points on the grid created and evaluated by HPOBench, complicating the optimization process, as Bayesian algorithms perform better in continuous hyperparameter spaces. To address this issue, the nearest neighbor approach can be employed. By mapping the continuous hyperparameter values to their closest counterparts on the predefined grid, it is possible to approximate the evaluation of these points while still leveraging the grid-based setup of HPOBench.

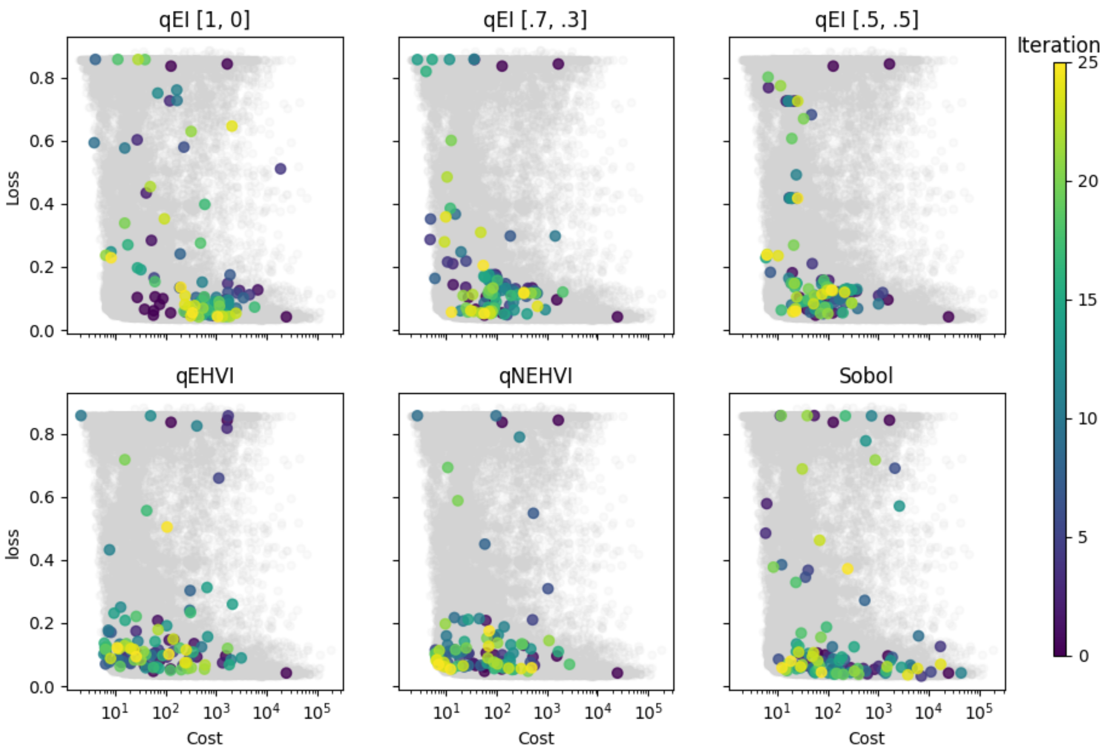


Figure 3.1: **Searching through 25 iterations**. The figure displays the performance of various optimization methods over 25 iterations. The methods include univariate Bayesian searches (qEI with different weightings of 1, 0.7, and 0.5), bivariate Bayesian searches (qEHVI, qNEHVI), and a random search (Sobol). Each plot shows the path followed by each method 25 steps after an initial sample. The color gradient represents the iteration number, with darker colors indicating earlier iterations and lighter colors indicating later iterations. The results show that when considering cost, the methods tend to explore regions closer to the origin.

Visually, Figures 3.1 and 3.2 indicate that the single-objective optimization which weights objectives ($qEI[0.7, 0.3]$ and $qEI[0.5, 0.5]$), is more effective than $qEI[1, 0]$ in finding models in promising regions, where both the loss and the cost are lower, nevertheless, the MOBO algorithms presented points closer to the origin. Furthermore, use the loss function by weighting multiple objectives involves making assumptions about the weights before the optimization process, which requires some prior knowledge of how the objectives relate to each other. On the other hand, Multi-Objective Bayesian Optimization (MOBO) provides a significant advantage as it allows researchers to explore and evaluate a broader set of solutions (Pareto-optimal

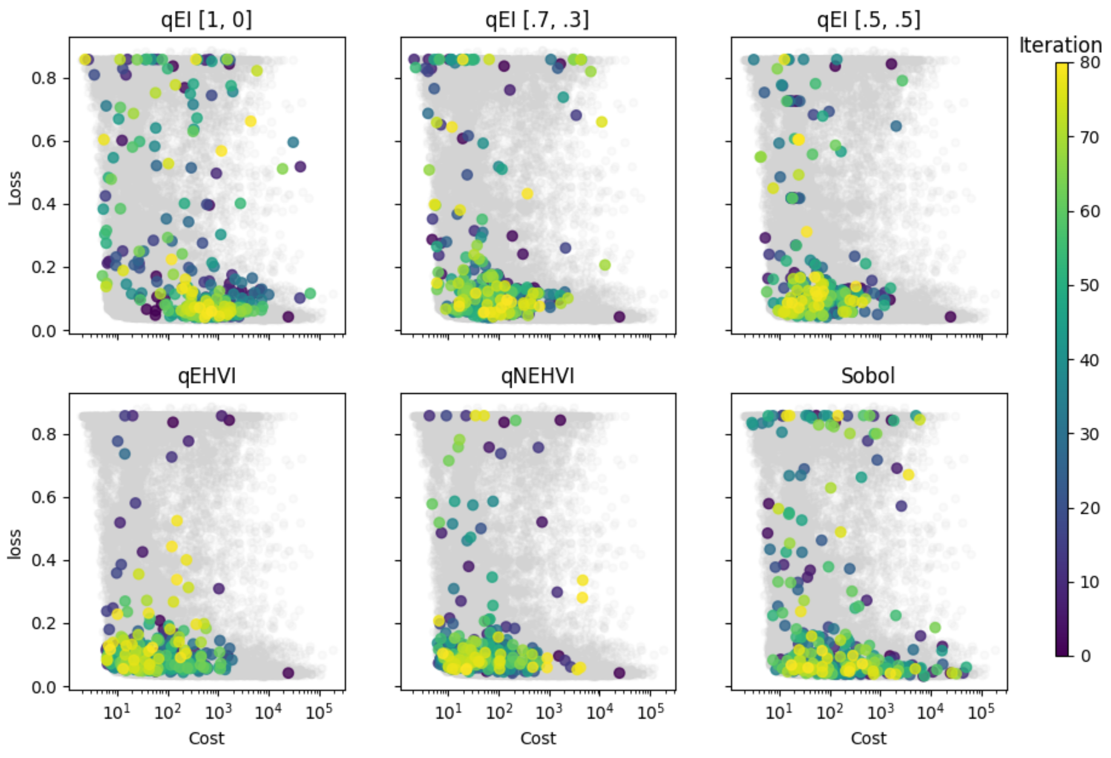


Figure 3.2: **Searching through 80 iterations**. The figure extends the previous analysis to 80 iterations, providing a deeper insight into the performance of the optimization methods over a longer period. The results further emphasize that as the number of iterations increases, the methods increasingly focus on regions near the origin in terms of cost.

solutions), after the optimization process, giving researchers the ability to balance loss and cost, more effectively. Additionally, qNEVI presented late observed points with values at the corner of objectives, indicating that the algorithm is capable of effectively exploring and exploiting the solution space to find optimal or near-optimal solutions close to the origin.

To better compare the algorithms, we conducted a simulation of 100 optimizations, each consisting of 25 steps under the same conditions as before. Each simulation started from 10 different initial points, and we calculated the average loss and cost. To ensure a consistent comparison, the step of selecting among the points on the Pareto frontier, typically done by the researcher, was automated by choosing the point with the smallest Mahalanobis distance from the origin. This solution is one of the simplest and was adopted because it provides an objective and reproducible method to select a representative point on the Pareto frontier, minimizing the influence of subjective decisions in the comparison process.

Figure 3.3 presents the distribution of the chosen points in the Cartesian plane over the simulations. Visually, it is evident that both MOBO methods select cheaper models without sacrificing too much loss. Table 3.3 shows that the average accuracy ($1 -$ loss) achieved by qEHVI is only approximately 6.3% less than the best accuracy obtained by the best algorithm, random search, but the average total cost is 53% less than that of the random implementation. The method used to make the decision, based on the distance to the origin, is the simplest possible. Future investigations regarding better decision-makers could explore more sophisticated techniques to achieve even better results.

Therefore, even if the researcher is not focused on finding the most efficient models, using MOBO can be a good solution as it prioritizes evaluating models near the Pareto frontier, reducing cost and speeding up hyperparameter tuning.

Surprisingly, the random search algorithm performed better than the Bayesian optimization in the evaluated dataset. This is an important limitation of this study and one possible explanation for this phenomenon is that Bayesian optimization is designed to work in a continuous domain. Since in this case the inputs are discrete, Bayesian techniques may struggle, diminishing their performance (Luong et al., 2019). The presence of a large mass of points in regions with low loss function values also contributes to the sampling of these regions. Subsequent studies could explore hybrid approaches that combine Bayesian optimization with techniques specifically tailored for discrete spaces, or modify the acquisition functions to better accommodate the discrete nature of the inputs, potentially improving the algorithm's performance in similar scenarios.

Table 3.3: Results of 100 Simulations Comparing Different Optimization Algorithms

Method	Average Loss	Average Total Cost
qEI [1, 0]	0.0365	168453.93
qEI [0.7, 0.3]	0.0381	46173.97
qEI [0.5, 0.5]	0.0387	39737.67
qEHVI	0.0380	46577.76
qNEHVI	0.0381	**44500.89**
Random	**0.0357**	99332.22

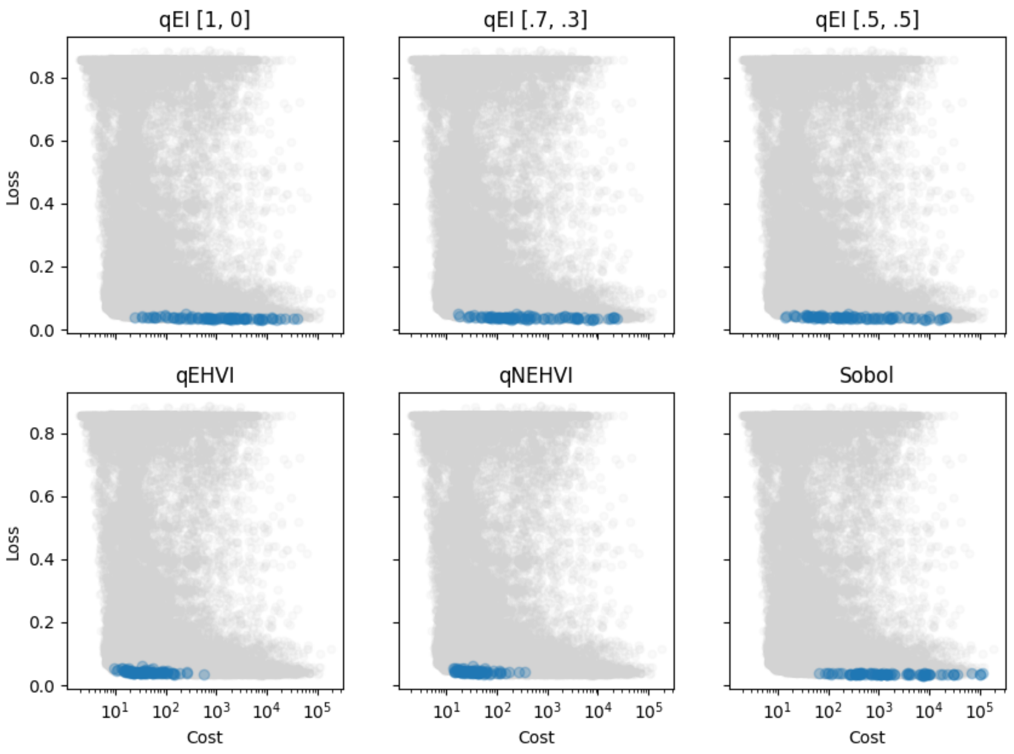


Figure 3.3: **Comparison of optimization methods through 100 simulations**. The plots illustrate how multi-objective techniques tend to find more cost-effective solutions without significantly sacrificing performance (loss) across 100 simulations, each consisting of 25 steps.

Chapter 4

Discussion and Future Works

The results of this study reveal several important insights into the efficiency of Multi-Objective Bayesian Optimization (MOBO) compared to traditional Bayesian Optimization (BO) and random search methods. Despite the surprising performance of the random search algorithm, which outperformed Bayesian methods, the advantages of MOBO in terms of cost efficiency and model evaluation speed were evident in the conducted simulation.

While random search achieved the best accuracy, MOBO methods, particularly qEHVI, a simulation demonstrated a significant reduction in total cost (53% less than random search). This indicates that MOBO can be highly effective in scenarios where computational resources and time are constrained.

Another critical finding is the challenge posed by the discrete nature of the hyperparameter space. Bayesian optimization techniques are generally designed for continuous domains, and their performance can be hindered when applied to discrete inputs. This limitation was reflected in the reduced performance of Bayesian methods in this study. Future research should explore strategies to enhance the effectiveness of Bayesian optimization in discrete spaces, potentially through hybrid methods or modifications to existing algorithms.

The application of Ad-hoc techniques as knowledge distillation and quantization should be quantified, once they were just commented as a possibility to finds efficient models. These methods could also be integrated into the MOBO framework to create even more compact and resource-efficient models.

References

Akaike, H. (1974). "A new look at the statistical model identification". *IEEE Transactions on Automatic Control* 19.6, pp. 716–723. DOI: 10.1109/TAC.1974.1100705.

Balandat, Maximilian et al. (2020). "BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization". *Advances in Neural Information Processing Systems 33*. URL: http://arxiv.org/abs/1910.06403.

Braga, Antônio Pádua et al. (2006). "Multi-Objective Algorithms for Neural Networks Learning". In: ed. by Yaochu Jin. Springer Berlin Heidelberg, pp. 151–171. ISBN: 978-3-540-33019-6. DOI: 10.1007/3-540-33019-4_7. URL: https://doi.org/10.1007/3-540-33019-4_7.

Brochu, Eric, Cora, Vlad M, and De Freitas, Nando (2010). "A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning". arXiv: 1012.2599v1.

Curry, B. and Morgan, P. H. (2006). "Model selection in Neural Networks: Some difficulties". *European Journal of Operational Research* 170.2, pp. 567–577. ISSN: 0377-2217. DOI: 10.1016/J.EJOR.2004.05.026.

Daulton, Samuel, Balandat, Maximilian, and Bakshy, Eytan (2020). "Differentiable Expected Hypervolume Improvement for Parallel Multi-Objective Bayesian Optimization". *Advances in Neural Information Processing Systems*. Vol. 33, pp. 9851–9864.

Hinton, Geoffrey and Dean, Jeff (2015). "Distilling the Knowledge in a Neural Network".

Horn, Daniel and Bischl, Bernd (2016). “Multi-Objective Parameter Configuration of Machine Learning Algorithms using Model-Based Optimization”.

Jacob, Benoit et al. (2017). “Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference”.

Luong, Phuc et al. (Dec. 2019). “Bayesian Optimization with Discrete Variables”. In: pp. 473–484. ISBN: 978-3-030-35287-5. DOI: `10.1007/978-3-030-35288-2_38`.

Moody, John E (1992). “The EEective Number of Parameters: An Analysis of Generalization and Regularization in Nonlinear Learning Systems”, pp. 847–854.

Schneider, Lennart et al. (2021). “HPOBench: A Collection of Reproducible Multi-Fidelity Benchmark Problems for HPO”. *arXiv preprint arXiv:2109.06716*.

Snoek, Jasper, Larochelle, Hugo, and Adams, Ryan P (2012). “PRACTICAL BAYESIAN OPTIMIZATION OF MACHINE LEARNING ALGORITHMS”. arXiv: `1206.2944v2`.

Takahashi, Ricardo HC (2004). “Otimização escalar e vetorial”.

Zhang, Guoqiang, Eddy Patuwo, B., and Y. Hu, Michael (1998). “Forecasting with artificial neural networks:: The state of the art”. *International Journal of Forecasting* 14.1, pp. 35–62. ISSN: 0169-2070. DOI: `10.1016/S0169-2070(97)00044-7`.

Zitzler, E. and Künzli, S. (2004). “Indicator-based selection in multiobjective search”. *Parallel Problem Solving from Nature-PPSN VIII*. Springer, pp. 832–842.

Zitzler, Eckart et al. (2003). “Performance assessment of multiobjective optimizers: An analysis and review”. *IEEE Transactions on Evolutionary Computation* 7.2, pp. 117–132.