University College London

Department of Computer Science

Advancing Bayesian Optimisation in Quantitative Finance: A Case Study in Framework Comparison and Model Extension

Author

Candidate Number MWSZ2

Academic Supervisor

Professor Philip Treleaven

Department of Computer Science

University College London

Industrial Supervisor

Dr Michal Galas

Banking Science Ltd

September 8, 2025

Abstract

This thesis explores original research on the application and extension of Bayesian Optimisation (BO) for hyperparameter tuning in black-box portfolio selection models. It addresses key methodological challenges, including the comparison of Tree-structured Parzen Estimator (TPE) based frameworks, the transition from sequential to batch optimisation, and the robustness of BO-derived configurations in a real-world financial context. Using both synthetic benchmarks and a proprietary model provided by an industry partner, the thesis demonstrates the advantages of Optuna over Hyperopt, shows how batch BO can accelerate convergence and improve runtime efficiency, and analyses how BO-based calibration affects model stability and interpretability. Together, these findings contribute theoretical insight and empirical evidence on how BO can be made more effective and robust in practical financial applications.

The thesis consists of the following three investigations:

1. **Framework Benchmarking on Synthetic Function.** A benchmark study on the Ackley function compares the performance of Hyperopt and Optuna under identical TPE-based settings. The evaluation covers convergence behaviour, runtime and stability across multiple runs, providing the basis for selecting the framework used in subsequent analyses.

2. **Batch Bayesian Optimisation in Financial Black-Box.** Extending the proprietary portfolio selection model from sequential to batch Bayesian Optimisation, this study assesses the impact of parallel evaluations on runtime efficiency and the consistency of optimisation outcomes.

3. **Sensitivity and Robustness Analysis.** The final study conducts a sensitivity and robustness analysis to identify hyperparameter importance and to examine the stability of optimal solutions under local perturbations, thereby offering insights into the interpretability and reliability of BO-tuned models.

The scientific contributions of this thesis are:

1. **Comparison of TPE-based Bayesian optimisation frameworks.** A systematic comparison of Hyperopt and Optuna demonstrates clear performance advantages of Optuna and links these differences to distinct design choices in candidate selection and thresholding, thereby providing guidance for framework selection in practice.

2. **Advancing batch Bayesian optimisation for financial black-box models.** The thesis formalises and implements the extension from sequential to batch Bayesian Optimisation. It combines theoretical justification of batch strategies with empirical evidence, showing that parallel evaluation improves runtime efficiency while preserving convergence quality.

3. **Robustness analysis of BO-tuned solutions.** A sensitivity and robustness framework analysis quantifies parameter importance and stability under perturbations, highlighting conditions under which solutions remain stable. This advances the understanding of reliability and interpretability in BO-driven financial optimisation.

A scientific publication based on this research is planned. All code and experimental implementations will be made openly available through the project's GitHub repository: `https://github.com/lfox1209/bayesian-optimisation-thesis`.

Impact Statement

This research addresses critical industry needs in portfolio management, where investment funds demand models that are practical, efficient to calibrate, and robust under changing market conditions. The findings of this work provide relevant implications for investment practice:

1. **Efficiency gains.** Batch Bayesian Optimisation reduces wall-clock runtime for portfolio calibration, enabling faster strategy iteration and quicker adaptation to market shifts.

2. **Performance potential.** Systematic hyperparameter tuning supports the design of more consistent and potentially higher-performing portfolios, validated out-of-sample with a newly implemented utility function that explicitly accounts for crash risks.

3. **Robust decision support.** Robustness analysis identifies which sector-specific expert weights matter most and demonstrates how optimal configurations remain stable under changes in risk–return preferences, helping funds allocate expert resources where they matter most and adapt preferences without destabilising the strategy.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter introduces the general motivation for the thesis and places it in the context of current developments in financial modelling and optimisation. It outlines the central research questions, summarises the objectives and contributions, and provides an overview of the thesis structure.

1.1 Research Motivation

The motivation for this thesis arises from the growing importance of efficient and reliable model calibration in quantitative finance. In capital markets, the competitive advantage of quantitative investment strategies increasingly depends not only on model design itself, but on the ability to calibrate models quickly, robustly and with limited resources. A recent McKinsey report estimates that up to 40% of quantitative teams' computing power is used for model parameter search and backtesting workflows [25]. Financial practitioners rely on these models to allocate capital, construct equity portfolios or execute algorithmic trading strategies. Many of these models, often proprietary and black-box in nature, require the specification of several hyperparameters that define their behaviour but are not directly estimated from data. These hyperparameters influence risk exposure, timing decisions, factor sensitivities and the rebalancing frequency of trading decisions. Their optimal configuration is typically determined by evaluating a utility function that aggregates financial performance metrics over historical data.

In practice, such utility evaluations are expensive as each trial may involve a full-scale backtest over multiple year timeframes, including portfolio rebalancing logics,

transaction cost modelling and market dynamics. This leads to optimisation problems where the objective is costly, non-differentiable, may involve noise or randomness and is defined only through simulation. Traditional tuning approaches like grid or random search scale poorly in this setting [31], especially when the dimensionality of the search space increases or when evaluation budgets are limited [4].

Bayesian Optimisation (BO) addresses these challenges by constructing a probabilistic surrogate model of the objective and using an acquisition function to decide where to evaluate next. This makes it a natural choice for financial applications, where sample efficiency and uncertainty quantification are of particular importance [28]. In contrast to exhaustive search or gradient-based methods, BO can identify high-performing regions of the search space while requiring relatively few evaluations, which is a crucial property when each backtest takes minutes to hours.

In particular, BO has proven effective for hyperparameter tuning of black-box models in settings where the objective reflects not only return maximisation, but also drawdown control, benchmark outperformance and portfolio robustness [3]. This thesis builds on a utility-based portfolio selection model, where candidate configurations are evaluated via DRACUS, a proprietary backtesting system provided by an industry partner that translates hyperparameter configurations into financial outcomes. The utility function reflects both short-term and long-term financial goals, combining return expectations with downside protection criteria. Its structure motivates the use of BO not just as an optimisation tool, but also as a framework for interpretable model exploration.

At the same time, the practical application of BO in financial contexts faces three main challenges, namely the lack of clear guidance on framework selection, the inefficient use of available parallel resources in sequential optimisation, and the difficulty of assessing the robustness of optimised configurations. These considerations motivate the research objectives formulated in the next section and provide the foundation for the analyses that follow.

1.2 Research Objectives

Building on the outlined challenges, this thesis aims to evaluate and extend Bayesian Optimisation methods in the context of computationally expensive model calibration for portfolio selection. The focus lies on understanding the empirical behaviour,

practical efficiency, and robustness of BO in a realistic financial setting. To guide the investigation, the following research questions are formulated:

1. How do the two widely used TPE-based frameworks, Hyperopt and Optuna, compare in terms of convergence behaviour, computational efficiency, and solution stability on a controlled benchmark?

2. How does the transition from sequential to batch Bayesian Optimisation affect runtime efficiency and convergence behaviour when optimising a real-world portfolio selection model?

3. Which hyperparameters contribute most to optimisation success, and how robust are the optimal configurations to local perturbations, changes in preference weights, and alternative batch design choices?

1.3 Research Experiments

The research questions introduced in the previous section are addressed through three complementary experiments. Each experiment targets a distinct aspect of Bayesian Optimisation in the context of portfolio model calibration, and together they provide the structure for Chapters 4 to 6.

1. Framework Benchmarking on Synthetic Function. The first experiment investigates the performance of two TPE-based frameworks, Hyperopt and Optuna, in a controlled synthetic setting. Using the Ackley function as benchmark, the study evaluates convergence behaviour, runtime efficiency, and stability across repeated runs. The experiment highlights how the respective program design choices influence optimisation outcomes and provides the basis for selecting the framework applied in later financial experiments.

2. Batch Bayesian Optimisation in Financial Black-Box. The second experiment applies Bayesian Optimisation in batch settings within the proprietary DRACUS backtesting environment. It uses a newly developed utility function that captures both return and crash risk. The experiment analyses how parallel evaluations affect runtime efficiency and convergence, and it examines the resulting portfolio configurations through out-of-sample testing, providing evidence on whether the model can be used reliably in practice.

3. Sensitivity and Robustness Analysis. The third experiment analyses the robustness and interpretability of BO-calibrated solutions. It quantifies the importance of different hyperparameters, tests the stability of optimal configurations under local perturbations, and explores the sensitivity to changes in preference weights and batch design choices. This experiment aims to identify which factors drive optimisation success and under what conditions the solutions remain stable in practice.

1.4 Scientific Contributions

This thesis makes three contributions to the scientific understanding of Bayesian Optimisation, both in general methodological terms and in the specific context of computationally expensive portfolio model calibration:

1. Comparison of TPE-based Bayesian Optimisation frameworks. The thesis systematically compares Hyperopt and Optuna on a synthetic benchmark, demonstrating clear performance differences between the two frameworks. These differences are traced back to distinct design choices in candidate selection and thresholding within their TPE implementations, thereby providing methodological insight into how framework-level mechanisms influence optimisation outcomes. This provides methodological guidance for selecting appropriate frameworks in practice.

2. Advancing batch Bayesian Optimisation for financial black-box models. This work establishes and evaluates a methodological framework for applying batch BO in computationally expensive financial settings. It combines a theoretical justification of batch selection strategies with empirical evidence on runtime efficiency and convergence behaviour, validating the approach out-of-sample. The findings provide evidence that batch BO can be integrated reliably into real-world portfolio optimisation tasks.

3. Robustness analysis of BO-tuned solutions. A sensitivity and robustness study quantifies hyperparameter importance and examines the stability of optimal configurations under perturbations of parameters and preference weights, as well as across alternative batch design choices. This advances the understanding of reliability and interpretability in BO-driven financial optimisation.

1.5 Thesis Structure

The remainder of this thesis is organised as follows. Chapter 2 reviews the relevant literature on Bayesian Optimisation and its applications in finance, and outlines the theoretical background. Chapter 3 describes the methodological setup, specifying the optimisation problem, defining the utility function, and introducing the DRACUS backtesting system. Chapters 4 to 6 present the three core experiments, with Chapter 4 benchmarking two TPE-based frameworks on a synthetic function, Chapter 5 applying batch BO to a proprietary portfolio model and evaluating its out-of-sample performance, and Chapter 6 analysing the robustness and interpretability through hyperparameter importance and perturbation tests. Finally, Chapter 7 concludes the thesis by summarising the main findings, discussing their implications, and outlining directions for future research.

Chapter 2

Literature Review and Background: Bayesian Optimisation in Finance

This chapter outlines the foundations of Bayesian Optimisation in the context of finance. It first reviews the relevant literature and applications to position this work within the existing research. The chapter then introduces the theoretical background of Bayesian Optimisation, details the Tree-structured Parzen Estimator surrogate model, and outlines the acquisition strategies that are central to both sequential and batch settings.

2.1 Literature Review and Applications

Bayesian Optimisation has emerged as one of the most effective frameworks for solving global optimisation problems where the objective function is expensive to evaluate and behaves as a black-box without closed-form structure. The central idea is to use a statistical model that approximates the objective together with a decision rule that guides where to search next. This combination makes BO sample efficient and well-suited for problems where only a limited number of evaluations are possible. Such characteristics have led to wide adoption in machine learning, engineering design and the natural sciences, where optimisation often depends on costly simulations or experiments. Financial applications share many of these challenges, as the objective functions are typically noisy, high-dimensional, and computationally expensive, making classical optimisation approaches often inadequate. As a result, BO has gained increasing attention as a principled approach to hyperparameter tuning, model cali-

bration, and portfolio optimisation within the financial world.

The roots of Bayesian Optimisation go back to the Efficient Global Optimisation (EGO) algorithm [20]. Jones and colleagues established the use of Gaussian processes as surrogate models, providing a probabilistic estimate of the objective function, in combination with the Expected Improvement (EI) criterion, which served as an acquisition function to determine the next point to evaluate. The EI captures the principle of searching in regions that promise the greatest improvement over the current best observation. Building on this foundation, numerous methodological advances have been proposed to extend Bayesian Optimisation beyond its original sequential form, ranging from alternative acquisition criteria to new surrogate models and adaptations for parallel evaluations. These developments have established Bayesian Optimisation as a suitable framework for black-box optimisation across a wide range of scientific and industrial domains and motivate the review of acquisition strategies, surrogate models and extensions in the following.

A central aspect of Bayesian Optimisation is the trade-off between exploring uncertain regions of the search space and exploiting regions already predicted to yield good outcomes. Different acquisition functions reflect different preferences along this trade-off and can therefore be more or less advantageous depending on the context. Early alternatives to the EI include the Probability of Improvement, which favours points with a high chance of improving upon the best observation [22], and the Upper Confidence Bound, which balances predicted performance and uncertainty by selecting points with a high mean and high variance [2]. Subsequent research has proposed information-based methods, such as Entropy Search, Predictive Entropy Search, and Max-value Entropy Search, which take a different approach by explicitly looking for information about the optimum's location [16, 18, 33]. In contrast to the Probability of Improvement or Upper Confidence Bound, which balance exploration and exploitation at each step, these approaches are more globally exploratory by actively reducing uncertainty about the location of the optimum. Another influential strategy is the Knowledge Gradient, which evaluates the expected value of information obtained from each new evaluation [35]. While these alternatives can outperform EI in specific situations, they are often more complex and computationally expensive. In practice, EI remains the most widely used baseline, with the other approaches serving as valuable alternatives in specialised settings [6, 28, 31].

While Gaussian processes have been the classical choice of surrogate model in Bayesian Optimisation, they face practical limitations in high-dimensional search spaces or

when dealing with categorical or conditional variables as well as large datasets. These challenges have motivated the development of alternative models. The Tree-structured Parzen Estimator (TPE) formulates Bayesian Optimisation as a density estimation problem by modelling how likely different input configurations are to lead to good or bad outcomes, based on the observations collected so far [4]. New candidates get selected based on the ratio of densities associated with good and bad outcomes. Another influential framework is the Sequential Model-based Algorithm Configuration, which uses random forests as surrogates and is particularly well suited for discrete–continuous search spaces [19]. Both methods have become widely used in hyperparameter optimisation of machine learning models, where flexibility and scalability are more important than the exact probabilistic calibration done by Gaussian processes. These developments show how the choice of surrogate model strongly depends on the structure of the optimisation problem and provide the motivation for our later focus on TPE as the surrogate for our optimisation task in this thesis.

The classical sequential design of Bayesian Optimisation has become increasingly limiting in applications where multiple evaluations can be run in parallel. Modern computing infrastructures and many scientific applications, such as the training of machine learning models or large-scale simulations, naturally support parallel executions. To take advantage of these settings, batch Bayesian Optimisation was developed to evaluate multiple points at once, thereby reducing wall-clock time without sacrificing sample efficiency. Several strategies have been developed to adapt acquisition functions to this context. Early approaches include the Constant Liar and Kriging Believer methods, which iteratively select multiple points by temporarily assigning fictitious outcomes to previously chosen candidates [13]. More advanced methods aim to encourage diversity within a batch, for example Local Penalisation, which penalises regions close to previously chosen points [14], or Thompson Sampling, which samples functions from the surrogate's posterior distribution and then selects the maxima or minima of these functions as candidates [17]. A more comprehensive and theoretically grounded alternative is the q-Expected Improvement, which extends the Expected Improvement by evaluating the joint expected gain from a batch of points [7]. The suitability of these methods depends strongly on the underlying surrogate model, since not all batch strategies can be applied outside of Gaussian processes. In Section 2.4, we will analyse Constant Liar, Thompson Sampling, and q-Expected Improvement in greater mathematical detail. This provides the basis for a comparison of their suitability in TPE-based Bayesian Optimisation, from which we will derive a choice of the batch strategy for the optimisation task studied in this

thesis.

The practical use of Bayesian Optimisation often faces additional challenges. One important case is optimisation under constraints, where both the objective and the feasibility of candidate points must be modeled jointly. To address this, methods such as constrained EI have been developed [12]. Another issue arises when observations are noisy, which is common in financial backtests or stochastic simulations. Approaches like augmented EI and quantile-based objectives aim to take this uncertainty into account and provide more reliable performance [24, 26]. A further line of research deals with the challenges of high dimensionality. Standard Gaussian processes often struggle in high-dimensional spaces, which has motivated the use of methods such as random embeddings [34], additive structure modelling [21] and local trust-region approaches [10]. These developments show how BO has been adapted to noisy, constrained and high-dimensional settings, which makes it suitable for complex real-world problems and especially relevant in finance.

This relevance is demonstrated in a range of financial applications where Bayesian Optimisation has already been explored. In model calibration, it has been applied to option pricing and volatility surface fitting, where the objective functions are highly non-linear and simulations are computationally demanding [8, 11]. In portfolio optimisation, BO has been used to search for allocations that optimise risk-adjusted measures such as the Conditional Value-at-Risk [23, 29]. Another useful application is to employ BO for tuning hyperparameters in trading strategies and asset prediction models [36, 37]. Similar approaches have also been used in portfolio modelling itself, where hyperparameter optimisation plays a central role in adapting models to different market conditions. This is directly related to the focus of this thesis, which investigates BO for hyperparameter tuning in portfolio selection models.

In summary, the literature on Bayesian Optimisation covers a wide range of surrogate models, acquisition functions and extensions that address challenges such as parallel evaluations, noisy observations and high-dimensional search spaces. Applications in finance demonstrate that BO is a powerful tool whenever objectives are expensive, stochastic and have no explicit analytic structure. However, most existing studies still rely on Gaussian process surrogates, while systematic investigations of density-estimation approaches such as the Tree-structured Parzen Estimator are scarce. In particular, direct comparisons across widely used frameworks like Hyperopt and Optuna, both of which implement TPE in different ways, are largely missing from the literature. Similarly, while batch acquisition strategies have been widely analysed

for Gaussian processes, their integration with TPE has received little attention. This thesis addresses these gaps by providing a framework comparison of TPE implementations, investigating TPE-compatible batch strategies, and applying them to a realistic portfolio optimisation setting. The next sections develop the theoretical background of Bayesian Optimisation in general, and then focus specifically on the Tree-structured Parzen Estimator as surrogate model and on selected batch acquisition strategies.

2.2 Theoretical Background

As outlined in the previous section, Bayesian Optimisation is a global optimisation strategy designed for objective functions that are expensive to evaluate, possibly noisy, and lacking closed-form or gradient information. We now turn to its mathematical formulation. In formal terms, BO aims to solve

$$\min_{x \in \mathcal{X}} f(x) \;\text{ with }\; f(x) : \mathcal{X} \to \mathbb{R},$$

where the feasible domain $\mathcal{X}$ is formally defined as

$$\mathcal{X} = \{\, x \in \mathbb{R}^d \mid c_j(x) \leq 0,\ j = 1, \ldots, m\} \subset \mathbb{R}^d,$$

with each c_j representing an inequality constraint. Any functional constraint of the form $c_j(x) \leq b_j$, $c_j(x) = b_j$ or $c_j(x) \geq b_j$ can be equivalently reformulated in this standard form, for instance by rewriting $c_j(x) \geq b_j$ as $-c_j(x)+b_j \leq 0$ and representing equality constraints via two inequalities. Without loss of generality, we focus on minimisation problems as maximisation problems $\max f(x)$ can be equivalently posed as $\min -f(x)$. This unifies the treatment and notation throughout the thesis.

As f is a black-box function over a compact search domain, we need to build a surrogate model, typically consisting of a probabilistic estimator $\hat{f}$ of the true function f and an acquisition function $\alpha_t(x)$ that guides the search for optimal inputs. Starting with an initial dataset of t prior evaluations

$$\mathcal{D}_t = \{(x_i, y_i)\}_{i=1}^{t} \;\text{ with }\; y_i = f(x_i) + \varepsilon_i, \tag{2.1}$$

where $\varepsilon_i \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ may model observational noise, the surrogate defines a posterior distribution $p(f \mid \mathcal{D}_t)$, expressing uncertainty over f conditioned on the observed data $\mathcal{D}_t$. The form of the posterior $p(f \mid \mathcal{D}_t)$ depends on the choice of the surrogate

model. For example, choosing a Gaussian process yields to closed-form posteriors, while random forests approximate them empirically. In this thesis, we focus on the Tree-structured Parzen Estimator, which we analyse in detail in Section 2.3.

The acquisition function $\alpha_t(x)$ quantifies the utility of evaluating f at a candidate point x, guiding the search toward promising regions of the search space. At each iteration, the next evaluation point is therefore selected by maximising the acquisition function

$$x_{t+1} = \arg\max_{x \in \mathcal{X}} \alpha_t(x),$$

or in the case of batch selection, by choosing a set of q points that will then be evaluated simultaneously

$$\{x_{t+1}^{(1)}, \ldots, x_{t+1}^{(q)}\} = \underset{\{x^{(1)},\ldots,x^{(q)}\} \subset \mathcal{X}}{\arg\max} \alpha_t(x^{(1)}, \ldots, x^{(q)}).$$

The acquisition function thus balances the exploration of uncertain regions and the exploitation of areas of the search space predicted to yield favourable objective values. A widely used acquisition strategy is the Expected Improvement (EI), which in the sequential setting is given by

$$\alpha_t^{\mathrm{EI}}(x) = \mathbb{E}\left[\max(y^* - f(x), 0) \mid \mathcal{D}_t\right], \tag{2.2}$$

where y^* is the best observed value so far. In batch Bayesian optimisation, its natural extension is the q-Expected Improvement (q-EI), defined as

$$\alpha_t^{\text{q-EI}}(x^{(1)}, \ldots, x^{(q)}) = \mathbb{E}\left[\max(y^* - \min_{j=1,\ldots,q} f(x^{(j)}), 0) \mid \mathcal{D}_t\right]. \tag{2.3}$$

In both cases, the expectation quantifies the average amount by which a new evaluation at x is expected to improve over the current minimum y^*, given the surrogate model and the observed data $\mathcal{D}_t$. In this thesis, we focus on batch Bayesian optimisation. Specific batch acquisition strategies, including Constant Liar, Thompson Sampling and q-EI, are analysed in detail in Section 2.4.

Bringing these components together, the general structure of Bayesian Optimisation can be formulated as an iterative procedure that combines probabilistic modelling with acquisition-driven exploration. At each iteration t, a surrogate posterior is constructed based on the current dataset $\mathcal{D}_t$ and used to identify the next input(s) that balance potential improvement and model uncertainty. This formulation naturally ex-

tends from sequential to batch settings by generalising the acquisition over multiple candidates. The general procedure is shown in the following Algorithm 1.

Algorithm 1 Bayesian Optimisation

1: **Input:** Initial dataset $\mathcal{D}_0$, batch size q, evaluation budget T
2: **for** $t = 0$ to $T - 1$ **do**
3: Fit surrogate model:

$$\hat{f}_t \sim p(f \mid \mathcal{D}_t)$$

4: Select next input(s) by maximising the acquisition function $\alpha_t(x)$:

$$\{x_{t+1}^{(j)}\}_{j=1}^{q} = \underset{\{x^{(j)}\} \subset \mathcal{X}}{\arg\max}\, \alpha_t(x^{(1)}, \ldots, x^{(q)})$$

5: Evaluate objective black-box function:

$$y_{t+1}^{(j)} = f(x_{t+1}^{(j)}) + \varepsilon_{t+1}^{(j)}, \quad \text{for } j = 1, \ldots, q$$

6: Update dataset:

$$\mathcal{D}_{t+1} = \mathcal{D}_t \cup \{(x_{t+1}^{(j)}, y_{t+1}^{(j)})\}_{j=1}^{q}$$

7: **end for**

The process repeats until a predefined budget T is exhausted or alternatively one could implement a criterion to detect convergence. In the special case $q = 1$, this reduces to the standard sequential formulation of Bayesian Optimisation. We adopt this generalised formulation throughout, enabling both sequential and batch variants to be treated within a unified framework.

The algorithmic formulation shows why Bayesian Optimisation is particularly suitable for the setting considered in this thesis. By explicitly modelling uncertainty and prioritising informative evaluations, BO avoids the inefficiencies of other search methods and the inapplicability of gradient-based methods to non-differentiable objectives. These properties make it well-suited for calibration problems in finance, where each evaluation corresponds to a costly backtest with structural constraints and noise. In the following, we build on this theoretical foundation by detailing the Tree-structured Parzen Estimator surrogate model and by analysing acquisition strategies for batch Bayesian Optimisation, which will together form the methodological basis for our empirical study.

2.3 TPE-Based Surrogate Modelling

Building on the discussion of surrogate models in Section 2.1, we now turn to the Tree-structured Parzen Estimator, which provides a density-estimation based alternative to Gaussian processes. Unlike GP-based approaches that model the conditional density $p(y \mid x)$, TPE reverses the direction and models $p(x \mid y)$, i.e., instead of predicting the objective value at a given input, the model defines a distribution over input configurations conditional on their observed performance. This inversion simplifies the density estimation process and will enable efficient sampling from the search space. It furthermore forms the basis for the optimisation software frameworks analysed in this thesis, namely Hyperopt and Optuna.

At each iteration t, the current dataset $\mathcal{D}_t$, as defined in Equation (2.1), is partitioned into two disjoint subsets, depending on whether the observed objective value lies below or above a threshold τ, defined as the γ-quantile of the observed values, i.e.,

$$\tau = \text{Quantile}_\gamma(\{y_i\}_{i=1}^t) \quad \text{with} \quad \gamma \in (0, 1),$$

which then yields to

$$\mathcal{D}_1 = \{x_i \mid y_i < \tau\} \quad \text{and} \quad \mathcal{D}_2 = \{x_i \mid y_i \geq \tau\}.$$

This decomposition separates promising regions of the search space from less promising ones, which will enable targeted modelling and sampling in the following. We first define the two induced conditional densities, given by

$$\ell(x) := p(x \mid y < \tau) \quad \text{and} \quad g(x) := p(x \mid y \geq \tau),$$

which are estimated independently using a kernel density estimation (KDE). KDE is a non-parametric method for approximating probability densities based on finite samples. For instance, the density $\ell(x)$ is approximated as

$$\hat{\ell}(x) = \frac{1}{|\mathcal{D}_1|} \sum_{x_i \in \mathcal{D}_1} K_h(x - x_i),$$

where K_h is a kernel function, typically a Gaussian kernel with bandwidth parameter h, which controls the smoothness of the estimation. In the one-dimensional case, it

is given by

$$K_h(x - x_i) = \frac{1}{\sqrt{2\pi h^2}} \exp\left(-\frac{(x - x_i)^2}{2h^2}\right).$$

The same idea extends to d dimensions by using product kernels or a full multivariate Gaussian kernel. In particular, the Gaussian kernel is a natural choice, since it yields to smooth local approximations around each sample and, at the same time, it has well-understood analytical properties as well as asymptotic consistency [30]. Under certain conditions, the resulting density estimator converges to the true distribution as the sample size grows [27]. The same approach is used to construct $\hat{g}(x)$ from the $\mathcal{D}_2$ subset.

Now that the surrogate model is specified, the TPE method proposes a new candidate point x_{t+1}, which is then evaluated in the true objective function, by maximising the Expected Improvement (EI). We focus here on the sequential setting, as the extensions to batch Bayesian optimisation will be discussed in Section 2.4. We recall from Equation (2.2) that the EI at a point x is defined as

$$\mathrm{EI}(x) = \mathbb{E}_{y \sim p(y|x)}\left[\max(y^* - y, 0)\right] = \int_{-\infty}^{y^*} (y^* - y) \cdot p(y \mid x)\, dy,$$

where y^* denotes the current best observed value. For the following derivation, we therefore write without loss of generality, $\gamma = p(y < y^*)$ instead of the equivalent formulation with τ, which is only used in practice. By applying the law of total probability with respect to y^*, the marginal distribution $p(x)$ is therefore given by

$$\begin{aligned} p(x) &= p(x \mid y < y^*) \cdot p(y < y^*) + p(x \mid y \geq y^*) \cdot p(y \geq y^*) \\ &= \gamma \cdot \ell(x) + (1 - \gamma) \cdot g(x). \end{aligned}$$

By using this identity and applying Bayes' rule we will transform the EI from Equation (2.2) as follows

$$\begin{aligned} \mathrm{EI}(x) &= \int_{-\infty}^{y^*} (y^* - y) \cdot p(y \mid x)\, dy \\ &= \int_{-\infty}^{y^*} (y^* - y) \cdot \frac{p(x \mid y) \cdot p(y)}{p(x)}\, dy \\ &= \frac{1}{p(x)} \int_{-\infty}^{y^*} (y^* - y) \cdot p(x \mid y) \cdot p(y)\, dy. \end{aligned}$$

The integral can now be simplified as follows using the assumption that $p(x \mid y) = \ell(x)$

for all $y < y^*$

$$\begin{aligned}\int_{-\infty}^{y^*} (y^* - y) \cdot p(x \mid y) \cdot p(y)\, dy &= \ell(x) \int_{-\infty}^{y^*} (y^* - y) \cdot p(y)\, dy \\ &= \ell(x) \left(y^* \int_{-\infty}^{y^*} p(y)\, dy - \int_{-\infty}^{y^*} y \cdot p(y)\, dy \right) \\ &= \ell(x) \cdot \gamma y^* - \ell(x) \int_{-\infty}^{y^*} y \cdot p(y)\, dy.\end{aligned}$$

Inserting this equation finally leads to

$$\begin{aligned}\mathrm{EI}(x) &= \frac{1}{p(x)} \int_{-\infty}^{y^*} (y^* - y) \cdot p(x \mid y) \cdot p(y)\, dy \\ &= \frac{1}{\gamma \cdot \ell(x) + (1-\gamma) \cdot g(x)} \left(\ell(x) \cdot \gamma y^* - \ell(x) \int_{-\infty}^{y^*} y \cdot p(y)\, dy \right) \\ &\propto \left(\gamma + \frac{g(x)}{\ell(x)} (1-\gamma) \right)^{-1}.\end{aligned}$$

This final form follows by factoring out $\ell(x)$ and treating the remaining integral term as independent of x. Since the expression is monotonically increasing in $\ell(x)/g(x)$, one can equivalently maximise this ratio instead of the EI to select the next input. This is exactly what the TPE method does by choosing

$$x_{t+1} = \arg\max_x \frac{\ell(x)}{g(x)}.$$

In the remainder of this thesis, we make use of two optimisation frameworks that implement the TPE surrogate strategy, which are Optuna [1] and Hyperopt [5]. Both rely on the modelling approach described above, but differ in the technical implementation. A notable distinction lies in their treatment of the threshold parameter γ. Hyperopt uses a fixed default value (typically $\gamma = 0.15$), while Optuna employs a dynamic threshold that adapts during the optimisation process. A more detailed comparison of the two frameworks will be presented in Chapter 4, where we also evaluate their empirical performance. To complete the methodological foundation of this thesis, we now turn to the design of acquisition strategies, which guide the selection of new candidates in both sequential and batch Bayesian Optimisation.

2.4 Acquisition Strategies

While the surrogate model provides a probabilistic approximation of the objective function, it is the acquisition strategy that determines how this model is exploited to guide the optimisation process. Especially in batch Bayesian Optimisation, where multiple candidate points are selected simultaneously, the design of acquisition strategies becomes very important. The goal is to find a balance between exploitation of known promising regions and exploration of uncertain areas, while also ensuring diversity within the batch to avoid redundant evaluations. In this section, we discuss the three main acquisition strategies, being the Constant Liar, Thompson Sampling, and q-Expected Improvement. We will highlight their properties, implementation aspects, and compatibility with the TPE surrogate framework.

One of the most widely used and practically effective strategies for batch Bayesian Optimisation is the Constant Liar method [13]. It extends a sequential acquisition strategy to batch selection by iteratively selecting new candidate points under the assumption of "lied" objective function values for those who were previously selected. As shown in Algorithm 2, the procedure starts with the current dataset and incrementally builds a temporary dataset by adding one selected point at a time. In each iteration, a surrogate model is fitted to the current state of this temporary dataset, and the point maximising the acquisition function is selected and added with a fixed, artificial objective value. This process continues until q candidates have been identified. Once the batch is complete, all q points are evaluated using the true objective function.

The choice of the artificial function value c, referred to as the liar value, influences the balance between exploration and exploitation in the batch. Common options include the

$$\text{max liar: } c = \max\{y_i\}_{i=1}^{t}, \quad \text{min liar: } c = \min\{y_i\}_{i=1}^{t}, \quad \text{mean liar: } c = \frac{1}{t}\sum_{i=1}^{t} y_i.$$

Setting c to a high value (max liar) favours exploitation, since the region then appears favourable to the model, making similar points more attractive. Conversely, assigning a low value (min liar) encourages exploration, as the selected region is classified into the low-performing density $g(x)$, and subsequent points will be chosen in other areas. Therefore, in practical settings, the mean liar strategy is often used as a compromise between these two extremes.

Algorithm 2 Constant Liar Strategy

1: **Input:** Current dataset $\mathcal{D}_t$, batch size q, liar value $c \in \mathbb{R}$
2: Initialise temporary dataset $\tilde{\mathcal{D}} \leftarrow \mathcal{D}_t$
3: **for** $j = 1$ to q **do**
4: Fit surrogate model to $\tilde{\mathcal{D}}$
5: Select next point by maximising acquisition function:

$$x^{(j)} = \arg\max_{x \in \mathcal{X}} \alpha_t(x)$$

6: Add liar observation to temporary dataset:

$$\tilde{\mathcal{D}} \leftarrow \tilde{\mathcal{D}} \cup \{(x^{(j)}, c)\}$$

7: **end for**
8: **Return:** Batch $\{x^{(j)}\}_{j=1}^{q}$

The Constant Liar method is simple to implement and efficiently extends any sequential acquisition strategy to the batch setting. Its main advantage lies in its broad applicability, as the strategy does not require sampling from the surrogate posterior and is fully compatible with TPE-based models. However, the quality of the batch depends on the lied value c, and suboptimal strategy choices may reduce diversity or introduce bias.

A second method for constructing batches in Bayesian Optimisation is Thompson Sampling, which is a strategy that selects points based on samples from the surrogate model's posterior. As outlined in Algorithm 3, the batch is constructed iteratively by first sampling a function $f^{(j)}$ from the posterior distribution $p(f \mid \mathcal{D}_t)$ and then selecting the point that minimises this function. This process is repeated q times in order to obtain a batch of q candidates. Since each function sample reflects a different plausible realisation of the true objective, the resulting batch naturally balances exploitation and exploration, often naturally leading to diverse candidates. An alternative formulation of batch Thompson Sampling draws a single function $f^{(1)} \sim p(f \mid \mathcal{D}_t)$ from the posterior and partitions the input space $\mathcal{X}$ into q regions. The q batch points then correspond to the local minima of $f^{(1)}$ within each region. However, as discussed in Section 2.3, the Tree-structured Parzen Estimator does not provide a posterior distribution over functions, but instead models the inverse density $p(x \mid y)$.

Algorithm 3 Thompson Sampling

1: **Input:** Current dataset $\mathcal{D}_t$, batch size q
2: Initialise batch $\mathcal{X}_q \leftarrow \emptyset$
3: **for** $j = 1$ to q **do**
4: Sample surrogate function:

$$f^{(j)} \sim p(f \mid \mathcal{D}_t)$$

5: Select point minimising the sampled function:

$$x^{(j)} = \arg\min_{x \in \mathcal{X}} f^{(j)}(x)$$

6: Add point to batch: $\mathcal{X}_q \leftarrow \mathcal{X}_q \cup \{x^{(j)}\}$
7: **end for**
8: **Return:** Batch $\mathcal{X}_q = \{x^{(1)}, \ldots, x^{(q)}\}$

As both versions of Thompson Sampling rely on a functional posterior, they are not directly applicable in this setting. A possible modification to preserve the main idea of Thompson Sampling and to make it compatible with TPE is to adopt a partition-based approximation. Specifically, a large set of candidate points $\tilde{\mathcal{X}} \sim \ell(x)$ could be sampled from the good density used in TPE, divided into q regions, and then the point with the highest acquisition value is selected in each region. This original approach approximates the effect of sampling from diverse posterior realisations while remaining fully compatible with the TPE framework.

For completeness, we briefly mention the q-Expected Improvement strategy, even though it is not compatible with the TPE framework. This approach selects a set of q points that jointly maximise the Expected Improvement over the current best observation y^*. The corresponding acquisition function, as introduced in Equation (2.3), is given by

$$\alpha_t^{\text{q-EI}}(x^{(1)}, \ldots, x^{(q)}) = \mathbb{E}_{f(X)}\left[\max\left(y^* - \min_{j=1,\ldots,q} f(x^{(j)}), 0\right)\right],$$

with $X := (x^{(1)}, \ldots, x^{(q)})$. Evaluating this expression requires access to the joint distribution of $f(X)$, which is not available in TPE-based models. In contrast, Gaussian Process surrogates assume a multivariate normal distribution $f(X) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and therefore allow the expectation to be approximated via high-dimensional sampling or

numerical integration.

The three acquisition strategies discussed in this section represent the most prominent approaches to batch selection in Bayesian Optimisation. While the q-Expected Improvement is not compatible with TPE-based surrogates, the Constant Liar strategy can be directly applied within the TPE framework and will therefore be used as our batch method in the empirical study in Chapter 5. Thompson Sampling, which in principle could be adapted to TPE according to our proposition, is left for future research as its functioning requires further validation, but it provides an interesting direction for extending the analysis and batch BO in general. Before turning to these experiments, we now specify the optimisation problem, define the utility function, and introduce the DRACUS backtesting system.

Chapter 3

Methodology: Problem Setup and Model Specification

This chapter formalises the optimisation problem at the centre of the thesis and outlines the methodology used to address it. It defines the utility-based objective function, specifies the hyperparameters and search space, and describes the DRACUS backtesting system that translates hyperparameter configurations into financial outcomes.

3.1 Optimisation Objective and Utility Function

Building on the theoretical foundations of Bayesian Optimisation outlined in Chapter 2, we now turn to the concrete financial problem at the core of this thesis. Our industry partner provides us with a proprietary portfolio selection model whose performance can only be assessed through computationally expensive backtests. Each evaluation therefore involves simulating longterm portfolio dynamics, including frequent rebalancing and the inclusion of transaction costs and market frictions. These factors make the objective function costly and non-differentiable. Therefore, this setting naturally motivates the use of Bayesian Optimisation, as it is designed for problems where evaluation budgets are limited and analytical gradients are unavailable. A central task is to formalise an optimisation objective that adequately captures economic preferences while remaining suitable for the surrogate-based optimisation. This requires the definition of a utility function that balances return generation, risk control, and robustness under adverse market conditions, thereby capturing the different goals of portfolio management.

We will now formally define our optimisation problem. We will denote by Θ the feasible search space and therefore by $\theta \in \Theta$ a vector of permitted hyperparameters that will then determine the behaviour of the portfolio selection model. Precisely, each configuration θ is evaluated through a backtest on historical data D, which produces a set of financial performance measures. These outputs are aggregated into a utility function $U(\theta; D)$ that serves as the objective for optimisation. The problem can thus be written as

$$\max_{\theta \in \Theta} U(\theta; D).$$

This is fully consistent with the theoretical formulation in Chapter 2, where maximisation and minimisation were shown to be equivalent through the simple transformation $\max f(x) \equiv \min -f(x)$.

We will now specify the utility function. Each hyperparameter configuration θ induces a backtest that generates portfolio and benchmark trajectories, denoted by $A(t)$ and $M(t)$. These trajectories represent the evolution of an initial investment in the portfolio and in the benchmark. Based on them, the utility can be written as

$$U(\theta, D) = U(A, M, t) = \lambda_1 f_1 + \lambda_2 f_2 + \lambda_3 f_3 + \lambda_4 f_4 \quad \text{with} \quad \sum_{i=1}^{4} \lambda_i = 1,$$

where f_1 to f_4 are component functions specified in the following. The weights λ_i determine the relative importance of these four components. Since θ uniquely determines the trajectories of A and M through the backtest, the utility can equivalently be written either as a function of θ or of A, M and t. We will adopt the latter representation in the following.

Throughout the definitions of our f_i's, fractions of the form $A(t+\Delta)/A(t)$ or $M(t+\Delta)/M(t)$ denote total return factors over a time horizon Δ. Values between zero and one correspond to a loss, a value of one indicates no change, and values above one represent positive returns. This makes the formulation numerically stable and ensures intuitive interpretability across all components.

The first component f_1 captures long-term performance by comparing the annualised five-year return of the portfolio to that of the benchmark. The portfolio is supposed to outperform by at least 3.5%, which we impose as a threshold. This results in

$$f_1(A, M, t) = \left(\frac{A(t+5y)}{A(t)}\right)^{\frac{1}{5}} - \left(\frac{M(t+5y)}{M(t)}\right)^{\frac{1}{5}} - 0.035.$$

Note that the annualisation is achieved by the power term 1/5. The common '–1' term in annualised returns is left out here, as it cancels out in the relative comparison between the portfolio and the benchmark.

The second component f_2 compares the one-year return and is therefore given by

$$f_2(A, M, t) = \frac{A(t+1y)}{A(t)} - \frac{M(t+1y)}{M(t)}.$$

The third component f_3 focuses on downside protection by penalising excessive short-term underperformance. It ensures that the three-month annualised return does not fall more than 20% below the benchmark. It holds

$$f_3(A, M, t) = \left(\frac{A(t+3m)}{A(t)}\right)^4 - \left(\frac{M(t+3m)}{M(t)}\right)^4 + 0.2.$$

The final component f_4 evaluates crash resilience. We define an indicator function $\mathbf{1}_{\{\text{Crash in } [t,t+3y]\}}$ that equals one if a drawdown of at least 20% within a six-month period is detected during the next three years, and zero otherwise. Conditional on such a crash, f_4 integrates the annualised relative performance of the portfolio against the benchmark starting from the crash date over the subsequent two years. As the indicator function may equal one for multiple time points, a general approach would be to take an average over all corresponding integrals. In this thesis, we simplify the definition by selecting only the starting point of the largest drawdown, denoted by t_{crash}. This results in

$$f_4(A, M, t) = \mathbf{1}_{\{\text{Crash in [t, t+3y]}\}} \int_{t_{\text{crash}}}^{t_{\text{crash}}+2y} \left[\left(\frac{A(\tau)}{A(t_{\text{crash}})}\right)^{0.5} - \left(\frac{M(\tau)}{M(t_{\text{crash}})}\right)^{0.5}\right] d\tau.$$

This choice is justified, as averaging across all possible crash windows would often create overlapping events that differ only by a few trading days, leading to redundant evaluations without changing the economic interpretation of crash resilience. Furthermore, fixing a single crash date per fold ensures consistency. The horizon of three years for the indicator function, combined with a two-year integration window, ensures that f_4 is consistently defined within a five-year evaluation window, aligning with f_1. In this way, the entire utility function is defined over a common five-year horizon, which will be important for later analysis. Economically, f_4 takes positive values if the portfolio either experiences smaller losses than the benchmark during the drawdown or achieves a faster recovery afterwards.

Taken together, these four components ensure that the utility function rewards long-term growth, a short-term downside protection and robustness under market stress, thereby providing a balanced objective for the optimisation task. The weights λ_i allow the investor to shift this balance according to his risk and return preferences. In the following section, we specify the hyperparameters of the portfolio model and the search space over which this utility will be optimised.

3.2 Hyperparameters and Search Space

The hyperparameters not only determine the value of the utility function but also govern key aspects of the portfolio selection model such as rebalancing frequency, signal thresholds and financial ratio constraints. In contrast to model parameters that can be estimated directly from data, they must be set externally and tuned through the optimisation process. Because they interact in a non-linear way through the backtest, their joint specification has an important impact on the attainable performance and computational cost. In the following, we describe each hyperparameter together with its economic role, mathematical type and admissible range, and combine them into the mixed-type search space $\Theta \subset \mathbb{R}^d \times \mathbb{Z}^k$, where $\mathbb{Z}$ denotes the integers.

We will discuss the hyperparameters group by group, starting with those that govern portfolio size and rebalancing behaviour. The parameter `NPORT` controls the number of stocks held in the portfolio and thus the degree of diversification. In our setup it is allowed to vary between 20 and 50 positions, as empirical evidence shows that most of the benefits of risk reduction are already realised with 20 to 30 securities, while further diversification only yields to marginal improvements [9]. The parameter `NFREQ` specifies the rebalancing frequency in weeks, i.e., how often the portfolio composition is updated. We consider values between two and five weeks, which provides a good trade-off between taking into account changing signals and the transaction costs induced by the rebalancing. This range also aligns with the short-term components of the utility function, which would significantly worsen if rebalancing was too infrequent. Other parameters in this group are kept fixed, with `SLIPPAGE` set to 0.0075 to represent transaction costs and `INITCASH` fixed at 250,000 being the initial portfolio capital.

A further group of hyperparameters directly influences the composition of the portfolio by imposing filters and caps. On the filtering side, `MINCAPITALISATION` sets

a minimum market capitalisation, while `MINPE` and `MAXPE` restrict the stock universe through the corresponding minimum and maximum price-to-earnings ratios, and `MINGAIN` defines a minimum expected return. These restrictions ensure that only sufficiently liquid securities with robust fundamental characteristics enter the investable universe. On the allocation side, caps such as `LIQUIDITYCAP`, `SHARESCAP`, `SECTORCAP` and `UCITSCAP` limit exposures to individual stocks, sectors or illiquid assets, thereby controlling the concentration risk. Additionally, `MIN_POSITION_WEIGHT` and `RATIO_OF_STOCKS_IN_SECTOR` regulate the minimum size of holdings and the diversification within sectors. In addition, further fixed parameters are used to control downside risk and the portfolio weights, which are documented in an exemplary request file available in the project's GitHub repository.

A final set of hyperparameters concerns the expert sector weights assigned to the value and growth factors of each stock. These weights allow investors to express sector-specific views or expert knowledge by increasing or reducing the relative importance of each sector in either the Value or Growth dimension. For each of the 20 sectors s, the raw value and growth signals are multiplied by the respective weights w_s^V and w_s^G, which subsequently enter the backtesting system. This will contribute to construct a score for each stock that in turn leads to a ranking of them, which will determine the portfolio composition. In the request file, these hyperparameters appear as `VALUE_EXPERTWGHT_*` and `GROWTH_EXPERTWGHT_*`, one pair for each sector. They are treated as continuous hyperparameters bounded between 0.01 and 0.99, so values close to zero strongly diminish the importance of a sector without eliminating it entirely, while values close to one maintain its full influence. The list of all sectors can be found in the request file available in the project's GitHub repository.

Table 3.1 summarises the hyperparameters that we want to optimise. Formally, the search space can be expressed as

$$\Theta \;=\; [20, 50] \;\times\; \{2, 3, 4, 5\} \;\times\; [0.01, 0.99]^{40} \;\subset\; \mathbb{Z}^2 \times \mathbb{R}^{40},$$

where the first term represents portfolio size, the second the rebalancing frequency, and the third the 40 sector-specific weights for value and growth. This mixed-type and high-dimensional search space, with strong interactions between parameters, underlines the suitability of Bayesian optimisation in this setting. In the following section, we introduce the DRACUS backtesting system, which constructs a portfolio based on a given hyperparameter configuration and outputs the corresponding financial performance metrics.

Table 3.1: Hyperparameters to be optimised and their respective search domains. The sector weights comprise 20 dimensions, one per sector. The full sector list can be found in the request file available in the project's GitHub repository.

Name	Description	Type	Search space
`NPORT`	Portfolio size	integer	$[20, 50]$
`NFREQ`	Rebalancing interval	integer	$\{2, 3, 4, 5\}$
`VALUE_EXPERTWGHT_*`	Value tilt per sector	continuous	$[0.01, 0.99]$
`GROWTH_EXPERTWGHT_*`	Growth tilt per sector	continuous	$[0.01, 0.99]$

3.3 Data and DRACUS Interface

The evaluation of each hyperparameter configuration is carried out with the DRACUS backtesting system. DRACUS simulates the investment process over a given time horizon and thereby generates the empirical performance measures on which the optimisation is based. Its role in our framework is to serve as the link between our hyperparameter choices and the corresponding financial outcomes. Each run takes a request file as input and returns a set of output files that document the backtest in detail. From these outputs we obtain the portfolio and benchmark trajectories $A(t)$ and $M(t)$ which enter the utility function introduced in Section 3.1.

DRACUS builds on a broad investment universe that is divided into long, short and index definitions. In this thesis, we only consider the long universe, which corresponds to taking long positions in individual stocks. The short universe, by contrast, would allow for borrowing securities and profiting from price declines, while the index universe contains tradable index instruments. The long universe we work with essentially covers all U.S. equities between 1 January 2000 and 31 December 2024. For each stock, complete price histories are available together with the fundamental financial ratios introduced in Section 3.2, such as market capitalisation and valuation ratios. This ensures that the screening rules and factor definitions described earlier can be applied consistently over the full backtesting horizon.

From this stock universe, DRACUS constructs investable portfolios through a sequence of well-defined steps. After the initial filtering, each stock is assigned value and growth scores, which are normalised using transformations such as sigmoid functions or interquartile range scaling and then adjusted with sector-specific weights. These scores are subsequently combined into a single ranking that determines which stocks enter the portfolio. The portfolio construction itself is handled by a constrained

optimisation routine based on the Simplex method, which ensures that all caps and weight restrictions are satisfied while allocating capital across the top-ranked securities. In this way, DRACUS combines the application of the investment rules and the solving of an explicit optimisation problem to arrive at the final portfolio weights. The portfolio is then rebalanced at the frequency specified by `NFREQ`, and transaction costs are accounted for through the slippage parameter. The outcome of this process is a dynamically managed portfolio whose evolution is fully determined by the chosen hyperparameters and the underlying data.

The interaction with DRACUS follows a simple request–response logic. The user specifies a request file that encodes all hyperparameters, caps, filters and methodological choices described in Section 3.2. This request is passed to the backtesting system and archived under a timestamped identifier, ensuring reproducibility. For each run, DRACUS generates a dedicated subfolder within the `simulation-results/` directory, which contains three files. First, the archived `.request` file, which documents the exact configuration submitted to the engine. Second, an extensive `_logs.zip` archive that provides detailed time series, trading information and a lot more, from which the portfolio and benchmark trajectories $A(t)$ and $M(t)$ are reconstructed. These trajectories constitute the essential inputs for the utility function defined in Section 3.1. Third, the `resultsTable.csv` file offers a short overview of performance metrics and risk indicators such as Sharpe ratios or maximum drawdowns. Together, these files form a complete record of each simulation run.

In summary, the DRACUS setup ensures that each simulation run is transparent and traceable, with both raw time series and summary metrics stored in a structured format and directly linked to the corresponding hyperparameter configuration.

3.4 Summary and Conclusions

This chapter has specified the methodological setup of the optimisation problem. We introduced a utility function that evaluates portfolio outcomes along four dimensions, namely long-term growth, short-term competitiveness, downside protection and crash resilience, and formalised how investor preferences are incorporated through the weight parameters λ_i. We then characterised the hyperparameters that determine portfolio size, rebalancing frequency and sectoral tilts, and defined the resulting mixed-type search space Θ. Finally, we described the DRACUS backtesting system,

which takes the stock data as input, applies the screening and scoring rules, constructs a portfolio under constraints, and returns the corresponding performance metrics for each hyperparameter configuration.

Taken together, these components establish a coherent methodological foundation. The mapping $\theta \mapsto (A(t), M(t))$ demonstrates the black-box nature of the problem, as each configuration can be simulated but the underlying function is expensive to evaluate, non-linear and non-differentiable. These characteristics underline the need for Bayesian optimisation, which we will apply in the subsequent chapters, first on synthetic benchmark problems and then on the empirical optimisation of the portfolio model. At the same time, the methodology relies on certain simplifications, such as defining f_4 with the largest crash rather than averaging over multiple events, restricting the optimisation to a selected set of hyperparameters, and limiting the investment universe to long positions in U.S. equities. These aspects will be revisited in the discussion of limitations in Chapter 7.

Chapter 4

Experiment 1 – Framework Benchmarking on Synthetic Function

This chapter presents a controlled benchmark study comparing the optimisation frameworks Hyperopt and Optuna. It outlines the experimental design, evaluates their performance on a synthetic test function, and reflects on the implications for the subsequent financial optimisation problem.

4.1 Experimental Setup and Evaluation Metrics

To assess the empirical performance of the two optimisation frameworks Hyperopt and Optuna, we conduct a Bayesian optimisation on a synthetic and challenging objective function. The goal is to determine the more suitable framework for the financial model calibration tasks discussed in Chapter 5.

As objective function, we use the Ackley function, a standard benchmark in global optimisation due to its complex and non-convex landscape [15]. It is defined on $\mathbb{R}^d$ and given by

$$f(x) = -a \cdot \exp\left(-b \cdot \sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d} \cos(c \cdot x_i)\right) + a + \exp(1), \quad (4.1)$$

with $x \in \mathbb{R}^d$ and $a, b, c \in \mathbb{R}$. We adopt the standard parameterisation $a = 20$, $b = 0.2$, and $c = 2\pi$. The combination of exponential and cosine terms creates a surface with many local minima, while the function attains a unique global minimum

at $x^* = \mathbf{0}$ with $f(x^*) = 0$. Taken together, the landscape is characterised by a central global minimum surrounded by a regular pattern of local minima. Further away from the center, the function flattens out and offers little gradient information. This shape makes the problem particularly demanding for optimisation algorithms, especially in higher dimensions, as they might be trapped in one of the many local minima. Therefore, the challenge lies in balancing exploration and exploitation. The optimiser needs to explore the wider search space to locate the region of the global minimum, while also exploiting local information to refine the solution once promising regions have been identified. This challenge is structurally similar to the one faced in our financial model optimisation, which makes the Ackley function a well-suited benchmark.

In our baseline setting, we require a search space that is wide enough to represent a substantial challenge for the optimisers, and therefore define it as the hypercube $[-32.768, 32.768]^d$. We further consider $d = 10$ as the standard dimensionality, which already poses significant challenges for the acquisition strategy and surrogate model. Formally, the baseline optimisation problem is therefore

$$\min_{x \in \mathbb{R}^{10}} -20 \exp\left(-0.2\sqrt{\frac{1}{10}\sum_{i=1}^{10} x_i^2}\right) - \exp\left(\frac{1}{10}\sum_{i=1}^{10}\cos(2\pi x_i)\right) + 20 + \exp(1)$$

$$\text{s.t.} \quad x_i \in [-32.768,\ 32.768] \quad \text{for all } i = 1, \ldots, 10.$$

An additional experiment with lower dimensionality ($d = 5$) will later help to assess how each framework performs in a reduced dimensional setting. We will also test on more constrained search spaces ($[-2, 2]^d$) and noisy variants of the Ackley function to explore the robustness of each approach under more controlled or perturbed conditions. In the noisy variant, we add the Gaussian term $\varepsilon \sim \mathcal{N}(0,\ 0.01)$ to Equation (4.1). For all experiments, we fix the number of trials to 500 and repeat each setup across 20 different random seeds to ensure consistent configurations.

Both Hyperopt and Optuna implement Bayesian optimisation using the Tree-structured Parzen Estimator, as introduced in Section 2.3. While the underlying idea is the same, the frameworks differ in several implementation details. Optuna begins modelling after 10 randomly sampled startup trials, while Hyperopt requires 20 initial observations. These values are fixed and cannot be modified. Another key distinction is the thresholding strategy used to separate "good" and "bad" observations when fitting the current surrogate model. Hyperopt applies a fixed quantile threshold with

default value $\gamma = 0.15$, whereas Optuna adjusts this threshold dynamically based on search progress and model feedback. Additional features offered by Optuna, such as pruning or early stopping, are not providing advantages in this Ackley function setting, however, their relevance for our financial model optimisation will be discussed in Section 4.3.

To assess the performance of each framework, we evaluate three key metrics across repeated runs with different random seeds. First, we analyse the convergence behaviour by plotting the objective value as a function of the number of trials. This allows us to study how quickly each optimiser improves across trials and how smooth the convergence behaviour is. Second, we compare the distribution of best values achieved after a fixed budget of evaluations to measure robustness and final solution quality. Third, we plot the best objective value against wall-clock time, which shows how quickly each framework attains good solutions relative to its computational cost.

4.2 Results: Performance and Runtime

We now present the results of the baseline comparison between Hyperopt and Optuna on the 10-dimensional Ackley benchmark introduced in the previous Section. All results are averaged over 20 independent runs with different random seeds and a budget of 500 trials per run.

Figure 4.1 shows the mean convergence curves, with shaded areas representing one standard deviation across seeds. Optuna displays a steady and generally monotonic improvement over time, consistently lowering the average function value. Towards the end of the budget, the variance across runs increases noticeably, indicating greater variation in the achieved final values. In contrast to Optuna's convergence behaviour, Hyperopt's progress is less continuous. We can observe extended plateaus with only slight improvements, which are then followed by sudden and substantial jumps in performance. These jumps frequently bring Hyperopt's average evaluation close to the level reached by Optuna at that point, but are then again followed by stagnation until the next large improvement. This contrast between constant and sudden improvement reveals distinct exploration–exploitation patterns in the two implementations.

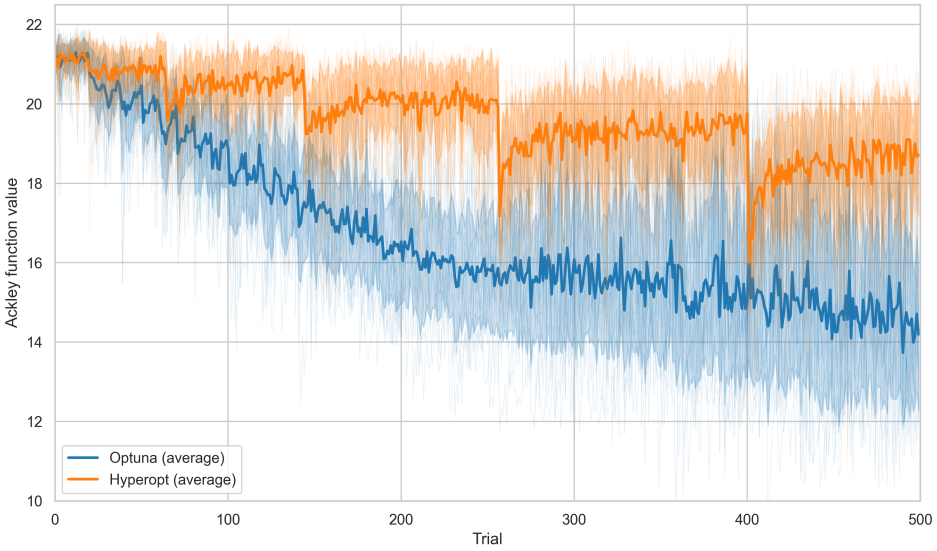

Figure 4.1: Convergence behaviour of Hyperopt and Optuna on the 10D Ackley benchmark over 500 trials, averaged across 20 independent runs. Lines indicate the mean objective value evaluated at each trial, with shaded areas showing ±1 standard deviation across seeds. Optuna exhibits a steady, near-monotonic improvement, whereas Hyperopt progresses in plateaus followed by sudden jumps.

Figure 4.2 compares the distribution of the best function values achieved by each framework across the 20 runs. Optuna attains substantially lower best values on average, which is approximately 11.5, whereas Hyperopt's average is close to 14.

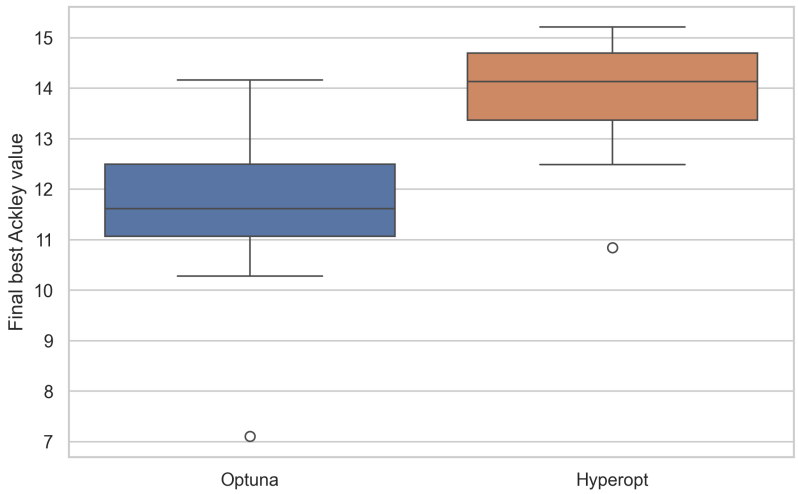

Figure 4.2: Distribution of the best Ackley function values after 500 trials across 20 independent runs per framework. Lower values indicate better optimisation performance at a fixed budget. Optuna's mean best value is ≈ 11.5 versus ≈ 14 for Hyperopt, demonstrating consistently better performance within the given trial budget.

For both frameworks, it is notable that they remain far from the global minimum of the Ackley function at $f(\mathbf{0}) = 0$ within the 500 evaluations. Optuna's lower mean nevertheless indicates a stronger capability to approach the global optimum within a predefined evaluation budget. Optuna also displays a single outlier with a best value of 7, representing a very strong result. Both frameworks exhibit a relatively similar spread of results, suggesting a comparable level of consistency across runs. Welch's t-test, conducted to test the null hypothesis that both best value distributions have the same mean, yielded a t-statistic of $t = -4.958$ with a p-value < 0.001. Hence, the null hypothesis can be rejected at any conventional significance level, and the difference in the distributions is statistically highly significant.

To assess optimisation progress in real time, Figure 4.3 shows the best function value found up to each point in cumulative wall-clock time. The x-axis shows the total runtime in seconds, while the y-axis reflects the smallest objective value identified so far. Optuna requires more computation time per trial, so as a result, a full run of 500 trials takes nearly one minute, while Hyperopt finishes after approximately 20 seconds. This explains the earlier termination of the Hyperopt curve.

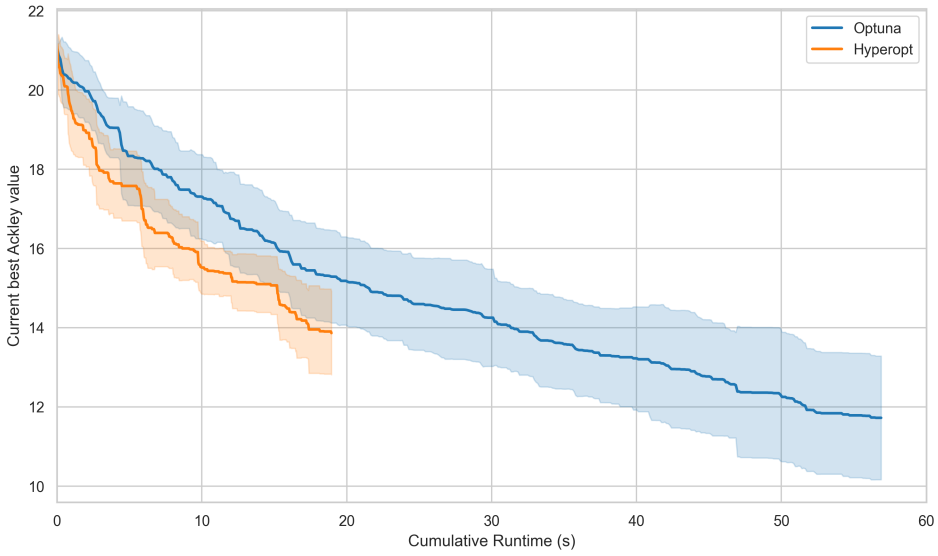

Figure 4.3: Current best Ackley function value over runtime, averaged across 20 independent runs per framework. Lines show the lowest value found up to each point, with shaded areas indicating ±1 standard deviation. Hyperopt completes all 500 trials in about 20 seconds, whereas Optuna requires nearly one minute. During the first 20 seconds, Hyperopt slightly outperforms Optuna.

Hyperopt slightly outperforms Optuna in terms of function value achieved per unit of time, but given that the gap is relatively small and that we are operating in the

range of seconds, this advantage is marginal. Interestingly, Hyperopt's curve exhibits a smooth convergence trajectory, which seems unexpected given its known pattern of extended plateaus followed by abrupt improvement jumps, seen in Figure 4.1.

To test the robustness of the previously observed differences, we repeated the experiment under three alternative setups, each modifying one parameter of the baseline configuration. The resulting convergence trajectories (top row) and final best-value distributions (bottom row) are presented in Figure 4.4. From left to right, the three panels correspond to a narrower search space $[-2, 2]^{10}$ instead of $[-32.768,\ 32.768]^{10}$, a lower-dimensional variant with $d = 5$ instead of 10, and a noisy 10D Ackley function with an additive Gaussian error term $\varepsilon \sim \mathcal{N}(0,\ 0.01)$. Across all settings, we observe convergence patterns consistent with those in the baseline scenario. Optuna continues to show relatively steady, near-monotonic improvements, while Hyperopt again alternates between plateaus and abrupt jumps in performance. Compared to the baseline scenario, Optuna's average convergence curve tends to flatten earlier. At the same time, the standard deviation bands widen, especially in the 5-dimensional and noise-injected setups, indicating that individual runs make significant improvements at different points in time.

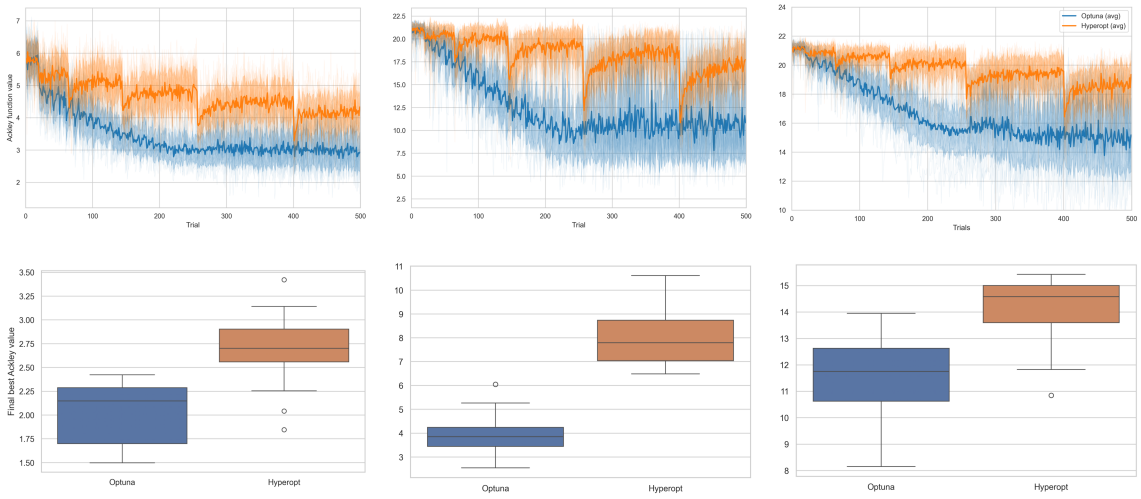

Figure 4.4: Comparison of Hyperopt and Optuna across three alternative setups compared to the baseline scenario: a reduced search space $[-2, 2]^{10}$ (left), a lower dimensionality $d = 5$ (center) and a noise-injected Ackley function (right). Each panel shows convergence plots (top row) and final best value distributions (bottom row), averaged over 20 independent runs. Optuna consistently achieves lower final values and smoother progress across settings, confirming the results from the baseline scenario.

While the mean trajectory suggests slower progress, the spread reflects that the runs eventually identify considerably better solutions at varying stages of the optimisation process. In all three cases, Optuna consistently outperforms Hyperopt in terms of final objective values, as indicated by the lower boxes and averages in all boxplots. Especially in the 5-dimensional case, Optuna reaches significantly lower best values, suggesting an increased advantage in simpler search landscapes. Welch's t-tests confirm the statistical significance of these differences, with test statistics of $t = -6.013$, $t = -12.646$, and $t = -5.591$ (left to right) and p-values well below 0.001 in all cases. Across all setups, we thus find that Optuna outperforms Hyperopt in terms of optimisation performance, maintaining its advantage even when the search landscape is altered. While no additional cumulative runtime plots are shown, the relative timing behaviour remains consistent with the baseline scenario, i.e., Hyperopt completes more trials in less time, but Optuna delivers better results per unit of evaluation budget. This motivates the following detailed investigation into the structural reasons behind these differences. After that, we will justify our final framework selection for our financial model optimisation.

4.3 Discussion and Practical Implications

While Section 4.2 provided a systematic empirical comparison, the observed performance gap between Optuna and Hyperopt raises the question of what drives these differences. Beyond differences in final objective values, the frameworks exhibit fundamentally distinct convergence behaviours. Optuna's is marked by gradual improvement, while Hyperopt's by stagnation and abrupt gains. To understand these behavioural traits, we now analyse a part of the internal algorithmic mechanisms.

A key mechanism influencing the search behaviour in TPE-based optimisation is the number of candidate samples considered per iteration. As introduced in Section 2.3, both Hyperopt and Optuna select the next point to evaluate by drawing samples from the density estimate $g(x)$ and choosing the one that maximises the acquisition criterion $l(x)/g(x)$. In Hyperopt, this process is controlled by the parameter `n_EI_candidates`, which determines how many of such candidates are sampled at each iteration. Its default value is 24. A larger number increases the likelihood of identifying a point that nearly maximises the acquisition function, thereby intensifying exploitation. Conversely, setting `n_EI_candidates` to 1 forces the algorithm to select a random draw from $g(x)$, which leads to greater exploration. Therefore, the

choice of this parameter represents a trade-off between exploration and exploitation, and its optimal setting is generally unknown and dependent on the problem.

This characteristic further interacts with Hyperopt's use of a fixed quantile threshold $\gamma = 0.15$ for separating "good" and "bad" observations, which are then used to fit the densities $l(x)$ and $g(x)$. If a large fraction of points is repeatedly sampled from the same region of the search space, which then are typically around the current global maximum of the acquisition function, then because of the fixed threshold, only a subset can be classified as "good". Over time, this leads to increasingly similar estimations for $l(x)$ and $g(x)$ in that region. As a result, the densities begin to overlap, flattening their ratio and reducing the incentive to continue sampling in that region. Once the acquisition surface reshapes, due to the sufficient change in the quantile composition, the model may shift focus and trigger a sudden improvement. This mechanism explains the characteristic convergence behaviour of Hyperopt, with long periods of stagnation followed by abrupt jumps. In contrast, Optuna overcomes this issue through dynamic thresholding, which adjusts the quantile threshold based on the observed performance landscape. This adaptive strategy helps maintain a sharper contrast between $l(x)$ and $g(x)$ throughout the optimisation, facilitating a smoother and more continuous search process.

Figure 4.5 illustrates the impact of setting `n_EI_candidates` to 1 in Hyperopt. As expected, the convergence curve becomes more continuous, since the acquisition function is no longer optimised over a pool of 24 candidates but simply evaluated at a single point drawn from $g(x)$ and thereby forces Hyperopt to update its model more frequently. This modification successfully avoids the plateau behaviour observed with the default setting and therefore supports our theoretical reasoning, i.e., lower values of `n_EI_candidates` increase exploration and help escape local focus. Nonetheless, the overall performance remains inferior to Optuna, which again converges to better final values. This residual performance gap is most notably explained by Optuna's dynamic thresholding mechanism.

Beyond the discussed mechanisms, Optuna offers additional practical advantages over Hyperopt, such as a more flexible interface design, better support for early stopping and pruning and built-in visualisation and diagnostic tools. While these features are not exploited in the synthetic Ackley benchmark experiment, they become valuable in computationally expensive and noisy problems. Additionally, it is worth noting that the open-source version of Hyperopt is no longer actively maintained, which limits its long-term usability for evolving or large optimisation projects. Given the smoother

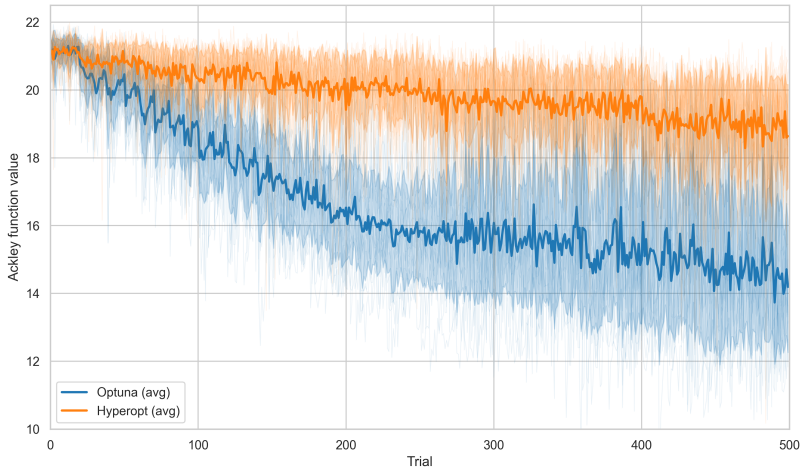

Figure 4.5: Effect of reducing `n_EI_candidates` from 24 to 1 in Hyperopt. In the baseline 10D Ackley setup, the curve becomes smoother and more continuous, without the extended plateaus and sudden jumps of the default setting. Nonetheless, overall performance remains inferior to Optuna.

and more stable convergence, the lower final function values and the availability of advanced functionalities, we conclude that Optuna is the more suitable framework for tuning our financial model in the subsequent chapters.

4.4 Summary and Conclusions

This chapter presented a systematic empirical comparison between the two Tree-structured Parzen Estimator frameworks Hyperopt and Optuna. Using the Ackley function as a synthetic benchmark, we found that Optuna consistently demonstrated better optimisation performance across various scenarios. In the baseline setting, Optuna not only achieved lower average and final best objective values, but also exhibited smoother convergence behaviour. In contrast, Hyperopt progressed through extended plateaus with abrupt improvement jumps, reaching comparable values at only a few points. These findings were confirmed in additional settings with reduced search space, lower dimensionality, and a noise-injected Ackley function. In terms of computational efficiency, Hyperopt showed a marginal advantage in runtime per trial, but Optuna's consistently better optimisation quality outweighed this runtime difference.

Beyond these empirical results, our analysis in Section 4.3 highlighted how the candidate sampling mechanism affects convergence behaviour. In Hyperopt, the fixed

setting of `n_EI_candidates` can bias the search towards certain regions of the search space, which in combination with its static quantile threshold may lead to stagnation phases. In contrast, Optuna's design, with adaptive thresholding and more flexible candidate selection, avoids this limitation and maintains smoother convergence. These considerations, together with Optuna's consistently better performance across all test scenarios, support our decision to adopt Optuna for the financial model calibration tasks in Chapter 5.

While this benchmark study was done on a synthetic test function that does not capture the full complexities of real-world applications, the results nevertheless generalise to such settings. The differences in candidate generation and thresholding strategies are likely to influence performance in real-world optimisation tasks especially when the objective function is noisy, expensive to evaluate, or structurally complex. These are typical characteristics of financial model calibration problems, where Bayesian optimisation must handle weak statistical signals and non-convex search landscapes. This provides the basis for moving from the benchmark study to the financial model in the next chapter.

Chapter 5

Experiment 2 – Batch Bayesian Optimisation in Financial Black-Box

This chapter extends the analysis to the proprietary financial model. It applies batch Bayesian Optimisation to the DRACUS backtesting environment, evaluates the impact on runtime efficiency and convergence, and discusses the practical implications for portfolio calibration.

5.1 Experimental Setup and Batch Strategy

In order to evaluate the impact of batch Bayesian Optimisation on the proprietary portfolio model, we first define the experimental setup. A first central aspect is the construction of a suitable cross-validation scheme. In contrast to many standard machine learning tasks, our portfolio optimisation is time-dependent. This means that the data, or more precisely the financial performance in earlier periods, cannot be used to validate later ones without risking information leakage. Hence, a form of time-series cross-validation is required, where training and testing windows are arranged chronologically so that only subsequent time periods are used for evaluation. Cross-validation is particularly important in this context, as the model is calibrated on limited and potentially non-stationary financial data, and performance needs to be assessed across multiple market regimes rather than relying on a single one. Our choice of training and testing folds is determined by two practical constraints, one arising from the backtesting system DRACUS and the other from the utility function. First, DRACUS requires approximately four years of historical input data before it

can construct a portfolio and generate valid performance numbers. Second, the utility function is defined on five-year horizons, so that each evaluation window must span at least this length. These conditions imply that each fold effectively consists of a four-year warm-up phase followed by a five-year period over which the utility is computed.

Table 5.1 summarises the fold structure. In the first fold, we use a training window from 2005 to 2010 and a test window from 2010 to 2015. Accounting for the necessary four-year warm-up period, the effective input ranges are 2001 to 2010 and 2006 to 2015, respectively. For longer training periods, as in Fold 2 and Fold 3, the window is split into consecutive five-year subperiods. The utility is computed separately on each of these inner folds and then averaged, ensuring that all parts of the training period contribute equally while preserving the five-year evaluation horizon imposed by the utility function. For instance, in the third fold the aggregated training utility is defined as

$$U_{\text{train}} = \frac{1}{3}\Big(U_{[2005,2010]} + U_{[2010,2015]} + U_{[2015,2020]}\Big),$$

where $U_{[a,b]}$ denotes the utility computed over the respective five-year subperiod $[a, b]$. Finally, the hyperparameter configuration achieving the highest average utility in the training folds is then validated out-of-sample on the designated test period.

Table 5.1: Time-series cross-validation folds used in the portfolio model optimisation. Each fold begins with a four-year warm-up phase required by DRACUS, followed by one or more five-year periods corresponding to the time horizon of the utility function. This structure leads to overlapping train and test windows.

Fold	**Train**	**Test**
1	2001–2010	2006–2015
2	2001–2015	2011–2020
3	2001–2020	2016–2025

We now turn to the specifications of the utility function. Since the portfolio universe is restricted to U.S. equities, we select the S&P 500 as the benchmark, as it constitutes the most widely recognised reference for the American market. For the implementation, two details are noteworthy. First, in f_1 we set the evaluation start value to the day on which DRACUS produces the first valid performance number, even if this occurs slightly after the nominal fold start date. In practice, this deviation amounts at most to only a few trading days and can be safely ignored in terms of economic relevance. Second, as discussed in Section 3.1, we fix the crash date in f_4 for each fold to

the start of the largest drawdown, rather than averaging across all two-year windows with a market drop of more than 20% within six months. This yields the following four crash dates: 19 July 2007, 21 July 2011, 20 September 2018, and 23 March 2020. These dates correspond to the starting point of well-documented stress periods in U.S. equity markets, which are the global financial crisis (2007), the Eurozone sovereign debt crisis (2011), the late-cycle equity correction preceding the COVID-19 pandemic (2018), and the sharp market crash at the outbreak of COVID-19 (2020). By fixing f_4 at these economically meaningful dates, we ensure that crash resilience is assessed consistently across folds and aligned with major real-world drawdowns.

The next step is to specify how batch Bayesian optimisation is implemented in our setting. We employ the Constant Liar strategy, introduced in Section 2.4, as it is the most compatible with a TPE framework. For the temporarily lied objective value, we choose the mean liar variant, which offers a balanced compromise between exploration and exploitation. Selecting the maximum or minimum liar would bias the search either too strongly towards exploitation or towards excessive exploration, whereas the mean liar tends to stabilise the acquisition surface and produce batches of consistent quality. To ensure comparability with the sequential baseline, we run two independent optimisations of equal budget, each comprising 20 trials. In the batch setting, this is achieved by conducting five Constant Liar iterations with a batch size of $q = 4$. Larger batch sizes would inflate the bias introduced by the lied values and reduce the diversity of the batch, while smaller sizes yield only limited wall-clock gains. Therefore, this choice ensures parallel efficiency and the quality of the surrogate updates. Both setups are implemented in Optuna using the TPE surrogate introduced in Section 2.3.

Due to the computational expense of the backtesting system, each optimisation run requires several hours of wall-clock time. While Folds 1 and 2 were fully processed, the third fold would have exceeded practical resource limits, as it involves an extended input horizon and hence substantially longer runtimes. The empirical comparison between sequential and batch optimisation is assessed along three complementary metrics, namely out-of-sample performance, convergence behaviour over the trial budget, and runtime efficiency in terms of best-so-far utility over wall-clock time.

5.2 Results: Performance, Convergence and Runtime

We begin by reporting the maximum utility achieved by each method under a trial budget of 20 evaluations. Table 5.2 shows, for each fold, the utility of the best configuration on the training set together with its corresponding out-of-sample performance. In both folds, sequential BO identifies slightly higher training utilities than Constant Liar BO, with 8.93 versus 7.17 in Fold 1 and 4.97 versus 4.20 in Fold 2. The generally lower values in Fold 2 indicate that it was harder to achieve high utility between 2015 and 2020. As expected, the out-of-sample performance U_{oos} is consistently below U_{train}, since portfolios optimised on past conditions rarely maintain the same utility when exposed to subsequent, and often very different, regimes. Between the two methods, Constant Liar achieves a relatively better out-of-sample performance in Fold 1, while performing slightly worse in Fold 2.

Table 5.2: Train utility (U_{train}) and out-of-sample utility (U_{oos}) of the best hyperparameter configuration, reported per fold and method. Across both folds, Constant Liar attains slightly lower training utilities than the sequential optimisation. In all cases, U_{oos} remains below U_{train}.

Period (Train → Test)	**Method**	$\mathbf{U}_{\text{train}}$	$\mathbf{U}_{\text{oos}}$
[2001–2010] → [2006–2015]	Sequential	8.93	-2.03
[2001–2010] → [2006–2015]	Constant Liar	7.17	-0.45
[2001–2015] → [2011–2020]	Sequential	4.97	-2.05
[2001–2015] → [2011–2020]	Constant Liar	4.20	-2.32

Because the utility value itself does not fully capture out-of-sample performance, we next examine the performance plots of the portfolios constructed from the best hyperparameter configurations identified in the training phase. Figures 5.1 and 5.2 plot the cumulative returns of the fund against the S&P 500 benchmark in the respective test periods. The shaded areas indicate the crash windows used for the calculation of our component f_4 in the utility function. In both folds, the optimised portfolios achieve a clear outperformance over the benchmark, with the effect being greater in the Constant Liar case.

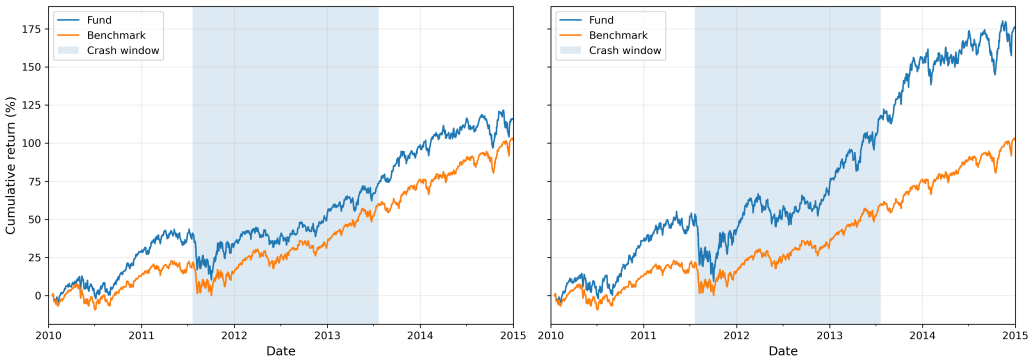

Figure 5.1: Out-of-sample performance measured by the cumulative return of the fund on Fold 1 (2010–2015). The sequential optimisation case is shown on the left and the batch Constant Liar case on the right. In both settings the fund outperforms the benchmark, with the advantage being greater under batch BO.

At the same time, the crash periods reveal steeper drawdowns for the funds compared to the benchmark. However, these larger losses occur after substantial prior outperformance, therefore the portfolios are naturally more exposed to greater downside movements during sharp corrections. This effect is particularly visible in Fold 2, where the downturn occurs towards the end of the test horizon after substantial gains have already accumulated, thereby amplifying the magnitude of the subsequent losses. Notably, in Fold 2 the fund recovers again rapidly after the downturn, keeping a clear lead over the benchmark towards the end of the test period.

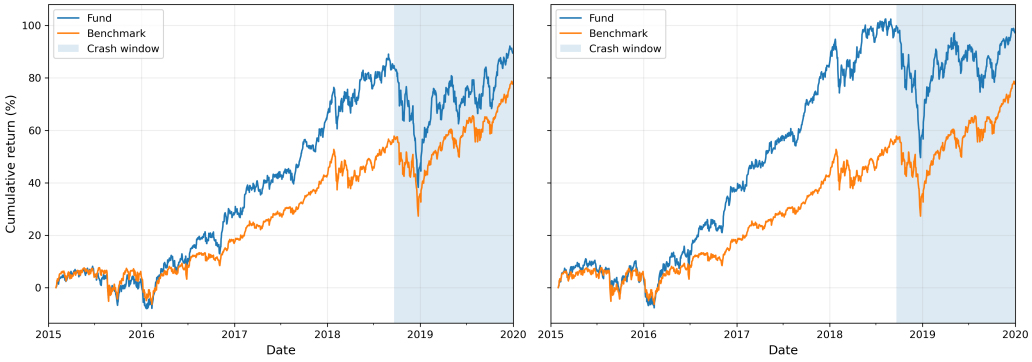

Figure 5.2: Out-of-sample performance measured by the cumulative return of the fund on Fold 2 (2015–2020). The sequential optimisation case is shown on the left and the batch Constant Liar case on the right. In both settings the fund outperforms the benchmark. In the batch case, the outperformance is stronger, but so is the subsequent drawdown during the crash.

We now examine the convergence of all methods in more detail. Figure 5.3 shows the per-trial utility trajectories for sequential BO and batch BO with Constant Liar,

displayed side-by-side for Fold 1 (left) and Fold 2 (right). The curves report the raw utility at each evaluation. In Fold 1, both methods improve over the budget, with sequential BO showing somewhat more variation and attaining a higher peak around the middle, whereas Constant Liar progresses more gradually. In Fold 2, sequential BO overtakes after mid-budget and remains higher towards the end, while Constant Liar is steadier overall with one notable upward and one downward spike around the middle. Overall, the plot indicates that sequential search explores more aggressively, while Constant Liar tends to deliver more homogeneous outcomes within each batch and therefore also overall.

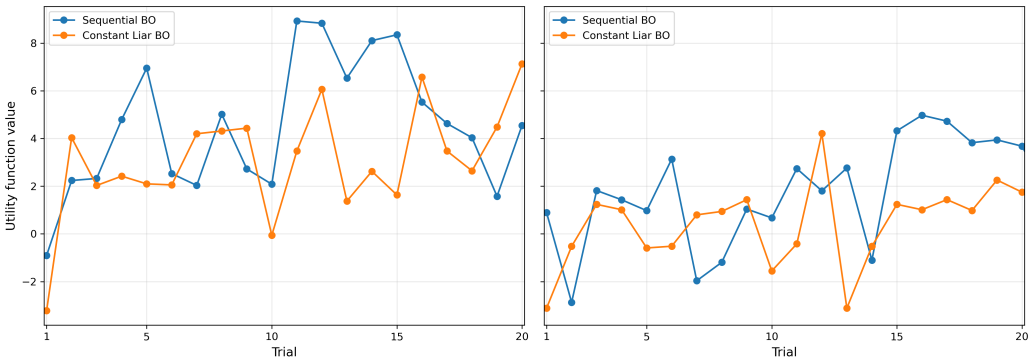

Figure 5.3: Per-trial utility values for sequential BO and Constant Liar BO. Fold 1 is shown on the left and Fold 2 on the right. The curves display the utility realised at each trial. Sequential BO produces more varied outcomes, while Constant Liar tends to deliver more homogeneous results within each batch and therefore also overall.

Finally, we compare the efficiency of the two optimisation strategies by relating the wall-clock runtime to the best-so-far utility. Figure 5.4 displays the trajectories for both folds. Since both approaches were run with the same number of trials, the batch method naturally terminates earlier, as several configurations are evaluated in parallel. In Fold 1, sequential BO eventually discovers the highest overall utility, but Constant Liar BO reaches competitive values much faster. The effect is even clearer in Fold 2, where the batch approach identifies a strong configuration within roughly two hours, whereas sequential BO only surpasses this level after five hours of computation. This illustrates how batch BO can accelerate the search for promising solutions in practical settings with limited time budgets. The average runtime per trial further highlights this advantage, amounting to 13.6 minutes (Fold 1) and 18.1 minutes (Fold 2) in the sequential case, compared to only 5.9 and 6.8 minutes under Constant Liar BO.

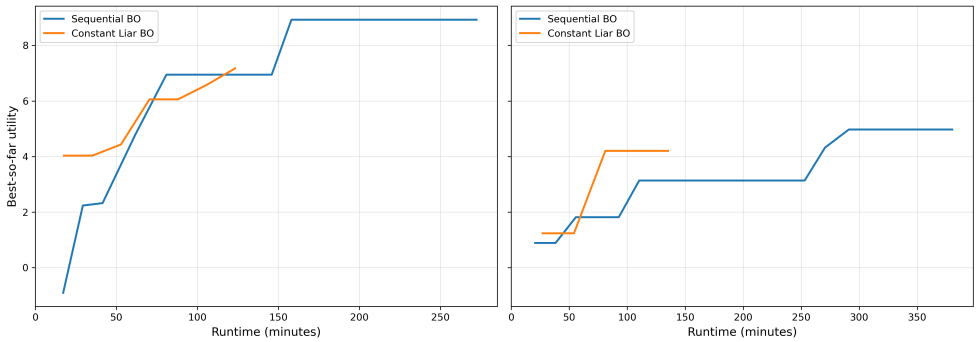

Figure 5.4: Runtime comparison of sequential and batch BO in terms of best-so-far utility. Results are shown for Fold 1 (left) and Fold 2 (right). Since both methods use equal trial rather than equal time budgets, the curves end at different horizons. In Fold 1, Constant Liar attains stronger utility early, while the sequential run only surpasses it later. In Fold 2, Constant Liar reaches a high utility much earlier, which is exceeded by the sequential run only after a substantially longer runtime.

Taken together, the results indicate that batch BO with the Constant Liar strategy offers a more favourable balance between runtime efficiency and solution quality. While sequential BO occasionally reaches higher peaks, the batch method identifies competitive configurations much faster and maintains a clear advantage in scenarios with tight time budgets. Therefore, we now take a closer look at the best-performing configurations found by Constant Liar BO. The corresponding hyperparameters and top three sector exposures are reported in Table 5.3.

Table 5.3: Best hyperparameters and top three sector exposures for the Constant Liar cases. Across both folds, Distribution and Health Technology have high Growth weights, while Energy and Commercial Services dominate on the Value side.

Case	**NPORT**	**NFREQ**	**Growth**	**Value**
Fold 1	33	3	Distribution (97.6%) Health Tech (95.8%) Finance (84.8%)	Energy (96.1%) Commercial Services (89.1%) Transportation (78.4%)
Fold 2	43	4	Distribution (98.3%) Technology (97.6%) Health Tech (89.3%)	Energy (96.0%) Commercial Services (95.3%) Tech Services (95.2%)

In both folds, the selected portfolio sizes of 33 and 43 stocks fall in the middle of the permitted range of 20 to 50, consistent with the idea that sufficient diversification is achieved at intermediate values. Larger portfolios would primarily increase trading costs without offering substantial gains in performance or risk diversification. Re-

garding sector tilts, the top exposures concentrate on a few key industries. On the Growth side, Distribution and Health Technology appear in both folds with very high expert weights, while Energy and Commercial Services have in both cases the highest Value weights. These overlaps suggest that certain sectors play a central role across folds.

5.3 Discussion and Practical Implications

We now turn to a critical discussion of the experimental results, focusing on the strengths and limitations of the optimisation framework as well as the implications for portfolio calibration in practice. A first point concerns the utility function component f_4, which proved to be disproportionately influential in determining the training and out-of-sample utility. By construction, f_4 evaluates crash resilience by integrating the relative performance of the portfolio against the benchmark during the crash and its subsequent two-year recovery. In practice, we observe that values of f_4 typically lie between -20 and 20, explaining the high utility values. In contrast, the components f_1 to f_3 mostly fall within a range of 0 to 1, as they are based on total return factors over fixed horizons. As a result, maximising the overall utility has effectively corresponded to maximising f_4.

As seen in the performance plots, this dominance has not necessarily resulted in poor portfolio outcomes. The corresponding funds perform reasonably well both in-sample and out-of-sample, in some cases even achieving clear benchmark outperformance. The real issue lies in the interpretability of the utility value itself, as it does not reliably reflect the actual fund performance. This problem is further reinforced by the fact that f_4 is highly sensitive to exact timing. If the benchmark or the fund drops even a single day before the predefined crash window begins, the resulting value of f_4 becomes distorted and fails to capture crash resilience as intended. It should also be acknowledged that fixing a single crash date per fold, instead of averaging across all two-year windows with a market drop of more than 20% within six months, represents a methodological limitation. The distortions caused by f_4 could have been diminished by empirically normalising its values, for example, by scaling them to bring the range closer to that of f_1 to f_3, or by reducing its influence through a smaller weighting parameter λ_4. Both approaches would, however, require more empirical evidence on the typical range of f_4 before they could be justified and applied in practice. Overall, the funds still delivered good results, but refining the utility function along these lines

could potentially yield more interpretable results and more robust portfolio outcomes.

We now examine the out-of-sample performance in more detail, as it constitutes the main criterion for evaluating whether the optimisation framework can be applied in practice. In our experiments, the funds generally outperformed the benchmark, but the results also reveal important shortcomings. In Fold 2, for example, the portfolio achieved strong gains over most of the horizon but then experienced a steep drawdown during the crash period. The realised gain of an investor would therefore have depended heavily on the entry point. While early investors would still have a cumulative outperformance, those entering just before the downturn would have faced substantial losses. This sensitivity to timing shows that the model does not yet deliver robust out-of-sample results across market conditions and investment horizons. One possible way to improve the model could be to calculate the utility on a rolling basis and average the outcomes, especially since the utility function components already address performance over the distinct horizons of three months, one year and five years. In principle, this would reduce the dependence on specific time windows and yield a more stable performance measure.

From a practical perspective, the comparison between sequential and batch optimisation is particularly relevant. In theory, sequential BO can achieve higher peak utilities if allowed to run for a sufficiently long time, but in realistic settings, the time budget for portfolio calibration is strictly limited. What matters most in practice is the speed at which competitive solutions are identified. Our experiments show that the Constant Liar batch strategy consistently delivers promising configurations much faster, while sequential BO requires substantially more computation time to reach similar levels of performance. For an institutional asset manager, where computational resources translate directly into costs and delayed decisions, the advantage of reaching solutions faster is critical. Batch optimisation, therefore, represents the better approach for deployment in practice.

Overall, the discussion shows both the strengths and the open issues of our optimisation framework. The methods can find competitive portfolios and batch BO clearly improves efficiency, but the definition of the utility function and the robustness of out-of-sample results remain challenges. In the next section, we summarise the main findings and draw the conclusions of this study.

5.4 Summary and Conclusions

This chapter investigated the application of batch Bayesian optimisation to the portfolio model. We compared the Constant Liar strategy against sequential Bayesian optimisation across multiple folds. The results showed that batch optimisation achieved utility values close to the sequential approach, but identified promising configurations way more quickly. Out-of-sample tests demonstrated that both methods found hyperparameter combinations that led to portfolios outperforming the benchmark. We concluded that the batch setting offered a more favourable trade-off between runtime efficiency and solution quality.

From a methodological perspective, these findings show that batch Bayesian optimisation can provide practically useful results when backtests are computationally expensive. The ability to identify strong configurations with fewer sequential steps is particularly relevant for applications where time and resources are limited. At the same time, several limitations must be acknowledged. The experiments were restricted to a relatively small number of evaluations, so potential differences in longer optimisation runs may not have been captured. Moreover, the crash-resilience component f_4 exerted a strong influence on the utility values, making the results sensitive to the fixed start date of crash periods. Furthermore, we found that out-of-sample performance varies substantially with the investor's entry date, which for now limits the practical use of the model. These aspects underline the need for robustness checks and sensitivity analyses, which we address in the following chapter.

Chapter 6

Experiment 3 – Sensitivity and Robustness Analysis

This chapter investigates the sensitivity and robustness of the obtained solutions. It analyses hyperparameter importance, assesses the stability of optimal configurations under perturbations, and outlines the implications for the interpretability and reliability of the optimisation model and Bayesian Optimisation in finance.

6.1 Experimental Setup for Robustness Analysis

In Bayesian Optimisation, the identification of a single best-performing configuration θ^* is only the starting point. For practical usability in financial settings, it is equally important to assess how robust such a solution is to small perturbations, methodological choices, or changing economic conditions. Robustness and sensitivity analyses therefore complement pure optimisation results by addressing the questions of which hyperparameters drive the performance most strongly and how stable the resulting allocations are under local or structural changes. From an economic perspective, these diagnostics are crucial, since investors require confidence that optimal configurations are not fragile outcomes of the optimisation process but represent strategies that remain effective under realistic variations in model inputs and market regimes. To address these issues, we design a set of four complementary experiments that we will introduce in the following.

The first experiment uses Optuna's built-in feature importance functionality to quan-

tify the relative contribution of each hyperparameter to the overall optimisation performance. The method estimates importance scores, leading to a ranking of the hyperparameters, by measuring the reduction in the objective value variance that can be attributed to each parameter. From an economic perspective, the main purpose of this analysis is to identify which sector weights have the strongest influence on the utility. Knowing which sectors matter most shows where expert weights need particular accuracy, as slight inaccuracies in these inputs can have a disproportionate impact on performance. The importance analysis therefore serves as a diagnostic tool to distinguish between structurally influential and less critical parameters.

The second experiment assesses local stability around the best configuration θ^*. Specifically, we add small bounded adjustments $\delta \in [-0.02, 0.02]$ to the sector-weight parameters, followed by a simple re-normalisation to ensure that the weights remain valid. We then evaluate the resulting utility change given by

$$\Delta U(\delta) = U(\theta^* + \delta) - U(\theta^*) \quad \text{and} \quad \Delta f_i(\delta) = f_i(\theta^* + \delta) - f_i(\theta^*), \quad i = 1, 2, 3, 4.$$

Local robustness is formally reached if the induced deviation remains bounded, i.e., if $|\Delta U(\delta)| \leq \varepsilon$ for a tolerance level $\varepsilon > 0$. If this condition holds, the perturbed configurations can be considered locally stable and do not substantially alter the optimised utility.

The third experiment examines the sensitivity of the optimised configuration to changes in the utility weights λ_i. Concretely, we apply small perturbations of ± 0.05 to each weight λ_i while keeping $\sum_{i=1}^{4} \lambda_i = 1$. We then evaluate the resulting change in utility at our best configuration θ^*. Formally, this can be expressed as

$$\Delta U(\theta^*) = \Delta\lambda_1 \cdot f_1(\theta^*) + \Delta\lambda_2 \cdot f_2(\theta^*) + \Delta\lambda_3 \cdot f_3(\theta^*) + \Delta\lambda_4 \cdot f_4(\theta^*),$$

where $\Delta\lambda_i$ denotes the change applied to λ_i. This setup serves to assess how sensitive the optimised solution is to variations in the underlying investor preference specifications.

Finally, the fourth experiment evaluates out-of-sample robustness by testing whether configurations optimised on one fold generalise to other folds. Concretely, we take the best configuration θ^* of every training fold and evaluate its utility on all remaining folds. This design allows us to check whether the optimised solutions capture structural features of the portfolio model that remain effective across different market regimes, rather than being specific to one time period. Robustness in this sense

requires that an optimal configuration keeps a high utility value.

Taken together, these four experiments provide a complementary set of diagnostics to evaluate robustness from different perspectives, which are the global parameter importance, local stability around the optimum, preference shifts in the utility specification, and a generalised out-of-sample performance.

6.2 Results: Hyperparameter Importance and Stability

We now present the empirical results of the robustness experiments, conducted for the Constant Liar batch configurations. Figure 6.1 shows the estimated hyperparameter importance for Fold 1 and Fold 2, with importance measured as the percentage contribution to the explained variance of the objective value. Bars in blue correspond to Growth expert weights, while bars in orange represent Value expert weights. In both folds, the distribution is highly concentrated, with the top three sectors already accounting for more than 45% of the overall importance. In Fold 1, Technology Services (17.9%), Energy Minerals (16.4%), and Health Technology (11.6%) form the leading group, while the remaining sectors contribute less than 9% each.

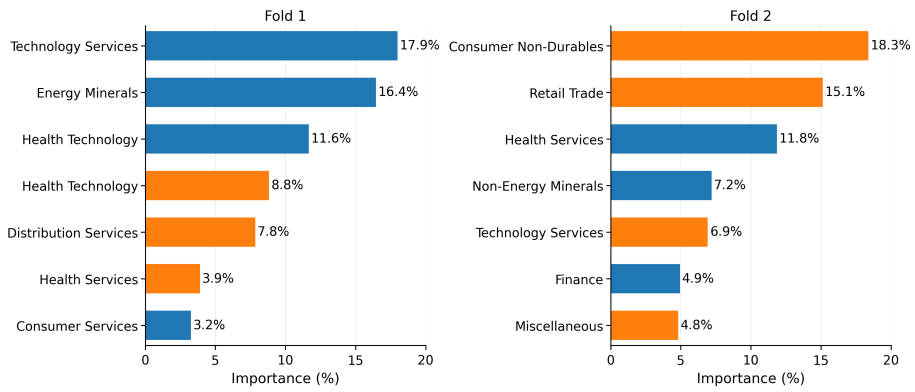

Figure 6.1: Hyperparameter importance scores estimated by Optuna's built-in functionality for Fold 1 (left) and Fold 2 (right). Importance values sum to 100%, with blue bars indicating Growth and orange bars Value expert weights. In both folds, the top three sectors already account for more than 45% of the total importance, demonstrating a concentration of influence among few parameters. Furthermore, we observe a shift from Growth weights in Fold 1 to Value weights in Fold 2, indicating that the relative role of Growth versus Value varies across folds.

Fold 2 exhibits a similar concentration, with Consumer Non-Durables (18.3%), Retail Trade (15.1%), and Health Services (11.8%) dominating the ranking. When aggregating the importance by factor dimension, Fold 1 is clearly dominated by Growth weights, which together account for 66.8% of the total importance. In contrast, Fold 2 shows the opposite pattern, with Value exposures contributing more than 63.4%, indicating that the relative importance of Growth versus Value weights varies substantially across folds. In both folds, more than 20 parameters have importance values below 1%, further underlining that only a small subset of hyperparameters drives the majority of the explained variance.

Table 6.1 reports the utilities obtained after modifying the sector weights of the optimal configurations under random perturbations $\delta \in [-0.02, 0.02]$. In Fold 1, all perturbed values lie below the training optimum of 7.17, with a maximum deviation of 0.66 and an average perturbed utility of 6.78. This indicates that the identified configuration is locally near-optimal, as normally small perturbations systematically reduce performance. In Fold 2, by contrast, four out of five perturbed outcomes are above the training optimum of 4.20, yielding an average perturbed utility of 4.40. This suggests that the optimisation has not yet converged to a truly optimal configuration in this case, potentially due to the limited number of trials. Overall, the results show that while Fold 1 exhibits local stability, Fold 2 remains more sensitive and would benefit from further exploitation.

Table 6.1: Utilities obtained after applying random perturbations to the sector weights of the optimal configuration. Reported are the training utility U_{train}, the five perturbed outcomes, and their mean. In Fold 1 all perturbed utilities lie below the baseline, indicating local stability, whereas in Fold 2 most perturbed values exceed it, suggesting that the configuration is not truly optimal.

Fold	$\mathbf{U}_{\text{train}}$	**Perturbed values**	**Mean**
Fold 1	7.17	6.86, 6.51, 6.68, 6.83, 6.98	6.78
Fold 2	4.20	4.32, 4.18, 4.63, 4.24, 4.63	4.40

Table 6.2 reports the results of the sensitivity analysis with respect to small shifts in the utility weights λ_i. For each fold, we display the base utility (U_{base}), the maximum and minimum perturbed values ($U_{\max}$ and $U_{\min}$), the corresponding maximal symmetric deviation from the base (ΔU), and the mean absolute change across all variants (ΔU_{avg}). By construction, the deviations are symmetric, since adding $+0.05$ to one weight and subtracting it from another is equivalent to the reverse shift. In Fold 1, the base utility of 7.17 shifts within a range of ± 1.17, with a comparatively

small average absolute change of 0.29 across all variants. In Fold 2, the corresponding range is narrower at ±0.68, with an average change of 0.17. Overall, the perturbations of λ_i do not substantially affect the achieved utility, indicating that the results are robust to moderate changes in the underlying preference specification.

Table 6.2: Sensitivity analysis of the utility with respect to shifts of ±0.05 in the preference weights λ_i. Reported are U_{base}, the maximum and minimum perturbed utilities ($U_{\max}$ and $U_{\min}$), the symmetric deviation (ΔU), and the mean absolute change (ΔU_{avg}). Across both folds, deviations remain small, indicating robustness to moderate changes in preferences.

Fold	$\mathbf{U}_{\text{base}}$	$\mathbf{U}_{\max}$	$\mathbf{U}_{\min}$	$\boldsymbol{\Delta}\mathbf{U}$	$\boldsymbol{\Delta}\mathbf{U}_{\text{avg}}$
Fold 1	7.17	8.34	6.00	1.17	0.29
Fold 2	4.20	4.89	3.52	0.68	0.17

Table 6.3 reports the cross-fold utilities of the batch-optimal configurations. In addition to the designated out-of-sample folds, each configuration was also evaluated on the two other test periods. For Fold 1, this adds results for 2015–2020 and 2020–2025, while for Fold 2 additional values are obtained for 2010–2015 and 2020–2025. The new cross-evaluations reveal several patterns. First, the Fold 2 configuration achieves a positive out-of-sample utility of 1.22 in 2010–2015, which is higher than the −0.45 obtained by the Fold 1 configuration on its designated test period. This difference is not surprising, since the 2010–2015 period is already included in the longer training horizon of Fold 2, so the configuration naturally benefits from overlap with its own training data. In the subsequent period, both configurations yield comparably weak utilities around −2.9 and −2.3, while in 2020–2025 both achieve very high values above 15. The latter, however, must be interpreted with caution, since utility values, as we have seen earlier, not always provide a reliable indicator of overall fund performance.

Table 6.3: Out-of-sample utilities when evaluating the batch-optimal configurations on all test folds rather than only the designated one. The columns indicate the test period on which the utility is calculated. Notably, both configurations reach very high utilities in the last fold (2020–2025), although utility values alone, as seen before, are not fully conclusive for assessing fund performance.

Best from	$\mathbf{U}_{\text{train}}$	**2010–2015**	**2015–2020**	**2020–2025**
Fold 1	7.17	−0.45	−2.95	15.57
Fold 2	4.20	1.22	−2.33	15.64

To provide a clearer picture, Figure 6.2 complements the table by plotting the cumulative fund performance against the benchmark in the non-designated folds. The left panel shows that the Fold 2 configuration strongly outperforms the benchmark in 2010–2015 at nearly every point in time. As mentioned above, this is not surprising given the overlap with the training period. In contrast, the middle panel indicates that the Fold 1 configuration tracks the benchmark very closely in 2015–2020, and a similar pattern is observed for both configurations in 2020–2025 (right panel). In the latter case, the funds experience larger losses before the crash date and therefore recover more strongly afterwards, which drives the high utility values, but this does not translate into genuine outperformance relative to the benchmark.

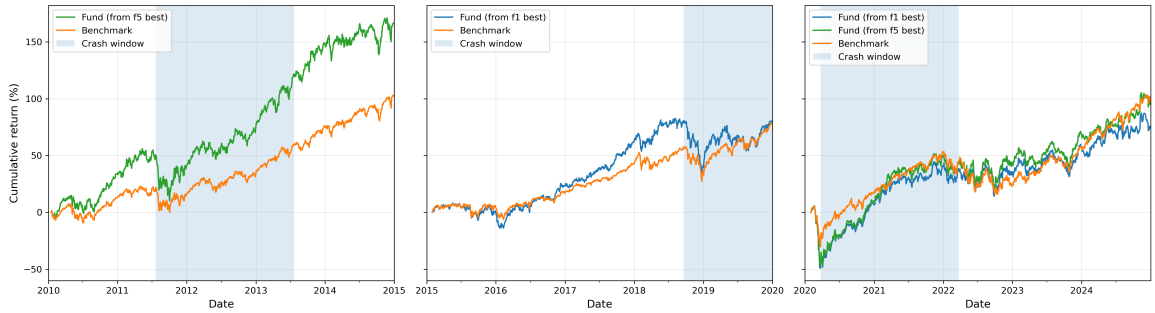

Figure 6.2: Cumulative fund performance of cross-fold configurations compared to the benchmark. The left panel shows the Fold 2 configuration applied to 2010–2015, the middle panel the Fold 1 configuration on 2015–2020, and the right panel both configurations on 2020–2025. While the Fold 2 configuration achieves consistent outperformance in the first period, the Fold 1 case and the final period track the benchmark more closely, highlighting discrepancies with the respective reported utility values.

Taken together, the results across all robustness experiments show that different diagnostics can lead to distinct perspectives on stability, showing the need to interpret parameter importance, sensitivity tests, preference shifts and out-of-sample performance in combination. We therefore turn to the discussion of these findings.

6.3 Discussion and Practical Implications

The previous section presented the robustness experiments in a purely descriptive way. In the following, we discuss their implications more broadly, combining methodological and economic perspectives. We first address the overall stability of the optimised solutions, then turn to sectoral patterns in the importance analysis and finally

consider the practical implications for portfolio management.

In the second experiment, where we modified the sector weights of the optimal configuration θ^* by a small δ, we observed only small changes in utility. This suggests that the solution is generally stable to such perturbations. Looking at the folds separately, in Fold 1 all perturbations led to lower utility values, which indicates a well-identified local minimum, while in Fold 2 most perturbations improved the outcome. This is consistent with the exploration–exploitation trade-off in Bayesian Optimisation, as due to the limited number of trials, the algorithm may prioritise the exploration of other regions instead of fully exploiting the neighbourhood of a good configuration. From an economic view, the result means that moderate revisions in expert sector weights only have a small impact on utility, which is a desirable property for implementation.

The robustness experiment, where we shifted the utility weights λ by ± 0.05, also demonstrates stability. The mean utility hardly changed in either fold, which indicates that the solution is not overly sensitive to a very specific preference vector. Economically, it means that the portfolios remain robust even if investor preferences shift moderately, for example due to changes in the risk behaviour of the portfolio manager or clients, the investment horizon, or an evolving market outlook. Such robustness is important, since in practice preferences are rarely fixed over time.

Our analysis of cross-fold transfers produced mixed results. The high out-of-sample utility of Fold 2 on 2010–2015 is expected since that period is part of its training horizon and therefore does not represent a strict out-of-sample test. Across the other folds, the configurations perform broadly in line with the benchmark, which means that there is no consistent outperformance but also no significant underperformance. This raises the question of how well the model will be able to outperform the benchmark in the future. In contrast, the very high utility values for 2020–2025 should be interpreted with caution, as we have seen in Chapter 5, they are mainly driven by the crash component f_4 and may give a misleading impression of the true robustness of the strategy.

A second perspective on robustness comes from the sectoral importance analysis. As we have seen, the distribution of parameter importance is highly concentrated, with the top three sectors already accounting for more than 45% of the explained variance. This raises the question of whether the sectors identified as most influential are economically plausible in the respective periods and factor tilts. For Fold 1, covering 2010–2015, the top three entries are Growth expert weights for Technology

Services, Energy Minerals and Health Technology. The strong weight of technology and healthcare is consistent with the macroeconomic environment of the time, which was characterised by a rapid growth in cloud computing, social media platforms and biotech innovation. The high importance of Energy Minerals is less straightforward. The sector was an important macroeconomic driver in the U.S. and experienced a sharp oil price collapse in late 2014 [32], which likely increased its statistical influence in our objective. In this sense, it may act more as a driver of risk than as a source of persistent return. In Fold 2, representing 2015–2020, the importance shifted towards Value expert weights in Consumer Non-Durables and Retail Trade and a Growth expert weight in Health Services. The importance of Consumer Non-Durables is plausible, as household consumption was a key driver of the U.S. economy in this period, supported by cheap credit and a strong labour market. Retail Trade also appears reasonable, although its importance may reflect the structural rise of e-commerce rather than traditional value exposures. The high ranking of Health Services as a Growth sector is more difficult to explain with market realities, since the sector is typically viewed as defensive and closer to a value profile. This suggests that not all importance results align perfectly with economic intuition and that some sectoral weights may arise from risk dynamics or limitations in the optimisation process.

Overall, the robustness experiments provide useful insights into the stability, sectoral drivers and practical implications of the optimisation outcomes. The following section summarises these findings and places them into the broader context of the thesis.

6.4 Summary and Conclusions

This chapter analysed the robustness of the optimisation outcomes across four complementary experiments. The importance analysis showed that only a few parameters explain a large share of the variation in utility, and the local sensitivity analysis demonstrated that small sector tilts around the optimum usually lead to only minor changes in performance. The experiments on utility weights confirmed that the results are stable under moderate preference shifts. Finally, the cross-fold evaluation revealed that while the configurations transfer reasonably well across regimes, the utility values may give a misleading impression and need to be complemented by actual performance checks.

From a broader perspective, the experiments indicate that Bayesian Optimisation

delivers solutions that are locally stable and robust to moderate changes in both expert inputs and investor preferences. This stability is encouraging for practical use, since views and objectives in portfolio management evolve continuously. At the same time, the concentration of importance in only a few sectors highlights the need for careful monitoring of these exposures, as misspecified weights in critical sectors can have disproportionate effects. The analysis also reveals limitations, in particular the relatively small sample size of trials and the sensitivity of the utility function to crash events, which restrict the significance of the conclusions. This is particularly relevant as the current best configuration identified in the experiments may still differ from the true optimum. Taken together, the robustness analysis confirms that BO-tuned portfolios are generally stable, while also underlining that robustness must be assessed across multiple dimensions rather than from utility values alone. These findings contribute to the overall assessment of Bayesian Optimisation in the concluding chapter.

Chapter 7

Conclusion and Future Work

This chapter concludes the thesis. It summarises the main findings and contributions, reflects on their implications for both research and practice, and outlines directions for future work.

7.1 Summary

This thesis aimed to advance the use of Bayesian Optimisation for the calibration of black-box portfolio selection models. Motivated by the high computational cost and complexity of financial backtests, the research aimed to evaluate which frameworks are most suitable for hyperparameter tuning, how parallelisation through batch optimisation affects efficiency, and whether BO-derived solutions remain robust and interpretable under perturbations. These guiding questions were addressed in three complementary investigations.

The first study benchmarked two widely used TPE-based frameworks, Hyperopt and Optuna, on the Ackley function. The experiments showed that Optuna consistently achieved lower objective values, displayed smoother convergence, and benefited from its adaptive thresholding strategy. In contrast, Hyperopt exhibited plateau behaviour with abrupt jumps in performance, a pattern explained by its fixed thresholding and candidate sampling scheme. Given Optuna's superior performance and continued maintenance, it was selected as the framework for the subsequent financial experiments.

The second study extended BO from the sequential to the batch setting within the

DRACUS backtesting environment. Applying the Constant Liar strategy with a moderate batch size, the study found that batch BO reduces runtime significantly without affecting convergence quality or the stability of the solutions. This finding underlines the practical advantage of parallel evaluation in financial calibration tasks, where each utility evaluation is highly computationally expensive.

The third study focused on robustness and sensitivity. By analysing the importance of sector-specific expert weights, the experiments revealed that a small subset of these weights drives much of the optimisation outcome, while the majority exert relatively minor influence. Perturbation tests further confirmed that optimal configurations remain locally stable under moderate changes in either the preference weights or sector weights, making the model robust for practical use.

Taken together, these results show that Optuna-based BO can be both efficient and robust in tuning complex portfolio models. The work addressed the three research questions set out at the beginning, demonstrating (i) clear framework differences in favour of Optuna, (ii) substantial efficiency gains from batch optimisation, and (iii) robust and interpretable outcomes in a realistic financial setting. The following section summarises these findings into the concrete scientific and practical contributions of this thesis.

7.2 Contributions

This thesis contributes to both the methodological literature on Bayesian Optimisation and to its practical application in quantitative finance. Each of the three experiments provides a distinct scientific advance together with an industry-relevant implication. These contributions are outlined in the following.

Experiment 1 provides a systematic comparison of two Tree-structured Parzen Estimator frameworks, Hyperopt and Optuna, under controlled benchmark conditions. On the scientific side, the study relates the performance differences to specific design choices in thresholding and candidate sampling, thereby offering methodological insight into how framework-level mechanisms influence optimisation outcomes. On the industry side, the findings give clear guidance on framework selection in computational finance. Optuna's smoother convergence and ongoing software support make it the more reliable choice for expensive financial backtests, where efficiency and maintainability are critical in practice.

Experiment 2 extends Bayesian Optimisation from the sequential to the batch setting within the DRACUS backtesting environment. The scientific contribution lies in showing how the Constant Liar strategy can be applied in a TPE-based framework and in confirming that parallel evaluations improve runtime without biasing the search process. The industry contribution lies in demonstrating that batch optimisation enables practitioners to reach competitive solutions much faster, which is highly valuable when calibration costs are high and quick adaptation to market changes is required.

Experiment 3 investigates the robustness and interpretability of BO-tuned configurations. Scientifically, the study combines hyperparameter importance analysis, perturbation tests, and cross-fold evaluations to highlight conditions under which solutions remain stable. This advances the understanding of reliability and interpretability in BO-driven financial optimisation. On the practical side, the findings show that the identified configurations are generally stable under moderate changes of the utility function and expert weights, and that only a small subset of sector weights has a strong impact. This allows practitioners to focus their attention on the exact estimation of the parameters that matter most and increases confidence that BO-calibrated models can be implemented reliably in practice.

7.3 Further Research Directions

While this thesis has demonstrated the effectiveness of Bayesian Optimisation in financial model calibration, several limitations and open questions remain. These provide opportunities for future research that can advance the methodological understanding of BO and its application in quantitative finance.

A first direction concerns methodological extensions of batch Bayesian Optimisation. In this thesis, the Constant Liar strategy was used as the most TPE-compatible approach, but several improvements are possible. New lying strategies could be developed that adapt dynamically to the optimisation phase, for example, favouring exploration in the early iterations and shifting towards exploitation later on. Alternative definitions of the liar value, such as quantile- or variance-based metrics, might also offer more balanced batches than the fixed mean used here. In addition, systematic experiments with different batch sizes could provide further insight into the trade-off between parallel efficiency and search quality. Beyond Constant Liar, future work

could focus on making Thompson Sampling more broadly compatible with TPE by designing partition-based posterior approximations. These developments could make batch BO more flexible in TPE-based settings and provide a better balance between efficiency and solution quality.

A second direction lies in the exploration of alternative surrogate models and frameworks. This thesis focused on Tree-structured Parzen Estimators, since they are widely used and naturally suited for mixed-type and high-dimensional search spaces. However, other non-Gaussian surrogates such as random forests, Bayesian neural networks, or ensemble methods could provide different and potentially better trade-offs between flexibility, sample efficiency, and scalability. A systematic comparison of these alternatives, especially in financial settings with black-box and noisy objectives, would extend the methodological basis of this work. Such studies could also clarify whether the advantages of Optuna over Hyperopt generalise to other surrogate families and whether hybrid approaches can further enhance robustness and efficiency in high-dimensional optimisation problems.

Another improvement opportunity concerns the design of the utility function. The crash-resilience component f_4 proved to be overly influential, occasionally making results more dependent on the timing and extent of the crisis period rather than on actual downside protection. In practice, a more robust definition of the objective is needed. Future work could investigate alternative formulations based on standardised risk-adjusted measures, such as the Conditional Value-at-Risk or drawdown-to-volatility metrics. A more balanced objective would provide more meaningful evaluations, i.e., enhance the interpretability and transferability of the optimisation results across different market regimes.

Finally, future work could extend the scope of applications and deployment. This analysis was restricted to U.S. equities, but extending the approach to other markets and asset classes such as bonds, commodities, or derivatives would provide a more comprehensive assessment of its practical value. Another promising direction is to develop BO variants for continuous re-calibration, which would align the method more closely with real-world trading environments. Together, these extensions would advance both the methodological foundations and the practical relevance of Bayesian Optimisation in quantitative finance.

Bibliography

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of KDD'19*, pages 2623–2631, 2019.

[2] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the Multi-armed Bandit Problem. *Machine Learning*, 47(2-3):235–256, 2002.

[3] Viktor Bengs and Oliver Grothe. Bayesian optimization for automated model selection in finance. *Expert Systems with Applications*, 184:115537, 2021.

[4] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, volume 24, pages 2546–2554, 2011.

[5] James Bergstra, Dan Yamins, and David D Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. *Journal of Machine Learning Research*, 13:281–305, 2013.

[6] 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.

[7] Clément Chevalier and David Ginsbourger. A fast computation of the multi-points expected improvement with applications in batch selection. In *Learning and Intelligent Optimization (LION 7)*, pages 59–69. Springer, 2013.

[8] Rama Cont and Nicholas Perkowski. Calibration of financial models with bayesian optimization. In *Handbook on Financial Econometrics and Statistics*. Springer, 2022.

[9] Edwin J. Elton and Martin J. Gruber. Risk reduction and portfolio size: An analytic solution. *The Journal of Business*, 50(4):415–437, 1977.

[10] David Eriksson, Michael Pearce, Jacob Gardner, Ryan D. Turner, and Matthias Poloczek. Scalable global optimization via local Bayesian optimization. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[11] Q. Gan, N. Ju, and T. Wang. Bayesian calibration of option pricing models. *Quantitative Finance*, 19(5):779–794, 2019.

[12] Jacob Gardner, Matt Kusner, Zhixiang Xu, Kilian Weinberger, and John Cunningham. Bayesian optimization with inequality constraints. In *Proceedings of the 31st International Conference on Machine Learning*, pages 937–945, 2014.

[13] David Ginsbourger, Rodolphe Le Riche, and Luc Carraro. Kriging is well-suited to parallelize optimization. In *Computational Intelligence in Expensive Optimization Problems*, pages 131–162. Springer, 2010.

[14] Javier González, Zhenwen Dai, Philipp Hennig, and Neil D Lawrence. Batch Bayesian optimization via local penalization. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 648–657, 2016.

[15] Nikolaus Hansen, Anne Auger, Steffen Finck, and Raymond Ros. Comparing continuous optimizers: COCO platform and benchmarking procedures. *arXiv preprint arXiv:1603.08785*, 2016.

[16] Philipp Hennig and Christian J Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13:1809–1837, 2012.

[17] José Miguel Hernández-Lobato, Yingzhen Li, Mark Rowland, Daniel Hernández-Lobato, Thang Bui, and Richard Turner. Parallel and distributed Thompson sampling. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[18] José Miguel Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. In *Advances in Neural Information Processing Systems*, pages 918–926, 2014.

[19] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International Conference on Learning and Intelligent Optimization*, pages 507–523. Springer, 2011.

[20] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 1998.

[21] Kirthevasan Kandasamy, Jeff Schneider, and Barnabás Póczos. High dimensional bayesian optimization and bandits via additive models. In *International Conference on Machine Learning*, pages 295–304, 2015.

[22] Harold J. Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86(1):97–106, 1964.

[23] X. Li, Y. Zhang, and Y. Jiang. Bayesian optimization for portfolio selection. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2020.

[24] Fengpei Liao, Mickaël Binois, and Henry P. Wynn. Adaptive quantile regression for bayesian optimization under noise. *Journal of the Royal Statistical Society: Series B*, 83(4):922–947, 2021.

[25] McKinsey & Company. Quantitative finance teams and compute resource allocation. Technical report, McKinsey & Company, 2024. Industry Report.

[26] Victor Picheny, David Ginsbourger, and Olivier Roustant. Quantile-based optimization of noisy computer experiments with tunable precision. *Technometrics*, 55(1):2–13, 2013.

[27] David W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. John Wiley & Sons, 2nd edition, 2015.

[28] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[29] W. Shen, H. Wang, and H. Jiang. CVaR-driven portfolio optimization using Bayesian optimization. *European Journal of Operational Research*, 282(1):255–267, 2020.

[30] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, London, 1986.

[31] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, volume 25, pages 2951–2959, 2012.

[32] U.S. Energy Information Administration (EIA). Crude oil prices fall sharply in late 2014. `https://www.eia.gov/todayinenergy/detail.php?id=19451`, 2015. Accessed: 2025-09-07.

[33] Zi Wang and Stefanie Jegelka. Max-value entropy search for efficient bayesian optimization. In *Proceedings of the 34th International Conference on Machine Learning*, pages 3627–3635, 2017.

[34] Ziyu Wang, Masrour Zoghi, Frank Hutter, David Matheson, and Nando de Freitas. Bayesian optimization in high dimensions via random embeddings. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, pages 1778–1784, 2013.

[35] Jianbin Wu and Peter I. Frazier. Efficient computation of the knowledge gradient for bayesian optimization. *Neural Computing and Applications*, 31:152–165, 2019.

[36] R. Xu and D. Huang. Bayesian optimization for algorithmic trading strategy calibration. *Journal of Financial Data Science*, 3(2):54–69, 2021.

[37] L. Zhang and J. Wang. Hyperparameter tuning for financial forecasting with Bayesian optimization. *Expert Systems with Applications*, 162:113891, 2020.