Journal of Artificial Intelligence Research 74 (2022) 1269-1349 Submitted 01/2022; published 07/2022

HEBO: Pushing The Limits of Sample-Efficient Hyperparameter Optimisation

Alexander I. Cowen-Rivers ALEXANDER.COWEN.RIVERS@HUAWEI.COM
(Corresponding author)
Huawei R&D
Technische Universität Darmstadt

Wenlong Lyu LVWENLONG2@HUAWEI.COM
Rasul Tutunov RASUL.TUTUNOV@HUAWEI.COM
Zhi Wang ZHIWOONG@163.COM
Antoine Grosnit ANTOINE.GROSNIT@HUAWEI.COM
Huawei R&D

Ryan Rhys Griffiths RRG27@CAM.AC.UK
University of Cambridge

Alexandre Max Maravel ALEXANDRE.MARAVEL@HUAWEI.COM
Hao Jianye HAOJIANYE@HUAWEI.COM
Huawei R&D

Jun Wang W.J@HUAWEI.COM
Huawei R&D
University College London

Jan Peters PETERS@IAS.TU-DARMSTADT.DE
Technische Universität Darmstadt

Haitham Bou-Ammar HAITHAM.AMMAR@HUAWEI.COM
Huawei R&D
University College London

Abstract

In this work we rigorously analyse assumptions inherent to black-box optimisation hyper-parameter tuning tasks. Our results on the Bayesmark benchmark indicate that heteroscedasticity and non-stationarity pose significant challenges for black-box optimisers. Based on these findings, we propose a Heteroscedastic and Evolutionary Bayesian Optimisation solver (HEBO). HEBO performs non-linear input and output warping, admits exact marginal log-likelihood optimisation and is robust to the values of learned parameters. We demonstrate HEBO's empirical efficacy on the NeurIPS 2020 Black-Box Optimisation challenge, where HEBO placed first. Upon further analysis, we observe that HEBO significantly outperforms existing black-box optimisers on 108 machine learning hyperparameter tuning tasks comprising the Bayesmark benchmark. Our findings indicate that the majority of hyper-parameter tuning tasks exhibit heteroscedasticity and non-stationarity, multi-objective acquisition ensembles with Pareto front solutions improve queried configurations, and robust acquisition maximisers afford empirical advantages relative to their non-robust counterparts. We hope these findings may serve as guiding principles for practitioners of Bayesian optimisation.

1. Introduction

Although achieving significant success across numerous applications (Bobadilla et al., 2013; Litjens et al., 2017; Fatima & Pasha, 2017; Kandasamy et al., 2018; Cowen-Rivers et al., 2020), the performance of machine learning models chiefly depends on the correct setting of hyper-parameters. As models grow larger and more complex, efficient and autonomous hyper-parameter tuning algorithms become crucial determinants of performance. A variety of methods from black-box and multi-fidelity optimisation (Kandasamy et al., 2017; Sen et al., 2018) have been adopted for hyperparameter tuning with varying degrees of success. Techniques such as Bayesian optimisation (BO), for example, enable sample efficiency (in terms of black-box evaluations) at the expense of high computational demands, while "unguided" bandit-based approaches can fail to converge (Falkner et al., 2018). Identifying such failure modes, the authors in (Falkner et al., 2018) built on (Li et al., 2017) and proposed a combination of bandits and BO that achieves the best of both worlds; fast convergence and computational scalability. More recently in the context of the 2020 NeurIPS competition on Black-Box Optimisation, many BO variants have been convincingly demonstrated to be superior to random search for the task of hyper-parameter tuning (Turner et al., 2021). Though impressive, such successes of BO and alternative black-box optimisers, belie a set of restrictive modelling and acquisition function assumptions. We begin by describing these assumptions.

Modelling Assumptions: A core determinant of BO performance is the set of data modelling assumptions required to specify an appropriate probabilistic model of the black-box objective (e.g., the choice of validation loss in hyper-parameter tuning tasks). The model should not only provide accurate point estimates, but should also maintain calibrated uncertainty estimates to guide exploration of the objective. Amongst many possible surrogates (Springenberg et al., 2016; Hutter et al., 2011), Gaussian processes (GPs) (Williams & Rasmussen, 1996) are the default choice due to their flexibility and sample efficiency. Growing interest in applications of Bayesian optimisation has catalysed engineering feats that enhance scalability and training efficiency of GP surrogates by exploiting graphical processing units (Knudde et al., 2017; Balandat et al., 2020).

Similar to any other framework, the correct specification of a GP model is dictated by the data modelling assumptions imposed by the user. For instance, a homoscedastic GP suffers from misspecification when required to model data with heteroscedastic noise whilst stationary GPs fail to track non-stationary targets. The aforementioned shortcomings are not unnatural across a range of real-world problems (Kersting et al., 2007; Griffiths et al., 2021a, 2021b) and hyper-parameter tuning of machine learning algorithms is no exception, as illustrated in our hypothesis tests of Section 3.2. Hence, even if one succeeds in improving computational efficiency, frequently-made assumptions such as homoscedasticity and stationarity can easily inhibit the performance of any BO-based hyper-parameter tuning algorithm. Despite the importance of these assumptions in practice, GPs that presume homoscedasticity and stationarity still constitute the most common choice of surrogate.

Acquisition Function & Optimiser Assumptions: Modelling choices such as those described above are not unique to the GP fitting procedure but rather transcend to other

steps in the BO algorithm. Precisely, given a model that adheres to some (or all) assumptions mentioned above, the second step involves maximising an acquisition function to query novel input locations that are then evaluated. Hence, practitioners introduce additional constraints relating to the category of optimisation variables and the choice of acquisition function. When it comes to variable categories, mainstream implementations (Knudde et al., 2017; Balandat et al., 2020) assume continuous domains and employ first and second-order optimisers such as LBFGS (Liu & Nocedal, 1989) and ADAM (Kingma & Ba, 2015) to propose query locations. Real-valued configurations cover but a subset of possible machine learning hyper-parameters rendering discrete variable categories out of scope, an example being the hidden layer size in deep networks. Moreover, from the point of view of acquisition functions, libraries tend to presuppose that one unique acquisition performs best in a given task, while research has shown that benefits that can arise from a combined solution (Shahriari et al., 2014, 2016; Lyu et al., 2018) as we demonstrate in Section 5.

Contributions: Having identified important modelling choices in BO, our goal in this paper is to provide empirical insight into the impact of modelling choice on empirical performance. As a case study, we consider best practices for hyper-parameter tuning. We wish for our findings to be applicable across a broad range of tasks and datasets, be attentive to the effect of random initialisation on algorithmic performance, and naturally, be reproducible. As such, we prefer to build on established benchmark packages, especially those that facilitate fast and scalable evaluations with multi-seeding protocols. To that end, we undertake our evaluation in 2140 experiments from 108 real-world problems from the UCI repository (Dua & Graff, 2017), which was also the testbed of choice for the NeurIPS 2020 Black-Box Optimisation challenge (Turner et al., 2021). Our findings point towards the following conclusions:

1. Hyper-parameter tuning tasks exhibit significant levels of heteroscedasticity and non-stationarity.

2. Input and output warping mitigate the effects of heteroscedasticity and non-stationarity giving rise to better performing tuning algorithms with higher mean and median performance across all 108 black-box functions under examination.

3. Individual acquisition functions tend to conflict in their solution (i.e., an optimum for one acquisition function can be a sub-optimal point for another and vice versa). Using a multi-objective formulation significantly improves performance;.

To verify our principal conclusions, we conduct additional ablation studies on our proposed solution method, Heteroscedastic and Evolutionary Bayesian Optimisation (HEBO) [1] which attempts to address the shortcomings identified in our analysis and placed first in the 2020 NeurIPS Black-Box Optimisation Challenge. We obtain a ranked order of importance for significant components of HEBO, finding that output warping, multi-objective acquisitions and input warping lead to the most significant improvements followed by robust acquisition function formulations.

[1] All code is made available at https://github.com/huawei-noah/HEBO.

2. Standard Design Choices in BO

As discussed earlier, the problem of hyper-parameter tuning can be framed as an instance of black-box optimisation:

$$\arg\max_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x}), \tag{1}$$

with $\boldsymbol{x}$ denoting a configuration choice, $\mathcal{X}$ a (potentially) mixed design space, and $f(\boldsymbol{x})$ a validation accuracy we wish to maximise. In this paper, we focus on BO as a solution concept for black-box problems of the form depicted in Equation 1. BO considers a sequential decision approach to the global optimisation of a black-box function $f : \mathcal{X} \to \mathbb{R}$ over a bounded input domain $\mathcal{X}$. At each decision round, i, the algorithm selects a collection of q inputs $\boldsymbol{x}_{1:q}^{(\text{new})} \in \mathcal{X}^q$ and observes values of the *black-box* function $\boldsymbol{y}_{1:q}^{(\text{new})} = f(\boldsymbol{x}_{1:q}^{(\text{new})})$. The goal is to rapidly approach the maximum $\boldsymbol{x}^\star = \arg\max_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})$. Since both $f(\cdot)$ and $\boldsymbol{x}^\star$ are unknown, solvers need to trade off exploitation and exploration during this search process.

To achieve this goal, BO algorithms operate in two steps. In the first, a Bayesian model is learned, while in the second an acquisition function determining new query locations is maximised. Next, we survey frequently-made assumptions in mainstream BO implementations and contemplate their implications for performance.

2.1 Modelling Assumptions

When black-boxes are real-valued, Gaussian processes (Rasmussen & Williams, 2006) are effective surrogates due to their flexibility and ability to maintain calibrated uncertainty estimates. In established implementations of BO, designers place GP priors on latent functions, $f(\cdot)$, which are fully specified through a mean function, $m(\boldsymbol{x})$, and a covariance function or kernel $k_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}')$ with $\boldsymbol{\theta} \in \mathbb{R}^p$ representing kernel hyper-parameters. The model specification is completed by defining a likelihood. Here, practitioners typically assume that observations y_l adhere to a Gaussian noise model such that $y_l = f(\boldsymbol{x}_l) + \epsilon_l$ where $\epsilon_l \sim \mathcal{N}(0, \sigma^2_{\text{noise}})$. This, in turn, generates a Gaussian likelihood of the form $y_l|\boldsymbol{x}_l \sim \mathcal{N}(f_l, \sigma^2_{\text{noise}})$ where we use f_l to denote $f(\boldsymbol{x}_l)$ with $f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), k_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}'))$. Additionally, a further design choice commonly made by practitioners is that the GP kernel is stationary, depending only on the norm between $\boldsymbol{x}$ and $\boldsymbol{x}'$, $||\boldsymbol{x} - \boldsymbol{x}'||$. From this exposition, we conclude two important modelling assumptions stated as *data stationarity* and *homoscedasticity of the noise distribution.* Where **homoscedasticity** implies a constant noise term σ^2_{noise}. **Heteroscedasticity** is usually harder to model as implies σ^2_{noise} is a function of the input: i.e., depending on the data, the noise changes around the mean. Of course, it is clear that there are significant differences between homoscedastic functions and heteroscedastic functions, and later we show indeed heteroscedastic functions require a different approach to optimise over than the typical homoscedastic (synthetic) functions usually researched in Bayesian Optimisation. If the true latent process does not adhere to these assumptions, the resultant model will be a poor approximation to the black-box. Realising the potential empirical implications of these modelling choices, we identify the first two questions addressed by this paper:

Q.I. Are hyper-parameter tuning tasks stationary?

Q.II. Are hyper-parameter tuning tasks homoscedastic?

In Section 3.2, we show that even amongst the simplest hyper-parameter tuning tasks, the null hypothesis may be rejected in the case of statistical hypothesis tests for heteroscedasticity and non-stationarity.

2.2 Acquisition Function & Optimisation Assumptions

Acquisition functions trade off exploration and exploitation by utilising statistics from the posterior $p_{\boldsymbol{\theta}}(f(\cdot)|\mathcal{D})$ with $\mathcal{D}$ denoting the data (hyper-parameter configurations as inputs and validation accuracy as outputs) collected so far. Under a GP surrogate with Gaussian-corrupted observations $y_\ell = f(\boldsymbol{x}_\ell) + \epsilon_\ell$ where $\epsilon_\ell \sim \mathcal{N}(0, \sigma^2)$, and given a data set $\mathcal{D} = \{\boldsymbol{x}, \boldsymbol{y}\}$, the joint distribution of $\mathcal{D}$ and an arbitrary set of input points $\boldsymbol{x}_{1:q}$ is given by

$$\left[\begin{array}{c} \boldsymbol{y} \\ f(\boldsymbol{x}_{1:q}) \end{array}\right] \Bigg| \, \boldsymbol{\theta} \sim \mathcal{N}\left(\left[\begin{array}{c} m(\boldsymbol{x}) \\ m(\boldsymbol{x}_{1:q}) \end{array}\right], \left[\begin{array}{cc} \boldsymbol{K}_{\boldsymbol{\theta}} + \sigma^2 \boldsymbol{I} & \boldsymbol{k}_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:q}) \\ \boldsymbol{k}_{\boldsymbol{\theta}}^{\intercal}(\boldsymbol{x}_{1:q}) & \boldsymbol{k}_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:q}, \boldsymbol{x}_{1:q}) \end{array}\right]\right),$$

where $\boldsymbol{K}_{\boldsymbol{\theta}} = \boldsymbol{K}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x})$ and $\boldsymbol{k}_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:q}) = \boldsymbol{k}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}_{1:q})$. From this joint distribution one can derive though marginalisation (Rasmussen & Williams, 2006) the posterior predictive $p(f(\boldsymbol{x}_{1:q})|\mathcal{D}) = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:q}), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:q}))$ with:

$$\begin{aligned} \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:q}) &= m(\boldsymbol{x}_{1:q}) + \boldsymbol{k}_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:q})^\top (\boldsymbol{K}_{\boldsymbol{\theta}} + \sigma^2 \boldsymbol{I})^{-1} (\boldsymbol{y} - m(\boldsymbol{x})) \\ \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:q}) &= \boldsymbol{K}_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:q}, \boldsymbol{x}_{1:q}) - \boldsymbol{k}_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:q})^\top (\boldsymbol{K}_{\boldsymbol{\theta}} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{k}_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:q}). \end{aligned}$$

As such we note that $p(f(\boldsymbol{x}_{1:q})|\mathcal{D}) = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:q}), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:q}))$. In this paper, we focus on three widely-used myopic acquisition functions which in a reparameterised form can be written as (Wilson et al., 2017):

Expected Improvement (EI):

$$\alpha^{\boldsymbol{\theta}}_{\mathrm{EI}}(\boldsymbol{x}_{1:q}|\mathcal{D}) = \mathbb{E}_{\mathrm{post.}}\left[\max_{j \in 1:q} \{\mathrm{ReLU}(f(\boldsymbol{x}_j) - f(\boldsymbol{x}^+))\}\right],$$

where the subscript 'post.' is the predictive posterior of a GP (Rasmussen & Williams, 2006), $\boldsymbol{x}_j$ is the j^{th} vector of $\boldsymbol{x}_{1:q}$, and $\boldsymbol{x}^+$ is the best performing input in the data so far.

Probability of Improvement (PI):

$$\alpha^{\boldsymbol{\theta}}_{\mathrm{PI}}(\boldsymbol{x}_{1:q}|\mathcal{D}) = \mathbb{E}_{\mathrm{post.}}\left[\max_{j \in 1:q} \{\mathbb{1}\{f(\boldsymbol{x}_j) - f(\boldsymbol{x}^+)\}\}\right],$$

where $\mathbb{1}\{\cdot\}$ is the left-continuous Heaviside step function.

Upper Confidence Bound (UCB):

$$\alpha^{\boldsymbol{\theta}}_{\mathrm{UCB}}(\boldsymbol{x}_j) = \mathbb{E}_{\mathrm{post.}}\left[\max_{j \in 1:q} \left\{\mu_{\boldsymbol{\theta}}(\boldsymbol{x}_j) + \sqrt{\beta\pi/2}|\gamma_{\boldsymbol{\theta}}(\boldsymbol{x}_j)|\right\}\right],$$

where $\mu_{\boldsymbol{\theta}}(\boldsymbol{x}_j)$ is the posterior mean of the predictive distribution and $\gamma_{\boldsymbol{\theta}}(\boldsymbol{x}_j) = f(\boldsymbol{x}_j) - \mu_{\boldsymbol{\theta}}(\boldsymbol{x}_j)$. When it comes to practicality, generic BO implementations make additional assumptions during the acquisition maximisation step. First, it is assumed that one of the aforementioned acquisitions works best for a specific task, and that the GP model is an accurate approximation to the black-box. However, when it comes to real-world applications, both of these assumptions are difficult to validate; the best-performing acquisition is challenging to identify upfront and GP models may easily be misspecified. With this in mind, we identify a third question that we wish to address:

Q.III. Can acquisition function solutions conflict in hyper-parameter tuning tasks?

In the following section, we affirm that acquisitions can conflict even on the simplest of hyper-parameter tuning tasks. Moreover, we show that a robust formulation to tackle misspecification of acquisition maximisation can improve overall performance (see Section 4.2.1).

3. Modelling Assumption Analysis

Before discussing the improvements afforded to BO via our solution method, we detail analyses conducted to answer questions (**Q.I.**, **Q.II.**, and **Q.III.**) posed in the previous section. Our analyses indicate:

A.I.: Even simple hyper-parameter tuning tasks exhibit significant heteroscedasticity.
A.II.: Even simple hyper-parameter tasks exhibit significant non-stationarity.
A.III.: Acquisition functions conflict in their optima, occasionally leading to opposing solutions.

Experiment Setting: We create a wide range of hyper-parameter tasks (108) across a variety of classification and regression problems. We use nine models, (e.g. multilayer perceptrons, support vector machines) and six datasets (two regression and four classification) from the UCI repository, and two metrics per dataset (such as negative log-likelihood or mean squared error). Each model possesses tuneable hyper-parameters, e.g. the number of hidden units of a neural network. The goal is to fit these hyper-parameters so as to maximise/minimise one of the specified metrics. Values of the black-box objective are stochastic with noise contributions originating from the train-test splits used to compute the losses. Experimentation was facilitated by the `Bayesmark`[2] package. Full hyper-parameter search spaces are defined in Table 2 and Table 3. [3].

Statistical Hypothesis Testing for Heteroscedasticity and Non-Stationarity: We describe here the statistical hypothesis tests we use to answer **Q.II.**. GP regression typically considers a conditional normal distribution of the observations $y|\cdot \sim \mathcal{N}(f(\cdot), \sigma^2(\cdot))$ and in most cases $\sigma(\cdot)^2$ is assumed to be constant, in which case the GP is termed homoscedastic. To assess whether the homoscedasticity assumption holds for the tasks under examination,

[2] https://github.com/uber/bayesmark
[3] It is these search spaces that are used by the random search baseline.

we make use of Levene's test and the Fligner-Killeen test. To give the reader intuition as to how we apply these tests, Levene's test asseses whether the variance is equal in two groups of data, assuming the data is normally distributed. I.e for a given task, given multiple evaluations of the black-box of two distinct hyperparameter sets, do the share the same variance (Homoscedasticity), or do their variances differ (Heteroscedasticity). Secondly, the Fligner-Killeen test is similarly a test for Homoscedasticity, however it is particularly useful when the data is non-normal. We refer the reader to the Appendix A for additional information regarding the tests.

To run these tests on a given task, we evaluate $k = 50$ distinct sets of hyperparameters $\{x_i\}_{1\leq i\leq k}$ for $n = 10$ times and obtain scores $\{Y_{ij}\}_{1\leq i\leq k, 1\leq j\leq n}$, where Y_{ij} is the j^{th} score observed when evaluating the i^{th} configuration. For $i = 1, \ldots, k$, let σ_i^2 denote the observed variance of $y|x_i$, then both Levene's test and the Fligner-Killeen test share the same null hypothesis of homoscedasticity:

$$H_0 : \sigma_1^2 = \cdots = \sigma_k^2.$$

In all 108 tests, we see a p-value significantly lower than 0.05 in 72 tasks using Levene's test, and in 73 tasks using Fligner-Killeen test. Such results (shown in detail in Appendix C) imply that at least 66% of the experimental tasks exhibit heteroscedastic behaviour.

Table 1: Hypothesis Testing for 108 tasks with respect to GP fit. In the table below we show, out of all 108 tasks, whether the GP fit (marginal log-likelihoods) was improved (Better) when either the Output transform or Input warping was added into the surrogate model, or was worse. Furthermore, we include significantly testing using the one sided t-test and detail how many tasks the GP fit was significantly better or worse with these additional modelling components. We find that output transformations which tackle heteroscedasticity significantly improve GP modelling capabilities in general (improve marginal log-likelihoods). Similarly, input transformations which tackle non-stationarity significantly improve GP modelling capabilities in general.

	Better	Sig. Better	Worse	Sig. Worse
Heteroscedasticity (Output Transform)	70 (65%)	58 (54%)	38 (35%)	25 (23%)
Non-Stationarity (Input Warping)	106 (98%)	79 (73%)	2 (2%)	0 (0%)

3.1 Answer A.I.: Simple Hyper-parameter Tuning Tasks are Non-stationary

To assess the impact of the extent of non-stationarity on BO performance, we conduct probabilistic regression experiments to gauge the predictive performance of a stationary GP on the hyper-parameter tuning tasks with and without input warping transformations which correct for non-stationarity. We first run a one-sided t-test for each of the 108 tasks where the null hypothesis is that the application of the input warping yields no difference in the log probability metric. In Table 1 significance tests show that in 106/108 tasks, the log probability metric is more favourable when input warping is applied. In 79/108 tasks, the gain is significant at the 95% level of confidence (p-value < 0.025). It is clear that tackling Non-stationarity improves GP fit as shown in Table 1 and improves BO performance, as

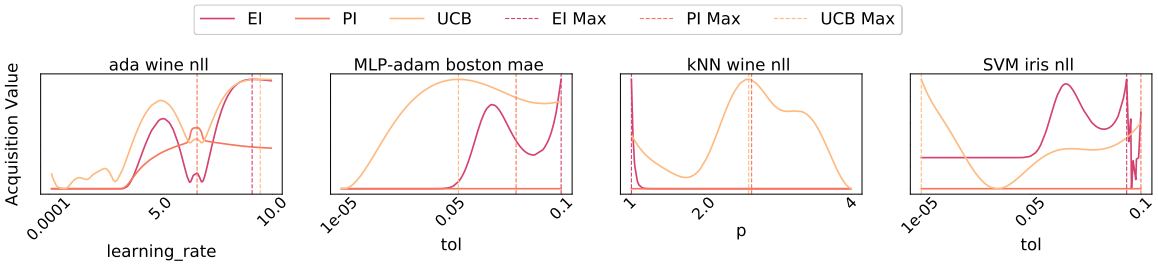


Figure 1: Examples depicting conflicting acquisitions across data sets (Wine, Boston Housing, and Iris) and models (AdaBoost, Multilayer perceptron, K-Nearest neighbours, and support vector machines). The y-axis shows the acquisition value, and x-axis a given configuration of hyperparameters. Clearly, in these examples, not only different acquisitions lead to different optima, but it can be seen that such solutions might conflict (minimum value for one acquisition function is a maximum value for another acquisition function).

shown by the algorithm BO Base w Non-stationarity in Figure 4b. We thus conclude that non-stationarity is an important consideration for BO performance due to the observed effect on the log probability metric.

3.2 Answers A.II.: Simple Hyper-parameter Tuning Tasks are Heteroscedastic

We perform an analogous hypothesis test as in Section 3.1, assessing a vanilla GP's performance with and without output transformations (Box-Cox/ Yeo-Johnson). We run a two-sided paired t-test for each of the 108 tasks where the null hypothesis is that the application of the output transform yields no difference in the log probability metric. In Table 1 significance tests show that in 70/108 tasks, the log probability metric is more favourable when output transformations are applied. In 58/108 tasks, the gain is significant at the 95% level of confidence (p-value < 0.025). It is clear that tackling Heteroscedasticity improves GP fit as shown in Table 1 and improves BO performance, as shown by the algorithm BO Base w Heteroscedasticity in Figure 4b. We thus conclude that heteroscedasticity is an important consideration for BO performance due to it isimpact on the log probability metric.

Furthermore, to gauge the level of heteroscedasticity in the underlying data, we use the Fligner-Killeen (Fligner & Killeen, 1976) and Levene (Levene, 1960) tests. For both tests, the null hypothesis is that the underlying black-box function noise process is homoscedastic. In all 108 tests, we see a p-value significantly lower than 0.05 in 72 tasks using Levene's test, and in 73 tasks using Fligner-Killeen. Such results (shown in detail in Appendix C) imply that at least 66% of the experimental tasks exhibit heteroscedastic behaviour.

3.3 Answer A.III.: No Clear Winner

It has previously been observed that acquisition functions can conflict in their optima (Shahriari et al., 2014). To provide further support for the answer to **Q.III.**, we collect 128 samples from each task by evaluating various hyper-parameter configurations across metrics. We then assemble a data set $\mathcal{D} = \{\textbf{hyper-param}_i, y_i\}_{i=1}^{32}$, where $\textbf{hyper-param}_i$ is a vector with dimensionality dependent on the number of hyper-parameters in a given model, and y_i is an evaluation metric, (e.g., mean squared error) We subsequently fit a GP surrogate model and

consider each of the three acquisition functions from Section 2.2. Given the difficulty involved in the graphical depiction of an acquisition function conflict in more than two dimensions, we examine a simple, two-dimensional illustrative example. From Figure 1, it is apparent that even in the simplest 2D case, many examples of conflicting acquisitions exist. Thus, in higher dimensions this behaviour will also occur.

4. Optimising Bayesian Optimisation

In this section we describe the component design choices that may mitigate for heteroscedastic and non-stationary aspects of commonly-encountered BO problems. Input and output transformations as well as multi-objective acquisition functions have been introduced in isolation previously, whilst acquisition function robustness is unique to this work. The overall design choices produce the method which we name Heteroscedastic and Evolutionary Bayesian Optimisation (HEBO).

4.1 Tackling Heteroscedasticity and Non-stationarity

To parsimoniously handle heteroscedasticity and non-stationarity, we leverage ideas from the warped GP literature (Snelson et al., 2004) where output transformations facilitate the modelling of complex noise processes. We observe that the well-known `Box-Cox` (Box & Cox, 1964) and `Yeo-Jonhson` (Yeo & Johnson, 2000) output transformations in conjunction with the `Kumaraswamy` (Kumaraswamy, 1980) input transformation, offer a balance between simplicity of implementation and empirical performance. In our ablation study (Section 5), we demonstrate that the addition of these two modelling components alone yields large performance gains. Note, we refit the parameters for the output transformation before we refit the GP after receiving a new samples.

Output Transformation for Heteroscedasticity: We consider the `Box-Cox` transformation most frequently used as a corrective mapping for non-Gaussian data. The transform depends on a tuneable parameter ζ and applies the following map to each of the labels: $\mathrm{T}_\zeta(y_l) = {y_l}^\zeta - 1/\zeta$ for $\zeta \neq 0$ and $\mathrm{T}_\zeta(y_l) = \log y_l$ if $\zeta = 0$, where in our case y_l denotes the validation accuracy of the l^{th} hyper-parameter configuration. ζ must be fit based on the observed data such that the distribution of the transformed labels closely resembles a Gaussian distribution. This is achieved by minimising the negative `Box-Cox` likelihood function:

$$\log \left[\sum_{l=1}^{n} \frac{(\mathrm{T}_\zeta(y_l) - \overline{\mathrm{T}}_\zeta(\boldsymbol{y}))^2}{n} \right]^{\frac{n}{2}} + \sum_{l=1}^{n} \log \left[\mathrm{T}_\zeta(y_l) \right]^{(1-\zeta)},$$

where n is the number of datapoints and $\overline{\mathrm{T}_\zeta(\boldsymbol{y})}$ is the sample mean of the transformed labels. `Box-Cox` transforms only consider strictly positive (or strictly negative) labels y_l.

When labels take on arbitrary values, we use the `Yeo-Johnson` transform in place of the `Box-Cox` transform. The `Yeo-Johnson` transform is defined as follows:

$$\texttt{Y.J.}_{\zeta}(y_l) = \begin{cases} \frac{(y_l+1)^{\zeta}-1}{\zeta}, & \text{if } \zeta \neq 0,\ y_l \geq 0 \\ \log(y_l+1), & \text{if } \zeta = 0,\ y_l \geq 0 \\ \frac{(1-y_l)^{2-\zeta}-1}{\zeta-2} & \text{if } \zeta \neq 2,\ y_l < 0 \\ -\log(1-y_l) & \text{if } \zeta = 2,\ y_l < 0. \end{cases}$$

In an analogous fashion to the `Box-Cox` transform, the `Yeo-Johnson`'s parameter is fit based on the observed data through solving the following 1-dimensional optimisation problem:

$$\max_{\zeta} -\frac{n}{2}\log\left[\frac{\sum_{j=1}^{n}(\texttt{Y.J.}_{\zeta}(y_l) - \overline{\texttt{Y.J.}_{\zeta}(\boldsymbol{y})})^2}{n-1}\right] + (\zeta-1)\sum_{i=1}^{n}\left[\text{sign}(y_l)\log(|y_l|+1)\right],$$

with $\overline{\texttt{Y.J.}_{\zeta}(\boldsymbol{y})}$ the sample average computed after applying the `Yeo-Johnson` transformation.

Input Transformations for Non-Stationarity: As a general solution concept for correcting for non-stationarity, we consider input warping see (Snoek et al., 2012). Input warping performs a (usually non-linear and learnable) transformation to the input variables ($\boldsymbol{x}_l$). It was proven in (Snoek et al., 2012) that Input warping also helps tackle non-stationary functions. We rely on the `Kumaraswamy` input warping transform as used in (Snoek et al., 2012), which operates as follows for each input dimension:

$$[\texttt{Kumaraswamy}_{\boldsymbol{\gamma}}(\boldsymbol{x}_l)]_k = 1 - \left(1 - [\boldsymbol{x}_l]_k^{a_k}\right)^{b_k} \ \ \forall k \in [1:d],$$

where d is the dimensionality of the decision variable (i.e. the number of free hyper-parameters), a_k and b_k are tuneable warping parameters for each of the dimensions, and $\boldsymbol{\gamma}$ is a vector concatenating all free parameters, i.e., $\boldsymbol{\gamma} = [a_{1:d}, b_{1:d}]^{\mathsf{T}}$. $\boldsymbol{\gamma}$ is fit based on the observed data. Similar to (Balandat et al., 2020), we optimise $\boldsymbol{\gamma}$ under the marginal likelihood objective used to fit the GP surrogate.

All Modelling Improvements Together: Combining the above corrective measures for heteroscedasticity and non-stationarity leads us to an improved GP surrogate with more flexible modelling capabilities. The implementation of such a model is relatively simple and involves maximising a new marginal likelihood which may be written as:

$$\max_{\boldsymbol{\theta},\boldsymbol{\gamma}} -\frac{1}{2}\text{T}_{\zeta^\star}(\boldsymbol{y})^{\mathsf{T}}(\boldsymbol{K}_{\boldsymbol{\theta}}^{\boldsymbol{\gamma}} + \sigma_{\text{noise}}\boldsymbol{I})^{-1}\text{T}_{\zeta^\star}(\boldsymbol{y}) - \frac{1}{2}|\boldsymbol{K}_{\boldsymbol{\theta}}^{\boldsymbol{\gamma}} + \sigma_{\text{noise}}^2\boldsymbol{I}| - \text{const},$$

where $\boldsymbol{\theta}$ are GP hyper-parameters, $\boldsymbol{\gamma}$ indicates the use of non-stationary transformations, and $\zeta^\star$ denotes the solution to a `Box-Cox` likelihood objective. It is worth noting that we use `Box-Cox` as a running example but as mentioned previously we interchange `Box-Cox` with `Yeo-Johnson` transforms based on the properties of the label y_l. We use $\boldsymbol{K}_{\boldsymbol{\theta}}^{\boldsymbol{\gamma}} \in \mathbb{R}^{n\times n}$ to represent a matrix such that each entry depends on both $\boldsymbol{\theta}$ and $\boldsymbol{\gamma}$, where $k_{\boldsymbol{\theta}}^{\boldsymbol{\gamma}}(\boldsymbol{x},\boldsymbol{x}') = k_{\boldsymbol{\theta}}(\texttt{Kumaraswamy}_{\boldsymbol{\gamma}}(\boldsymbol{x}), \texttt{Kumaraswamy}_{\boldsymbol{\gamma}}(\boldsymbol{x}'))$.

4.2 Tackling Acquisition Conflict & Robustness

Having proposed modifications to the surrogate model component of the Bayesian optimisation scheme, we now turn our attention to the acquisition maximisation step. In particular, we focus on two considerations, the first related to the assumption of a perfect GP surrogate, and the second centred on conflicting acquisitions.

4.2.1 A Robust Acquisition Objective

As mentioned in Section 2.2, the acquisition maximisation step assumes that an adequate surrogate model is readily available. During early rounds of training especially, where data is scarce, such a property is often violated, leading to potentially severe model misspecification. One way to tackle such model misspecification is to adopt a robust formulation (Kirschner et al., 2020; Klein et al., 2017) which attempts to identify the best-performing query location under the worst-case GP model, i.e., solving $\max_{\boldsymbol{x}} \min_{\boldsymbol{\theta}} \alpha^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D})$. Though such a formulation admits a solution $\boldsymbol{x}^\star$ that is robust to worst-case misspecification in $\boldsymbol{\theta}$, having a max min acquisition is problematic for several reasons. From a conceptual perspective max min formulations are known to lead to very conservative solutions if not correctly constrained or regularised since the optimiser possesses the power to impair the GP fit while updating $\boldsymbol{\theta}$[4]. From the perspective of implementation, one encounters two further issues. First, no global convergence guarantees are known for the non-convex, non-concave case (Lin et al., 2020), and second, ensuring gradients can propagate through the computation graph restricts surrogates and acquisition functions to be within the same programming framework.

To avoid worst-case solutions and engender independence between acquisition functions and surrogate models, given a set of parameters from a trained GP $\boldsymbol{\theta}$, we leverage ideas from domain randomisation (Tobin et al., 2017) and consider an expected formulation instead over these parameters: $\max_{\boldsymbol{x}} \alpha^{\boldsymbol{\theta}}_{\text{rob.}}(\boldsymbol{x}|\mathcal{D}) \equiv \max_{\boldsymbol{x}} \mathbb{E}_{\epsilon\sim\mathcal{N}(\mathbf{0},\sigma_\epsilon^2\boldsymbol{I})}\left[\alpha^{\boldsymbol{\theta}+\epsilon}(\boldsymbol{x}|\mathcal{D})\right]$. Importantly, this problem seeks to find new query locations that perform well on average over a distribution of surrogate models in favour of assuming a perfect surrogate. Despite on an intractable nature of $\alpha^{\boldsymbol{\theta}}_{\text{rob.}}(\cdot|\mathcal{D})$, in HEBO we show (the rigorous representation of this result is presented in Appendix B) that it can be approximated with any arbitrary precision and high confidence with $\overline{\alpha}^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) = \alpha^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) + \mathcal{N}(0,\sigma_n^2)$ by properly choosing parameters σ_ϵ and σ_n:

Theorem: (Informal) Let us consider the stochastic version of the acquisition function utilised in HEBO and given by $\overline{\alpha}^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) = \alpha^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) + \mathcal{N}(0,\sigma_n^2)$ and Let $\alpha^{\boldsymbol{\theta}}_{\text{rob.}}(\boldsymbol{x}|\mathcal{D}) \equiv \mathbb{E}_{\epsilon\sim\mathcal{N}(\mathbf{0},\sigma_\epsilon^2\boldsymbol{I})}\left[\alpha^{\boldsymbol{\theta}+\epsilon}(\boldsymbol{x}|\mathcal{D})\right]$ be the robust form of the standard acquisition function given as expectation over random perturbation of parameter $\boldsymbol{\theta}$. Then, with proper choice of parameters σ_n and σ_ϵ, HEBO acquisition function $\overline{\alpha}^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D})$ accurately approximates the robust acquisition function $\alpha^{\boldsymbol{\theta}}_{\text{rob.}}(\boldsymbol{x}|\mathcal{D})$ with high probability [5] and for any $\boldsymbol{\theta}, \boldsymbol{x}$.

[4] One can make a case for augmenting the objective with a constraint such that updates for $\boldsymbol{\theta}$ remain close to $\boldsymbol{\theta}^\star$ of the marginal likelihood. The ideal enforced proximity value however remains unclear in the robust acquisition literature to date (Abdullah et al., 2019; Kirschner et al., 2020).

[5] Here we use the common approach for proving stochastic expressions with high probability (see (Tripuraneni et al., 2017), (Zhu & Li, 2016)). Specifically, we show that for any confidence parameter $\delta \in (0,1)$ the stochastic expression under consideration is valid with probability at least $1-\delta$.

4.2.2 Multi-objective Acquisition functions

As a final component of our general framework, we propose the use of multi-objective acquisitions seeking a Pareto-front solution. This formulation facilitates the process of "hedging" between different acquisitions such that no single acquisition dominates the solution (Lyu et al., 2018). Formally, we solve

$$\max_{\boldsymbol{x}} \left(\overline{\alpha}^{\boldsymbol{\theta}}_{\mathrm{EI}}(\boldsymbol{x}|\mathcal{D}), \overline{\alpha}^{\boldsymbol{\theta}}_{\mathrm{PI}}(\boldsymbol{x}|\mathcal{D}), \overline{\alpha}^{\boldsymbol{\theta}}_{\mathrm{UCB}}(\boldsymbol{x}|\mathcal{D}) \right), \tag{2}$$

where $\overline{\alpha}^{\boldsymbol{\theta}}_{\mathrm{type}}(\boldsymbol{x}|\mathcal{D})$ is a robust acquisition of type $\in \{\mathrm{EI}, \mathrm{PI}, \mathrm{UCB}\}$ as introduced in the previous section. We also note that our formulation is designed to admit the use of a robust objective value of $\overline{\alpha}^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) = \alpha^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) + \eta_n$ with η_n being a sample from $\mathcal{N}(0, \sigma_n)$ at each iteration of the evolutionary solver.

Although solving the problem in Equation 2 is a formidable challenge, we note the existence of many mature multi-objective optimisation algorithms. These range from first-order (Kingma & Welling, 2014) to zero-order (Loshchilov & Hutter, 2016; Gabillon et al., 2020) and evolutionary methods (Hansen, 2016; Deb et al., 2002). Due to the discrete nature of hyper-parameters in machine learning tasks, we advocate the use of evolutionary solvers that naturally handle categorical and integer-valued variables. In our experiments, we employ the non-dominated sorting genetic algorithm II (`NSGA-II`) which allows for mixed variable crossover and mutation to optimise real-valued and integer-valued inputs (Deb et al., 2002). We use the implementation of `NSGA-II` found in the `Pymoo` (Blank & Deb, 2020) library. Alternatively, one may use the GP Hedge acquisition as used in `Dragonfly` (Kandasamy et al., 2020) in (Hoffman et al., 2011b) or in `SkOpt` to select between acquisitions. We however, observed this formulation to perform poorly when compared against individual acquisitions.

5. Experiments and Results

In this section, we continue our empirical evaluation and validate gains (if any) that arise from the improvements proposed in Section 4. The experimental setup remains as described in Section 3. To assess performance, we use the normalised task score[6]. We run experiments on either 16 iterations with a batch of 8 query points per iteration or 100 iterations with 1 query point. Each experiment is repeated for 20 random seeds. We baseline against a wide range of solvers that either rely on BO-strategies or follow zero-order techniques such as differential evolution or particle swarms. These include `SkOpt` (Pedregosa et al., 2011) [7] `pySOT` [8] a parallel global optimisation package (Eriksson et al., 2019a), `HyperOpt` (Bergstra et al., 2015) [9], `OpenTuner` [10] a package for ensembling methods (Ansel et al., 2014), `NeverGrad` (Rapin & Teytaud, 2018) [11] a gradient-free optimisation toolbox (where we use the One Plus One optimiser with the associated label `NeverGrad (1+1)`), `BOHB` (Falkner

[6] Note, we don't report the time to compute query points per algorithm as this was under 20 seconds per query point batch.

[7] https://github.com/scikit-optimize/scikit-optimize

[8] https://github.com/dme65/pySOT

[9] https://github.com/hyperopt/hyperopt

[10] https://github.com/jansel/opentuner

[11] https://github.com/facebookresearch/nevergrad

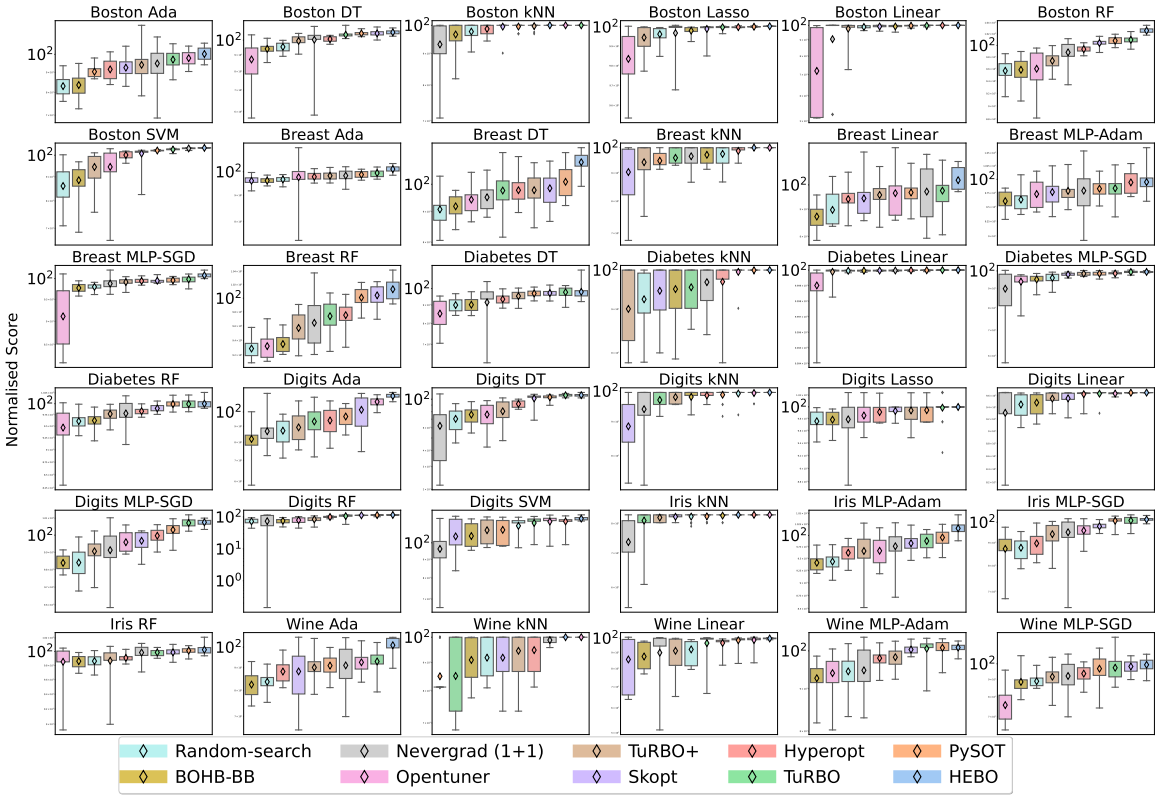


Figure 2: HEBO compared against all baselines for 16 iterations and a batch size of 8 query points per iteration. Each experiment is repeated with 20 random seeds. We average each seed over both metrics in all tasks and display a subset of 36 summary plots for the 108 black-box functions. HEBO achieves the highest normalised mean score in 68.5% of the 108 black-box functions. Full results for the 108 tasks are presented in Appendix D in tabular format.

et al., 2018) [12] and Dragonfly (Kandasamy et al., 2020) [13]. Additionally, we carried our modelling improvements to TuRBO [14] (Eriksson et al., 2019b), augmenting the standard GP with mitigation strategies from Section 4 producing a new baseline that we entitle TuRBO+. Finally, we introduce Heteroscedastic Evolutionary Bayesian Optimisation (HEBO), in which we construct an optimiser with the improvements introduced in Section 4.

Implementation Details for BOHB: BOHB is a scalable hyper-parameter tuning algorithm introduced in (Falkner et al., 2018) mixing bandits and BO approaches to achieve both competitive anytime and final performance. Contrary to the other solvers considered in this paper, BOHB is specifically designed to tackle multi-fidelity optimisation and uses the Hyperband (Li et al., 2017) routine to define the fidelity levels under which points are asynchronously evaluated. The selection of points follows a BO strategy based on the Tree

[12] https://github.com/automl/HpBandSter
[13] https://github.com/dragonfly/dragonfly
[14] https://github.com/rdturnermtl/bbo_challenge_starter_kit/

Parzen Estimator (TPE) method. Given a data set $\mathcal{D}$ of observed data points and a threshold $\alpha \in \mathbb{R}$, the TPE models $p(\boldsymbol{x}|y)$, using kernel density estimates of

$$\ell(\boldsymbol{x}) = p(y < \alpha|\boldsymbol{x}, \mathcal{D})$$
$$g(\boldsymbol{x}) = p(y \geq \alpha|\boldsymbol{x}, \mathcal{D}).$$

In the TPE algorithm, maximising the expected improvement criterion

$$\alpha_{\text{EI}}(\boldsymbol{x}) = \int \max(0, \alpha - p(y|\boldsymbol{x}))p(y|\boldsymbol{x})dy$$

is equivalent to maximising the ratio $r(x) = \frac{\ell(\boldsymbol{x})}{g(\boldsymbol{x})}$ which is carried out to select a single new candidate point at a time.

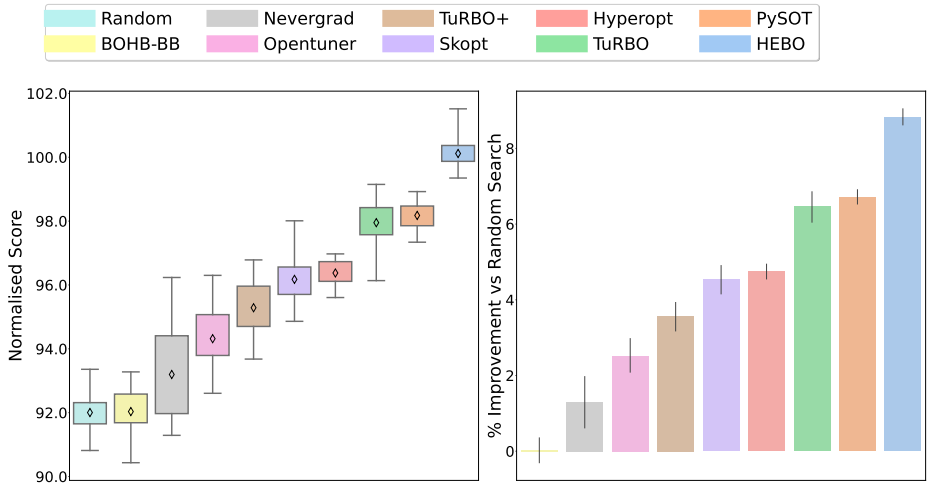
In the absence of a multi-fidelity setup in our experiments, we run a modified version of the `BOHB` algorithm implemented in the `HpBandSter` package. We leave the TPE method for modelling unchanged but ignore the fidelity level assignment from Hyperband. Moreover, as our experimental setup involves batch acquisitions, we tested two alternatives to the standard BOHB acquisition procedure to support synchronous suggestion of multiple points. In the first approach, we run q independent maximisation processes of $r(\boldsymbol{x})$ from random starting points and recover a single candidate from each process to form the q-batch suggestion. In the second approach, we obtain one point as a result of a single maximisation of $r(\boldsymbol{x})$ and we sample $q-1$ random points to complete the q-batch suggestion. As the latter method yields better overall performance, the results reported under the `BOHB-BB` label are obtained using the second approach.

5.1 Black-box Functions

As discussed in Section 3, we evaluate black-box optimisation solvers on a large set of tasks from the `Bayesmark` package. Each task involves optimising the hyper-parameters of a machine learning algorithm to minimise the cross validation loss incurred when this model is applied in a regression (reg) or a classification (clf) setting for a given data set. Thus, a task is characterised by a model, a data set and a loss function (metric) quantifying the quality of the regression or classification performance. In total, 108 distinct tasks can be defined from the valid combinations of the nine models specified in Table 2, the following six real-world UCI datasets (Dua & Graff, 2017), Boston (reg), Breast Cancer (clf), Diabetes (reg), Digits (clf), Iris (clf) and Wine (clf); the following two regression metrics, negative mean-squared error (MSE), negative mean absolute error (MAE), and two classification metrics, negative log-likelihood (NLL) and negative accuracy (ACC). The results reported in Figures 3 and 4 have been obtained by applying each black-box optimisation method using 16 iterations of 8-batch acquisition steps on all of the 108 tasks. In order to provide a reliable evaluation of the different solvers, we repeated each run with 20 random seeds and considered the normalised score given by:

$$\textbf{Normalised Score} = 100 \times \frac{\mathcal{L} - \mathcal{L}^*}{\mathcal{L}^{\text{rand}} - \mathcal{L}^*} \tag{3}$$

where $\mathcal{L}$ is the best-achieved cross validation loss at the end of the 16 acquisition steps, $\mathcal{L}^*$ is the estimated optimal loss for the task and $\mathcal{L}^{\text{rand}}$ is the mean loss (across multiple runs)



(a) Empirical Performance Gain

Figure 3: Analysis of the results on 108 tuning tasks. (Left) Normalised score comparison demonstrating that HEBO (i.e., BO with improvements from Section 4) outperforms competitor algorithms. We observe a 5% relative improvement to SOTA optimisers such as TuRBO. (Right) HEBO yields an 8% improvement compared to random search.

obtained using random search with the same number of acquisition steps. The normalisation procedure permits aggregation of the scores across tasks despite the different cross-validation loss functions used.

5.2 Black-box Optimisation Input Variables

We provide in Table 2 and Table 3 the list of the hyper-parameters controlling the behaviour of each model along with their optimisation domains, which can differ depending on whether the model is used for a classification or a regression task. The search domain may include a mix of continuous and integer-valued variables (e.g. the MLP-SGD hyper-parameter set includes an integer-valued hidden layer size, and a continuous-valued initial learning rate that can take on values between 10^{-5} and 10^{-1}). The dimensionality of the input space, i.e. the number of hyper-parameters to tune, ranges from 2 to 9. We specify in the final column of the tables whether the search domain is modified through a standard transformation (log or logit) in order to facilitate optimisation.

Table 4 synthesises the performance achieved on the 108 tasks by the black-box optimisation solvers considered in our experiments. We note that the distribution of the scores attained by HEBO has the largest mean and the smallest standard deviation, indicating that HEBO significantly outperforms competitor algorithms.

Figure 2 and Figure 3 demonstrates gains from adopting the general HEBO framework. We note that due to optimising over numerous regression and classification metrics, we show that irrelevant of the validation score HEBO performs better than other optimisers. In Figure 4, we compare HEBO against baselines and report up to an 8% performance gain relative to

Table 2: Search spaces for hyper-parameter tuning on classification tasks. We specify the variable type of each hyper-parameter (with $\mathbb{R}$ for real-valued and $\mathbb{Z}$ for integer- valued) as well as the search domain. We specify $\log -\mathcal{U}$ (resp. $\text{logit} - \mathcal{U}$) to indicate that a log (resp. logit) transformation is applied to the optimisation domain.

Model	Parameter	Type	Domain
kNN	n_neighbors	$\mathbb{Z}$	$\mathcal{U}(1, 25)$
	p	$\mathbb{Z}$	$\mathcal{U}(1, 4)$
Support Vector Machine	C	$\mathbb{R}$	$\log -\mathcal{U}(1, 10^3)$
	gamma	$\mathbb{R}$	$\log -\mathcal{U}(10^{-4}, 10^{-3})$
	tol	$\mathbb{R}$	$\log -\mathcal{U}(10^{-5}, 10^{-1})$
Decision Tree	max_depth	$\mathbb{Z}$	$\mathcal{U}(1, 15)$
	min_samples_split	$\mathbb{R}$	$\text{logit} - \mathcal{U}(0.01, 0.99)$
	min_samples_leaf	$\mathbb{R}$	$\text{logit} - \mathcal{U}(0.01, 0.49)$
	min_weight_fraction_leaf	$\mathbb{R}$	$\text{logit} - \mathcal{U}(0.01, 0.49)$
	max_features	$\mathbb{R}$	$\text{logit} - \mathcal{U}(0.01, 0.99)$
	min_impurity_decrease	$\mathbb{R}$	$\mathcal{U}(0, 0.5)$
Random Forest	max_depth	$\mathbb{Z}$	$\mathcal{U}(1, 15)$
	max_features	$\mathbb{R}$	$\text{logit} - \mathcal{U}(0.01, 0.99)$
	min_samples_split	$\mathbb{R}$	$\text{logit} - \mathcal{U}(0.01, 0.99)$
	min_samples_leaf	$\mathbb{R}$	$\text{logit} - \mathcal{U}(0.01, 0.49)$
	min_weight_fraction_leaf	$\mathbb{R}$	$\text{logit} - \mathcal{U}(0.01, 0.49)$
	min_impurity_decrease	$\mathbb{R}$	$\mathcal{U}(0, 0.5)$
MLP-Adam	hidden_layer_sizes	$\mathbb{Z}$	$\mathcal{U}(50, 200)$
	alpha	$\mathbb{R}$	$\log -\mathcal{U}(10^{-5}, 10^{1})$
	batch_size	$\mathbb{Z}$	$\mathcal{U}(10, 250)$
	learning_rate_init	$\mathbb{R}$	$\log -\mathcal{U}(10^{-5}, 10^{-1})$
	tol	$\mathbb{R}$	$\log -\mathcal{U}(10^{-5}, 10^{-1})$
	validation_fraction	$\mathbb{R}$	$\text{logit} - \mathcal{U}(0.1, 0.9)$
	beta_1	$\mathbb{R}$	$\text{logit} - \mathcal{U}(0.5, 0.99)$
	beta_2	$\mathbb{R}$	$\text{logit} - \mathcal{U}(0.9, 1 - 10^{-6})$
	epsilon	$\mathbb{R}$	$\log -\mathcal{U}(10^{-9}, 10^{-6})$
MLP-SGD	hidden_layer_sizes	$\mathbb{Z}$	$\mathcal{U}(50, 200)$
	alpha	$\mathbb{R}$	$\log -\mathcal{U}(10^{-5}, 10^{1})$
	batch_size	$\mathbb{Z}$	$\mathcal{U}(10, 250)$
	learning_rate_init	$\mathbb{R}$	$\log -\mathcal{U}(10^{-5}, 10^{-1})$
	power_t	$\mathbb{R}$	$\text{logit} - \mathcal{U}(0.1, 0.9)$
	tol	$\mathbb{R}$	$\log -\mathcal{U}(10^{-5}, 10^{-1})$
	momentum	$\mathbb{R}$	$\text{logit} - \mathcal{U}(0.001, 0.999)$
	validation_fraction	$\mathbb{R}$	$\text{logit} - \mathcal{U}(0.1, 0.9)$
AdaBoost	n_estimators	$\mathbb{Z}$	$\mathcal{U}(10, 100)$
	learning_rate	$\mathbb{R}$	$\log -\mathcal{U}(10^{-4}, 10^{1})$
Lasso	C	$\mathbb{R}$	$\log -\mathcal{U}(10^{-2}, 10^{2})$
	intercept_scaling	$\mathbb{R}$	$\log -\mathcal{U}(10^{-2}, 10^{2})$
Linear	C	$\mathbb{R}$	$\log -\mathcal{U}(10^{-2}, 10^{2})$
	intercept_scaling	$\mathbb{R}$	$\log -\mathcal{U}(10^{-2}, 10^{2})$

Table 3: Models and search spaces for hyper-parameter tuning on regression tasks. Models having the same search spaces for classification and regression tasks are omitted (cf. Table 2).

Model	Parameter	Type	Domain
AdaBoost	n_estimators	$\mathbb{Z}$	$\mathcal{U}(10, 100)$
	learning_rate	$\mathbb{R}$	$\log -\mathcal{U}(10^{-4}, 10^{1})$
Lasso	alpha	$\mathbb{R}$	$\log -\mathcal{U}(10^{-2}, 10^{2})$
	fit_intercept	$\mathbb{Z}$	$\mathcal{U}(0, 1)$
	normalize	$\mathbb{Z}$	$\mathcal{U}(0, 1)$
	max_iter	$\mathbb{Z}$	$\log -\mathcal{U}(10, 5000)$
	tol	$\mathbb{R}$	$\log -\mathcal{U}(10^{-5}, 10^{-1})$
	positive	$\mathbb{Z}$	$\mathcal{U}(0, 1)$
Linear	alpha	$\mathbb{R}$	$\log -\mathcal{U}(10^{-2}, 10^{2})$
	fit_intercept	$\mathbb{Z}$	$\mathcal{U}(0, 1)$
	normalize	$\mathbb{Z}$	$\mathcal{U}(0, 1)$
	max_iter	$\mathbb{Z}$	$\log -\mathcal{U}(10, 5000)$
	tol	$\mathbb{R}$	$\log -\mathcal{U}(10^{-4}, 10^{-1})$

Algorithm	Mean	Std	Median	40^{th} Centile	30^{th} Centile	20^{th} Centile	5^{th} Centile
HEBO	**100.12**	**8.70**	**100.01**	**100.00**	**99.88**	**98.64**	**85.71**
PySOt	98.18	9.03	100.00	99.81	98.60	95.36	80.00
TuRBO	97.95	10.80	100.00	99.88	98.75	95.26	78.63
HyperOpt	96.37	8.79	99.31	98.16	95.94	92.38	78.52
SkOpt	96.18	11.51	99.78	98.66	96.73	91.62	74.77
TuRBO+	95.29	10.93	98.97	97.60	95.27	90.92	74.77
OpenTuner	94.32	14.18	98.44	96.93	93.84	89.97	68.96
Nevergrad (1+1)	93.20	17.52	99.65	97.84	94.57	88.28	55.34
BOHB	92.03	11.16	96.02	93.55	90.14	85.71	67.82
Random-Search	92.00	11.71	96.18	93.55	90.05	85.16	69.55

Table 4: Mean and n-th percentile normalised scores over 108 black-box functions, each repeated with 20 random seeds. We observe significant mean improvements from HEBO compared to all competitor algorithms.

a random search strategy. It is also worth noting that TuRBO+ tends to underperform [15], achieving ca. 4% improvement relative to random search. We believe such a result is related to the interplay between our approach's capabilities to address heteroscedasticity and non-stationarity as well as the size of the trust regions; an interesting avenue that we plan to explore in future work, as well as experimenting with deeper neural networks as well as other architectures such as convolutional/ recurrent neural networks. Overall, HEBO achieves the highest normalised mean scores on 74 of the 108 datasets. Complete results on all tasks may be found in Appendix D.

Comparison to Asynchronous BO Algorithms: We perform a comparison to black-box optimisers, such as Dragonfly and BOHB, which operate in the asynchronous setting. We run each method for 100 iterations of data collection with a single query location per iteration. We label the asynchronous algorithms without their multi-fidelity components with an addition BB for black-box optimiser (Dragonfly-BB and BOHB-BB) to assess black-box optimisation performance only. The results of Figure 4a show that in the asynchronous setting, both Dragonfly-BB and BOHB-BB under-perform relative to other black-box optimisers, with HEBO performing best. However, this result is not surprising as asynchronous methods trade off sample efficiency with speed. Nevertheless, this experiment reveals a large gap in suggestion power between SOTA asynchronous and synchronous methods.

5.3 Ablation Results

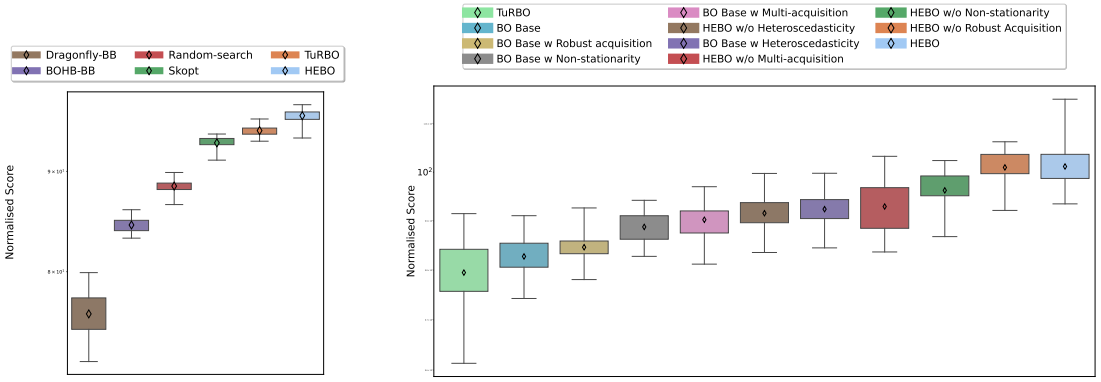
To better understand the relative importance of each component of the HEBO algorithm, we conduct an ablation study by first removing each component of HEBO and testing the remaining components and second, by starting with basic BO and sequentially adding and testing each component of HEBO. The components comprise the consideration of heteroscedasticity, non-stationarity and robustness, as well as the use of a multiobjective acquisition function. We report average normalised scores in Figure 4b. The precedence order observed is: heteroscedasticity, multi-objective acquisition functions, non-stationarity and robustness.

6. Related Work

We introduce work on the following topics relating to modelling, acquisition and optimisers in Bayesian optimisation:

Heteroscedasticity with output transforms: Among various approaches to handling heteroscedasticity (Kersting et al., 2007; Lázaro-Gredilla & Titsias, 2011; Kuindersma et al., 2013; Calandra, 2017; Griffiths et al., 2021a), transforming the output variables is a straightforward option giving rise to warped Gaussian processes (Snelson et al., 2004). More recently, output transformations have been extended to compositions of elementary functions (Rios & Tobar, 2019) and normalising flows (Rezende & Mohamed, 2015; Maronas et al., 2020). Output transformations have not featured prominently in the Bayesian optimisation literature, perhaps due to the commonly-held opinion that warped GPs require more data relative to standard GPs in order to function as effective surrogates (Nguyen & Osborne,

[15] We believe this due to the trust region not being modelled correctly with input warping.



(a) Competitor Comparison. (b) Ablation Study

Figure 4: (a) We compare `HEBO` against several popular hyper-parameter tuning approaches including `BOHB-BB` and `Dragonfly-BB`, running all methods for 100 iterations with a batch size of 1 (i.e. one set of hyper-parameters queried per iteration). `BOHB-BB` and `Dragonfly-BB` feature asynchronous queries, suggesting a batch of one set of hyper-parameters at each iteration. We remove the multi-fidelity components from `BOHB` and `Dragonfly` to assess Black-Box optimisation alone, hence the additional BB appended to their label. (b) Ablation study where X denotes a general component of `HEBO`. `HEBO` w/o X takes one component X out at a time and BO Base w X adds one component X in at a time. We show `TuRBO` as a baseline and refer to `HEBO` with all significant components removed as `BO Base`. The ablation demonstrates that the corrections for each misspecified modelling assumption yield a tangible gain in empirical performance.

2020). Rather than introduce additional hyper-parameters to the GP, we enable efficient output warping through methods that only require pre-training. Recent work (Eriksson & Poloczek, 2021) has also investigated Gaussian copula transforms which may prove to be particularly effective in situations where there are outliers.

Non-stationarity with input warping: Many surrogate models with input warping exist for optimising non-stationary black-box objectives (Snoek et al., 2014; Calandra et al., 2016; Oh et al., 2018) and have enjoyed particular success in hyper-parameter tuning where the natural scale of parameters is often logarithmic. Traditionally, a Beta cumulative distribution function is used. In this paper, we adopt the `Kumaraswamy` warping which is another instance of the generalised Beta class of distributions which we have observed to achieve superior performance (Snoek et al., 2014) [16]; confirming results reported in (Balandat et al., 2020).

Multi-objective acquisition ensembles: Multi-objective acquisition ensembles were first proposed in (Lyu et al., 2018) and are closely related to portfolios of acquisition functions (Hoffman et al., 2011a; Shahriari et al., 2014; Balandat et al., 2020). In this form, the optimisation problem involves at least two conflicting and expensive black-box objectives

[16] For clarity we note that the input warping function used in (Snoek et al., 2014) is the same one used in this work.

and as such, solutions are located along the Pareto-efficient frontier. The multi-objective acquisition ensemble employs these ideas to find a Pareto-efficient solution amongst multiple acquisition functions. Although we utilised the multi-objective acquisition ensemble, we note that our framework is solver agnostic in so far as any multi-objective optimiser (Abdolshah et al., 2019) may be applied.

Robustness of Acquisitions: Methods achieving robustness with respect to either surrogates (Park et al., 2020) or the optimisation process (Bogunovic et al., 2018; Bertsimas et al., 2010) have been previously proposed. Most relevant to our setting, is the approach of (Bogunovic et al., 2018) that introduces robustness to BO by solving a max min objective to determine optimal input perturbations. Their method, however, relies on gradient ascent-descent-type algorithms that require real-valued variables and are not guaranteed to converge in the general non-convex, non-concave setting (Lin et al., 2020). On the other hand, our solution possesses two advantages: 1) simplicity of implementation as we merely require random perturbations of acquisition functions to guarantee robustness, and 2) support for mixed variable solutions through the use of evolutionary solvers.

7. Conclusion & Future Work

In this paper, we presented an in-depth empirical study of Bayesian optimisation for hyper-parameter tuning tasks. We demonstrated that even the simplest among machine learning problems can exhibit heteroscedasticity and non-stationarity. We also reflected on the affects of misspecified models and conflicting acquisition functions. We augmented BO algorithms with various enhancements and revealed that with a revised set of assumptions BO can in fact act as a competitive baseline in hyper-parameter tuning. We highlight the large discrepancy between suggestion power of synchronous and asynchronous methods. We hope for future work to focus on integrating the best of asynchronous and synchronous methods for optimal performance. We hope this paper's findings can guide the community when employing black-box and Bayesian optimisation in practice.

8. Limitations and Broader Impact Statement

Whilst we optimise models for downstream tasks, we do not consider controlling/ preventing the biases learnt from the machine learning models. We urge practitioners to always perform in-depth analysis of the features used for machine learning models and, particularly when models being optimised input sensitive information, attend to sensitive information accordingly.

Acknowlegments

We would like to ackowledge that the Alexander I. Cowen-Rivers, Wenlong Lyu and Rasul Tutunov are joint first authors. We would also like to acknowledge that Haitham Bou-Ammar holds an Honorary position at University College London.

Appendix A. Addition Detail Of Hypothesis Tests

Levene's Test Levene's test statistic is defined as

$$W = \frac{N-k}{k-1} \cdot \frac{\sum_{i=1}^{k} n(\bar{Z}_{i\cdot} - \bar{Z}_{\cdot\cdot})^2}{\sum_{i=1}^{k}\sum_{j=1}^{n}(Z_{ij} - \bar{Z}_{i\cdot})^2},$$

where $N = k \times n$, $Z_{ij} = |Y_{ij} - \frac{1}{n}\sum_{j=1}^{n} Y_{ij}|$, $\bar{Z}_{i\cdot} = \frac{1}{n}\sum_{j=1}^{n} Z_{ij}$ and $\bar{Z}_{\cdot\cdot} = \frac{1}{k}\sum_{i=1}^{k} \bar{Z}_{i\cdot}$, for all $i = 1, \ldots, k$, $j = 1, \ldots, n$. Levene's test rejects the null hypothesis of homoscedasticity H_0 if

$$W > F_{\alpha,k-1,N-k},$$

where $F_{\alpha,k-1,N-k}$ is the upper critical value at a significance level α of the F distribution with $k-1$ and $N-k$ degrees of freedom. The Fligner-Killeen test is an alternative to Levene's test that is particularly robust to outliers.

Fligner-Killeen Test: Computation of the Fligner-Killeen test involves ranking all the absolute values $\{|Y_{ij} - \tilde{Y}_i|\}_{1\le i\le k, 1\le j\le n}$, where $\tilde{Y}_i$ is the median of $\{Y_{ij}\}_{1\le j\le n}$. Increasing scores $a_{N,r} = \Phi^{-1}\left(\frac{1+\frac{r}{N+1}}{2}\right)$ are associated with each rank $r = 1, \ldots, N$, where $N = kn$ and $\Phi(\cdot)$ is the cumulative distribution function for a standard normal random variable. We denote the rank score associated with Y_{ij} as r_{ij}. The Fligner-Killeen test statistic is given by

$$\chi_o^2 = \frac{\sum_{i=1}^{k} n\left(\bar{A}_i - \bar{a}\right)^2}{V^2},$$

where $\bar{A}_i = \frac{1}{n}\sum_{j=1}^{n} a_{N,r_ij}$, $\bar{a} = \frac{1}{N}\sum_{r=1}^{N} a_{N,r}$ and $V^2 = \frac{1}{N-1}\sum_{r=1}^{N}(a_{N,r} - \bar{a})^2$. As χ_0 has an asymptotic $\mathcal{X}^2$ distribution with $(k-1)$ degrees of freedom, the test rejects the null hypothesis of homoscedasticity H_0 if

$$\chi_0 > \mathcal{X}^2_{\alpha,k-1}$$

where $\mathcal{X}^2_{\alpha,k-1}$ is the upper critical value at a significance level α of the $\mathcal{X}^2$ distribution with $k-1$ degrees of freedom.

Appendix B. Details of Robust Acquisition

Though appealing, our formulation still assumes access to the GP hyper-parameters which complicates the implementation by restricting models and optimisers to the same programming paradigm. Ideally, we would wish to illicit robustness through only the GP predictive mean and predictive variance. Fortunately, we are able to show that upon a simple acquisition perturbation it becomes possible to approximate $\alpha_{\text{rob}}(\cdot)$ above. As such, we demonstrate that robust acquisition formulations are achievable using only the GP predictive mean and variance.

Theorem: Let us consider the stochastic version of the acquisition function utilised in HEBO and given by $\overline{\alpha}^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) = \alpha^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) + \eta\sigma_n$ with $\eta \sim \mathcal{N}(0,1)$ and standard deviation

parameter $\sigma_n > 0$ [17]. Let $\alpha^{\boldsymbol{\theta}}_{\text{rob.}}(\boldsymbol{x}|\mathcal{D}) \equiv \mathbb{E}_{\epsilon\sim\mathcal{N}(\mathbf{0},\sigma_\epsilon^2\boldsymbol{I})}\left[\alpha^{\boldsymbol{\theta}+\epsilon}(\boldsymbol{x}|\mathcal{D})\right]$ be the robust form of the standard acquisition function given as expectation over random perturbation of parameter $\boldsymbol{\theta}$. Then, by properly choosing parameters σ_n and σ_ϵ with high probability[18], HEBO acquisition function $\overline{\alpha}^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D})$ accurately approximates the robust acquisition function $\alpha^{\boldsymbol{\theta}}_{\text{rob.}}(\boldsymbol{x}|\mathcal{D})$ for any $\boldsymbol{\theta}, \boldsymbol{x}$. More formally, for any $\rho \in (0,1)$ and $\delta \in (0,1)$, there are parameters $\sigma_n = \sigma_n(\rho,\delta)$ and $\sigma_\epsilon = \sigma_\epsilon(\rho,\delta)$ such that:

$$\left|\overline{\alpha}^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) - \alpha^{\boldsymbol{\theta}}_{\text{rob.}}(\boldsymbol{x}|\mathcal{D})\right| \leq \rho, \qquad \forall \boldsymbol{\theta}, \boldsymbol{x}$$

with probability at least $1-\delta$.

B.0.1 Proof of the Robustness Bound

Let $\delta \in (0,1)$ be the desired probability threshold, and $\rho \in (0,1)$ be a desired accuracy parameter. Consider the GP with mean function $m(\boldsymbol{x})$ and covariance function $k_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{x}')$ such that $\forall \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$, $\boldsymbol{\theta} \in \mathbb{R}^p$:

$$\begin{aligned} &|k_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{x})| \geq M_0, \quad |k_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{x}')| \leq M_1, \\ &||\nabla_{\boldsymbol{\theta}} k_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{x}')||_2 \leq M_2, \quad |m(\boldsymbol{x})| \leq M_4. \end{aligned} \tag{4}$$

Moreover, assume that observations $y \in \mathcal{D}$ are bounded, i.e. $|y| \leq C$ and let $\overline{\alpha}^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) = \alpha^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) + \eta\sigma_n$ with η a standard normal random variable. Then, we are going to show that there are values $c_1 = c_1(\rho,\delta)$ and $c_2 = c_2(\rho,\delta)$, such that choosing $\sigma_n \leq c_1$ and $\sigma_\epsilon \leq c_2$:

$$\left|\overline{\alpha}^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) - \mathbb{E}_{\epsilon\sim\mathcal{N}(\mathbf{0},\sigma_\epsilon^2\boldsymbol{I})}\left[\alpha^{\boldsymbol{\theta}+\epsilon}(\boldsymbol{x}|\mathcal{D})\right]\right| \leq \rho.$$

with probability at least $1-\delta$. Note, the robust form of the acquisition function given as $\alpha^{\boldsymbol{\theta}}_{\text{rob.}}(\boldsymbol{x}|\mathcal{D}) \equiv \mathbb{E}_{\epsilon\sim\mathcal{N}(\mathbf{0},\sigma_\epsilon^2\boldsymbol{I})}\left[\alpha^{\boldsymbol{\theta}+\epsilon}(\boldsymbol{x}|\mathcal{D})\right]$ constitutes an intractable integral. Therefore, in order to be optimised during the course of Bayesian optimisation, the intractable integral must be replaced by an accurate approximation. Without loss of generality we choose the UCB acquisition function $\alpha^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) = \alpha^{\boldsymbol{\theta}}_{\text{UCB}}(\boldsymbol{x}|\mathcal{D})$ and to avoid technical complications relating to multivariate calculus we consider a batch size $q = 1$. In this case, the UCB acquisition function can be written as $\alpha^{\boldsymbol{\theta}}_{\text{UCB}}(\boldsymbol{x}|\mathcal{D}) = \mu_{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) + \sqrt{\frac{\beta\pi}{2}}\sigma_{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D})$, where $\mu_{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D})$ and $\sigma_{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D})$ are the posterior mean and posterior standard deviation respectively. Consider a Monte-Carlo estimation of $\alpha^{\boldsymbol{\theta}}_{\text{rob.}}(\boldsymbol{x}|\mathcal{D}) \equiv \mathbb{E}_{\epsilon\sim\mathcal{N}(\mathbf{0},\sigma_\epsilon^2\boldsymbol{I})}\left[\alpha^{\boldsymbol{\theta}+\epsilon}(\boldsymbol{x}|\mathcal{D})\right]$:

$$\hat{\alpha}^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) = \frac{1}{N_\epsilon}\sum_{j=1}^{N_\epsilon} \alpha^{\boldsymbol{\theta}+\boldsymbol{\epsilon}_j}(\boldsymbol{x}|\mathcal{D})$$

[17] We note that gradient-based algorithms remain applicable upon addition of the $\eta\sigma_n$ term. In our formulation however, we use an evolutionary method which utilises acquisition function values. Consequently, the path followed by the optimiser will be altered based on η samples leading to more robust query locations.

[18] Here we use the common approach for proving stochastic expressions with high probability (see (Tripuraneni et al., 2017), (Zhu & Li, 2016)). Specifically, we show that for any confidence parameter $\delta \in (0,1)$ the stochastic expression under consideration is valid with probability at least $1-\delta$.

where $\boldsymbol{\epsilon}_j$ are i.i.d samples drawn from $\mathcal{N}(\mathbf{0}, \sigma_\epsilon^2 \boldsymbol{I})$. Then, adding and subtracting $\hat{\alpha}^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D})$ gives:

$$\left|\overline{\alpha}^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) - \mathbb{E}_{\epsilon\sim\mathcal{N}(\mathbf{0},\sigma_\epsilon^2\boldsymbol{I})}\left[\alpha^{\boldsymbol{\theta}+\epsilon}(\boldsymbol{x}|\mathcal{D})\right]\right| \leq$$
$$\left|\overline{\alpha}^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) - \hat{\alpha}^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D})\right| + \left|\hat{\alpha}^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) - \mathbb{E}_{\epsilon\sim\mathcal{N}(\mathbf{0},\sigma_\epsilon^2\boldsymbol{I})}\left[\alpha^{\boldsymbol{\theta}+\epsilon}(\boldsymbol{x}|\mathcal{D})\right]\right|.$$

Using the definition of $\hat{\alpha}^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D})$ in the above result yields:

$$\left|\overline{\alpha}^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) - \mathbb{E}_{\epsilon\sim\mathcal{N}(\mathbf{0},\sigma_\epsilon^2\boldsymbol{I})}\left[\alpha^{\boldsymbol{\theta}+\epsilon}(\boldsymbol{x}|\mathcal{D})\right]\right| \leq \tag{5}$$
$$\frac{1}{N_\epsilon}\sum_{j=1}^{N_\epsilon}\left|\overline{\alpha}^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) - \alpha^{\boldsymbol{\theta}+\boldsymbol{\epsilon}_j}(\boldsymbol{x}|\mathcal{D})\right| + \left|\frac{1}{N_\epsilon}\sum_{j=1}^{N_\epsilon}\alpha^{\boldsymbol{\theta}+\boldsymbol{\epsilon}_j}(\boldsymbol{x}|\mathcal{D}) - \mathbb{E}_{\epsilon\sim\mathcal{N}(\mathbf{0},\sigma_\epsilon^2\boldsymbol{I})}\left[\alpha^{\boldsymbol{\theta}+\epsilon}(\boldsymbol{x}|\mathcal{D})\right]\right|$$

Let us now study separately each term in the above result. Applying the Chebyshev inequality for the second term in the above expression, we have that with probability at least $p_1 = 1 - \frac{8\left[\mathbb{E}_{\boldsymbol{\epsilon}}\left[\mu^2_{\boldsymbol{\theta}+\boldsymbol{\epsilon}}(\boldsymbol{x}|\mathcal{D})\right] + \frac{\beta\pi}{2}\mathbb{E}_{\boldsymbol{\epsilon}}\left[\sigma^2_{\boldsymbol{\theta}+\boldsymbol{\epsilon}}(\boldsymbol{x}|\mathcal{D})\right]\right]}{N_\epsilon\rho^2}$:

$$\left|\frac{1}{N_\epsilon}\sum_{j=1}^{N_\epsilon}\alpha^{\boldsymbol{\theta}+\boldsymbol{\epsilon}_j}(\boldsymbol{x}|\mathcal{D}) - \mathbb{E}_{\epsilon\sim\mathcal{N}(\mathbf{0},\sigma_\epsilon^2\boldsymbol{I})}\left[\alpha^{\boldsymbol{\theta}+\epsilon}(\boldsymbol{x}|\mathcal{D})\right]\right| \leq \frac{\rho}{2}. \tag{6}$$

In order to ensure that $p_1 = 1 - \frac{\delta}{2}$, the sample number $\boldsymbol{\epsilon}_j$ should be taken:

$$N_\epsilon = \left\lceil \frac{16\left[\mathbb{E}_{\boldsymbol{\epsilon}}\left[\mu^2_{\boldsymbol{\theta}+\boldsymbol{\epsilon}}(\boldsymbol{x}|\mathcal{D})\right] + \frac{\beta\pi}{2}\mathbb{E}_{\boldsymbol{\epsilon}}\left[\sigma^2_{\boldsymbol{\theta}+\boldsymbol{\epsilon}}(\boldsymbol{x}|\mathcal{D})\right]\right]}{\delta\rho^2} \right\rceil.$$

We will later simplify this expression using the bounds in (4). For now, we restrict our focus on the second term in (5). To bound it, we will establish a bound on $|\overline{\alpha}^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) - \alpha^{\boldsymbol{\theta}+\boldsymbol{\epsilon}_j}(\boldsymbol{x}|\mathcal{D})$. For a small random perturbation $\boldsymbol{\epsilon}_j$ we have (with probability 1):

$$\alpha^{\boldsymbol{\theta}+\boldsymbol{\epsilon}_j}(\boldsymbol{x}|\mathcal{D}) = \alpha^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) + \boldsymbol{\epsilon}_j^\mathsf{T}\nabla_{\boldsymbol{\theta}}\alpha^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) + o(||\boldsymbol{\epsilon_j}||) =$$
$$\alpha^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) + \boldsymbol{\epsilon}_j^\mathsf{T}\nabla_{\boldsymbol{\theta}}\left[\mu_{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) + \sqrt{\frac{\beta\pi}{2}}\sigma_{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D})\right] + o(||\boldsymbol{\epsilon_j}||_2).$$

Let us define

$$\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) = \nabla_{\boldsymbol{\theta}}\left[\mu_{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) + \sqrt{\frac{\beta\pi}{2}}\sigma_{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D})\right],$$

then, using the Cauchy–Schwarz inequality we have:

$$\left|\alpha^{\boldsymbol{\theta}+\boldsymbol{\epsilon}_j}(\boldsymbol{x}|\mathcal{D}) - \alpha^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D})\right| \leq ||\boldsymbol{\epsilon_j}||_2||[\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D})||_2 + o(1)]$$

Since $\boldsymbol{\epsilon}_j \sim \mathcal{N}(0,1)$, then with probability at least $1 - \frac{\delta}{4N_\epsilon}$:

$$||\boldsymbol{\epsilon}_j||_2 \leq 4\sigma_\epsilon\sqrt{p} + 2\sigma_\epsilon\sqrt{\log\frac{4N_\epsilon}{\delta}}$$

Let us assume (and later we will prove the existence of such a bound) that $||\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D})||_2 \leq A_1$. Then, with probability at least $1 - \frac{\delta}{4N_\epsilon}$:

$$\left|\alpha^{\boldsymbol{\theta}+\boldsymbol{\epsilon}_j}(\boldsymbol{x}|\mathcal{D}) - \alpha^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D})\right| \leq \left[4\sigma_\epsilon\sqrt{p} + 2\sigma_\epsilon\sqrt{\log\frac{4N_\epsilon}{\delta}}\right][A_1 + o(1)]$$

On the other hand, for $\overline{\alpha}^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) = \alpha^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) + \eta\sigma_\eta$ with probability at least $1 - \frac{\delta}{4N_\epsilon}$ we have:

$$\left|\overline{\alpha}^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) - \alpha^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D})\right| \leq \Phi^{-1}\left(1 - \frac{\delta}{8N_\epsilon}\right)\sigma_n.$$

where $\Phi(\cdot)$ is the cumulative distribution function for a standard Gaussian random variable. Hence, by choosing $\sigma_{\boldsymbol{\epsilon}} = \min\left\{1, \frac{\Phi^{-1}\left(1-\frac{\delta}{8N_\epsilon}\right)\sigma_n}{\left[4\sqrt{p}+2\sqrt{\log\frac{4N_\epsilon}{\delta}}\right][A_1+o(1)]}\right\}$ with probability at least $1 - \frac{\delta}{2N_\epsilon}$ we have that both $\overline{\alpha}^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D})$ and $\alpha^{\boldsymbol{\theta}+\boldsymbol{\epsilon}_j}(\boldsymbol{x}|\mathcal{D})$ belong to the interval centred at $\alpha^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D})$ of size $\Phi^{-1}\left(1 - \frac{\delta}{8N_\epsilon}\right)\sigma_n$. Therefore, with probability at least $1 - \frac{\delta}{2N_\epsilon}$:

$$\left|\overline{\alpha}^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) - \alpha^{\boldsymbol{\theta}+\boldsymbol{\epsilon}_j}(\boldsymbol{x}|\mathcal{D})\right| \leq 2\Phi^{-1}\left(1 - \frac{\delta}{8N_\epsilon}\right)\sigma_n$$

Hence, by choosing $\sigma_n = \frac{\rho}{4\Phi^{-1}\left(1-\frac{\delta}{8N_\epsilon}\right)}$ we arrive at:

$$\left|\overline{\alpha}^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) - \alpha^{\boldsymbol{\theta}+\boldsymbol{\epsilon}_j}(\boldsymbol{x}|\mathcal{D})\right| \leq \frac{\rho}{2}$$

and, therefore, for the first term in (5) with probability at least $1 - \frac{\delta}{2}$ we have:

$$\frac{1}{N_\epsilon}\sum_{j=1}^{N_\epsilon}\left|\overline{\alpha}^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) - \alpha^{\boldsymbol{\theta}+\boldsymbol{\epsilon}_j}(\boldsymbol{x}|\mathcal{D})\right| \leq \frac{\rho}{2}$$

Combining this result with (6) gives, that with probability at least $1 - \delta$:

$$\left|\overline{\alpha}^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) - \mathbb{E}_{\epsilon\sim\mathcal{N}(\mathbf{0},\sigma_\epsilon^2\boldsymbol{I})}\left[\alpha^{\boldsymbol{\theta}+\epsilon}(\boldsymbol{x}|\mathcal{D})\right]\right| \leq \rho, \qquad \forall\boldsymbol{\theta}, \boldsymbol{x},$$

upon defining:

$$\sigma_n = \frac{\rho}{4\Phi^{-1}\left(1 - \frac{\delta}{8N_\epsilon}\right)},\ \sigma_{\boldsymbol{\epsilon}} = \min\left\{1, \frac{\rho}{8\left[2\sqrt{p} + \sqrt{\log\frac{4N_\epsilon}{\delta}}\right][A_1 + o(1)]}\right\}, \tag{7}$$

with

$$N_\epsilon = \lceil\frac{16\left[\mathbb{E}_{\boldsymbol{\epsilon}}\left[\mu^2_{\boldsymbol{\theta}+\boldsymbol{\epsilon}}(\boldsymbol{x}|\mathcal{D})\right] + \frac{\beta\pi}{2}\mathbb{E}_{\boldsymbol{\epsilon}}\left[\sigma^2_{\boldsymbol{\theta}+\boldsymbol{\epsilon}}(\boldsymbol{x}|\mathcal{D})\right]\right]}{\delta\rho^2}\rceil.$$

Our last step is to prove the existence of a constant A_1 such that $||\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x})||_2 \leq A_1$ and also to simplify these expressions by deriving bounds on $\mathbb{E}_{\boldsymbol{\epsilon}}\left[\mu_{\boldsymbol{\theta}+\boldsymbol{\epsilon}}(\boldsymbol{x}|\mathcal{D})\right]$ and $\mathbb{E}_{\boldsymbol{\epsilon}}\left[\sigma^2_{\boldsymbol{\theta}+\boldsymbol{\epsilon}}(\boldsymbol{x}|\mathcal{D})\right]$. This will be provided as a separate Claim:

Claim: Let the bounds in (4) hold, then there are positive constants A_1, A_2 and A_3, such that

$$||\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x})||_2 \leq A_1, \quad \mathbb{E}_{\boldsymbol{\epsilon}}\left[\mu_{\boldsymbol{\theta}+\boldsymbol{\epsilon}}(\boldsymbol{x}|\mathcal{D})\right] \leq A_2, \quad \mathbb{E}_{\boldsymbol{\epsilon}}\left[\sigma^2_{\boldsymbol{\theta}+\boldsymbol{\epsilon}}(\boldsymbol{x}|\mathcal{D})\right] \leq A_3. \tag{8}$$

Proof. We start with the bound on $||\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x})||_2$. Let us denote for simplicity $\boldsymbol{a}_{\boldsymbol{\theta}} = [k_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}_i)]_{\boldsymbol{x}_i \in \mathcal{D}}$, $\boldsymbol{B}_{\boldsymbol{\theta}} = \left[[k_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}')]_{\boldsymbol{x}\in\mathcal{D}, \boldsymbol{x}'\in\mathcal{D}} + \boldsymbol{I}\right]^{-1}$, $\boldsymbol{y} = [y(\boldsymbol{x})]_{\boldsymbol{x}\in\boldsymbol{\mathcal{D}}}$, $\boldsymbol{m}_{\mathcal{D}} = [m(\boldsymbol{x})]_{\boldsymbol{x}\in\mathcal{D}}$, $m = m(\boldsymbol{x})$, and $k_{\boldsymbol{\theta}} = k_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x})$, then

$$\mu_{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) = \boldsymbol{a}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{B}_{\boldsymbol{\theta}}[\boldsymbol{y} - \boldsymbol{m}_{\mathcal{D}}] + m, \quad \sigma^2_{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) = \boldsymbol{a}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{B}_{\boldsymbol{\theta}} \boldsymbol{a}_{\boldsymbol{\theta}} + k_{\boldsymbol{\theta}}$$

Let us also denote the size of $\mathcal{D}$ as N, then we have:

$$\nabla_{\boldsymbol{\theta}} \mu_{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) = \sum_{i=1}^{N}\sum_{j=1}^{N} \nabla_{\boldsymbol{\theta}}\left[[y_j - m_j][\boldsymbol{a}_{\boldsymbol{\theta}}]_i [\boldsymbol{B}_{\boldsymbol{\theta}}]_{ij}\right] =$$
$$\sum_{i=1}^{N}\sum_{j=1}^{N}\left[[y_j - m_j][\boldsymbol{B}_{\boldsymbol{\theta}}]_{ij} \nabla_{\boldsymbol{\theta}}[\boldsymbol{a}_{\boldsymbol{\theta}}]_i\right] + \sum_{i=1}^{N}\sum_{j=1}^{N}\left[[y_j - m_j][\boldsymbol{a}_{\boldsymbol{\theta}}]_i \nabla_{\boldsymbol{\theta}}[[\boldsymbol{B}_{\boldsymbol{\theta}}]_{ij}]\right].$$

Consider each term in this expression separately:

$$\left|\left|\sum_{i=1}^{N}\sum_{j=1}^{N}[y_j - m_j][\boldsymbol{B}_{\boldsymbol{\theta}}]_{ij}\nabla_{\boldsymbol{\theta}}[\boldsymbol{a}_{\boldsymbol{\theta}}]_i\right|\right|_2 = \left|\left|\sum_{i=1}^{N}[\boldsymbol{B}_{\boldsymbol{\theta}}[\boldsymbol{y} - \boldsymbol{m}_{\mathcal{D}}]]_i \nabla_{\boldsymbol{\theta}}[\boldsymbol{a}_{\boldsymbol{\theta}}]_i\right|\right|_2 \leq$$
$$\sum_{i=1}^{N} ||\boldsymbol{B}_{\boldsymbol{\theta}}(i,:)||_2 ||\boldsymbol{y} - \boldsymbol{m}_{\mathcal{D}}||_2 \, ||\nabla_{\boldsymbol{\theta}}[\boldsymbol{a}_{\boldsymbol{\theta}}]_i||_2 \,.$$

Using $|y| \leq C$ and $|m(\boldsymbol{x})| \leq M_4$, we have $||\boldsymbol{y} - \boldsymbol{m}_{\mathcal{D}}||_2 \leq (C + M_4)\sqrt{N}$ and $||\nabla_{\boldsymbol{\theta}}[\boldsymbol{a}(\boldsymbol{\theta})]_i||_2 \leq M_2$ and as such:

$$\left|\left|\sum_{i=1}^{N}\sum_{j=1}^{N}[y_j - m_j][\boldsymbol{B}_{\boldsymbol{\theta}}]_{ij}\nabla_{\boldsymbol{\theta}}[\boldsymbol{a}_{\boldsymbol{\theta}}]_i\right|\right|_2 \leq (C + M_4)\sqrt{N}\sum_{i=1}^{N} ||\boldsymbol{B}(\boldsymbol{\theta})(i,:)||_2 \, ||\nabla_{\boldsymbol{\theta}}[\boldsymbol{a}_{\boldsymbol{\theta}}]_i||_2 \leq \tag{9}$$
$$(C + M_4)\sqrt{N}||\boldsymbol{B}_{\boldsymbol{\theta}}||_F \sum_{i=1}^{N} ||\nabla_{\boldsymbol{\theta}}[\boldsymbol{a}_{\boldsymbol{\theta}}]_i||_2 \leq (C + M_4)\sqrt{N}\sqrt{\mathrm{rank}(\boldsymbol{B}_{\boldsymbol{\theta}})}||\boldsymbol{B}_{\boldsymbol{\theta}}||_2 \sum_{i=1}^{N} ||\nabla_{\boldsymbol{\theta}}[\boldsymbol{a}_{\boldsymbol{\theta}}]_i||_2 \leq$$
$$\frac{(C + M_4)N}{\sigma_n^2} \sum_{i=1}^{N} ||\nabla_{\boldsymbol{\theta}}[\boldsymbol{a}(\boldsymbol{\theta})]_i||_2 = \frac{(C + M_4)N^2 M_2}{\sigma_n^2}$$

Now, let us consider the second term in the expression for the posterior mean:

$$\sum_{i=1}^{N}\sum_{j=1}^{N}[y_j - m_j][\boldsymbol{a}_{\boldsymbol{\theta}}]_i \nabla_{\boldsymbol{\theta}}\left[[\boldsymbol{B}_{\boldsymbol{\theta}}]_{ij}\right] = \sum_{i=1}^{N}[\boldsymbol{a}_{\boldsymbol{\theta}}]_i \left[\sum_{j=1}[y_j - m_j]\nabla_{\boldsymbol{\theta}}\left[[\boldsymbol{B}_{\boldsymbol{\theta}}]_{ij}\right]\right].$$

Notice, that the gradient expression above is presented in the form of a vector:

$$\nabla_{\boldsymbol{\theta}}\left[[\boldsymbol{B}_{\boldsymbol{\theta}}]_{ij}\right] = \begin{bmatrix} \frac{\partial}{\partial\theta_1}\left[\boldsymbol{K}_{\boldsymbol{\theta}} + \sigma_n \boldsymbol{I}\right]^{-1}_{ij}, \\ \vdots \\ \frac{\partial}{\partial\theta_p}\left[\boldsymbol{K}_{\boldsymbol{\theta}} + \sigma_n \boldsymbol{I}\right]^{-1}_{ij} \end{bmatrix},$$

where we use the notation $\boldsymbol{K}_{\boldsymbol{\theta}} = [k_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_j)]^{N,N}_{i=1,j=1}$. For the r^{th} component we have

$$\frac{\partial}{\partial\theta_r}\left[\boldsymbol{K}_{\boldsymbol{\theta}} + \sigma_n \boldsymbol{I}\right]^{-1}_{ij} = \left[-\left[\boldsymbol{K}_{\boldsymbol{\theta}} + \sigma_n \boldsymbol{I}\right]^{-1}\frac{\partial}{\partial\theta_r}\left[\boldsymbol{K}_{\boldsymbol{\theta}} + \sigma_n \boldsymbol{I}\right]\left[\boldsymbol{K}_{\boldsymbol{\theta}} + \sigma_n \boldsymbol{I}\right]^{-1}\right]_{ij} \tag{10}$$

Now we can study the gradient of the second term in the posterior mean expression,

$$\left|\left|\sum_{i=1}^{N}[\boldsymbol{a}_{\boldsymbol{\theta}}]_i \left[\sum_{j=1}[y_j - m_j]\nabla_{\boldsymbol{\theta}}\left[[\boldsymbol{B}_{\boldsymbol{\theta}}]_{ij}\right]\right]\right|\right|_2 \leq \sum_{i=1}^{N}|[\boldsymbol{a}_{\boldsymbol{\theta}}]_i| \left[\sum_{j=1}^{N}||\boldsymbol{y} - \boldsymbol{m}_{\mathcal{D}}||_2 \, ||\nabla_{\boldsymbol{\theta}}\left[[\boldsymbol{B}_{\boldsymbol{\theta}}]_{ij}\right]||_2\right] \leq$$
$$(C + M_4)\sqrt{N}M_1 \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{r=1}^{p}\left|\frac{\partial}{\partial\theta_r}\left[\boldsymbol{K}_{\boldsymbol{\theta}} + \sigma_n \boldsymbol{I}\right]^{-1}_{ij}\right|.$$

Using result (10) in the above expression we now have

$$\left|\left|\sum_{i=1}^{N}[\boldsymbol{a}_{\boldsymbol{\theta}}]_i \left[\sum_{j=1}[y_j - m_j]\nabla_{\boldsymbol{\theta}}\left[[\boldsymbol{B}_{\boldsymbol{\theta}}]_{ij}\right]\right]\right|\right|_2 \leq$$
$$(C + M_4)\sqrt{N}M_1 \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{r=1}^{p}\left|\frac{\partial}{\partial\theta_r}\left[\boldsymbol{K}_{\boldsymbol{\theta}} + \sigma_n \boldsymbol{I}\right]^{-1}_{ij}\right| \leq$$
$$(C + M_4)\sqrt{N}M_1 \sum_{r=1}^{p}\sum_{i=1}^{N}\sum_{j=1}^{N}\left|\frac{\partial}{\partial\theta_r}\left[\boldsymbol{K}_{\boldsymbol{\theta}} + \sigma_n \boldsymbol{I}\right]^{-1}_{ij}\right| \leq$$
$$(C + M_4)N\sqrt{N}M_1 \times \sum_{r=1}^{p}\left|\left|\left[\boldsymbol{K}_{\boldsymbol{\theta}} + \sigma_n \boldsymbol{I}\right]^{-1}\frac{\partial}{\partial\theta_r}\left[\boldsymbol{K}_{\boldsymbol{\theta}} + \sigma_n \boldsymbol{I}\right]\left[\boldsymbol{K}_{\boldsymbol{\theta}} + \sigma_n \boldsymbol{I}\right]^{-1}\right|\right|_F,$$

where we used that $\sum_{i=1}^{N}\sum_{j=1}^{N}|\boldsymbol{C}_{ij}| \leq N||\boldsymbol{C}||_F$ for any arbitrary matrix $\boldsymbol{C} \in \mathbb{R}^{N\times N}$. Because $\frac{\partial}{\partial\theta_r}[\boldsymbol{K_\theta} + \sigma_n\boldsymbol{I}] = \frac{\partial}{\partial\theta_r}\boldsymbol{K_\theta}$

$$\frac{\left|\left|\sum_{i=1}^{N}[\boldsymbol{a_\theta}]_i\left[\sum_{j=1}[y_j - m_j]\nabla_{\boldsymbol{\theta}}\left[[\boldsymbol{B_\theta}]_{ij}\right]\right]\right|\right|_2}{(C+M_4)N\sqrt{N}M_1} \leq$$
$$\sum_{r=1}^{p}\left|\left|[\boldsymbol{K_\theta}+\sigma_n\boldsymbol{I}]^{-1}\frac{\partial}{\partial\theta_r}\boldsymbol{K_\theta}\,[\boldsymbol{K_\theta}+\sigma_n\boldsymbol{I}]^{-1}\right|\right|_F \leq$$
$$\sqrt{N}\sum_{r=1}^{p}\left|\left|[\boldsymbol{K_\theta}+\sigma_n\boldsymbol{I}]^{-1}\frac{\partial}{\partial\theta_r}\boldsymbol{K_\theta}\,[\boldsymbol{K_\theta}+\sigma_n\boldsymbol{I}]^{-1}\right|\right|_2$$

and using the properties of the matrix 2-norm $||\cdot||_2$

$$\left|\left|[\boldsymbol{K_\theta}+\sigma_n\boldsymbol{I}]^{-1}\right|\right|_2 \leq \frac{1}{\sigma_n^2}$$

Hence,

$$\frac{\left|\left|\sum_{i=1}^{N}[\boldsymbol{a_\theta}]_i\left[\sum_{j=1}[y_j - m_j]\nabla_{\boldsymbol{\theta}}\left[[\boldsymbol{B_\theta}]_{ij}\right]\right]\right|\right|_2}{(C+M_4)N^2M_1} \leq \tag{11}$$
$$\sum_{r=1}^{p}\left|\left|[\boldsymbol{K_\theta}+\sigma_n\boldsymbol{I}]^{-1}\right|\right|_2\left|\left|\frac{\partial}{\partial\theta_r}\boldsymbol{K_\theta}\right|\right|_2\left|\left|[\boldsymbol{K_\theta}+\sigma_n\boldsymbol{I}]^{-1}\right|\right|_2 \leq \frac{1}{\sigma_n^4}\sum_{r=1}^{p}\left|\left|\frac{\partial}{\partial\theta_r}\boldsymbol{K_\theta}\right|\right|_2.$$

Let us study the last term in the expression above. Using $\sqrt{c_1^2+\ldots+c_R^2} \leq |c_1|+\ldots+|c_R|$ for any set of real numbers $c_1,\ldots,c_R \in \mathbb{R}$ we have

$$\left|\left|\frac{\partial}{\partial\theta_r}\boldsymbol{K_\theta}\right|\right|_2 = \left|\left|\begin{bmatrix} \frac{\partial}{\partial\theta_r}k_{\boldsymbol{\theta}}(\boldsymbol{x}_1,\boldsymbol{x}_1), & \ldots & \frac{\partial}{\partial\theta_r}k_{\boldsymbol{\theta}}(\boldsymbol{x}_1,\boldsymbol{x}_N) \\ \vdots & \vdots & \vdots \\ \frac{\partial}{\partial\theta_r}k_{\boldsymbol{\theta}}(\boldsymbol{x}_N,\boldsymbol{x}_1), & \ldots & \frac{\partial}{\partial\theta_r}k_{\boldsymbol{\theta}}(\boldsymbol{x}_N,\boldsymbol{x}_N) \end{bmatrix} l\right|\right|_2 \leq$$
$$\left|\left|\begin{bmatrix} \frac{\partial}{\partial\theta_r}k_{\boldsymbol{\theta}}(\boldsymbol{x}_1,\boldsymbol{x}_1), & \ldots & \frac{\partial}{\partial\theta_r}k_{\boldsymbol{\theta}}(\boldsymbol{x}_1,\boldsymbol{x}_N) \\ \vdots & \vdots & \vdots \\ \frac{\partial}{\partial\theta_r}k_{\boldsymbol{\theta}}(\boldsymbol{x}_N,\boldsymbol{x}_1), & \ldots & \frac{\partial}{\partial\theta_r}k_{\boldsymbol{\theta}}(\boldsymbol{x}_N,\boldsymbol{x}_N) \end{bmatrix}\right|\right|_F =$$
$$\sqrt{\sum_{i=1}^{N}\sum_{j=1}^{N}\left[\frac{\partial}{\partial\theta_r}k_{\boldsymbol{\theta}}(\boldsymbol{x}_i,\boldsymbol{x}_j)\right]^2} \leq \sum_{i=1}^{N}\sum_{j=1}^{N}\left|\frac{\partial}{\partial\theta_r}k_{\boldsymbol{\theta}}(\boldsymbol{x}_i,\boldsymbol{x}_j)\right|.$$

Substituting this expression in (11) gives us

$$\frac{\left|\left|\sum_{i=1}^{N}[\boldsymbol{a_\theta}]_i\left[\sum_{j=1}[y_j - m_j]\nabla_{\boldsymbol{\theta}}\left[[\boldsymbol{B_\theta}]_{ij}\right]\right]\right|\right|_2}{(C+M_4)N^2M_1} \leq \frac{1}{\sigma_n^4}\sum_{r=1}^{d}\sum_{i=1}^{N}\sum_{j=1}^{N}\left|\frac{\partial}{\partial\theta_r}k_{\boldsymbol{\theta}}(\boldsymbol{x}_i,\boldsymbol{x}_j)\right| \leq \tag{12}$$
$$\frac{\sqrt{p}}{\sigma_n^4}\sum_{i=1}^{N}\sum_{j=1}^{N}||\nabla_{\boldsymbol{\theta}}k_{\boldsymbol{\theta}}(\boldsymbol{x}_i,\boldsymbol{x}_j)||_2 \leq \frac{N^2\sqrt{p}M_2}{\sigma_n^4}.$$

Hence, combining results (9) and (12) we have

$$||\nabla_{\boldsymbol{\theta}}\mu_{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D})||_2 \leq \frac{(C+M_4)N^2M_2}{\sigma_n^2}\left[1+\frac{N^2M_1\sqrt{p}}{\sigma_n^2}\right]. \tag{13}$$

Now, let us focus on the gradient of the posterior standard deviation,

$$\nabla_{\boldsymbol{\theta}}\sigma_{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) = \nabla_{\boldsymbol{\theta}}\left[\sqrt{k_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{x}) - \boldsymbol{a}_{\boldsymbol{\theta}}^{\intercal}[\boldsymbol{K}_{\boldsymbol{\theta}}+\sigma_n^2\mathbf{I}]^{-1}\boldsymbol{a}_{\boldsymbol{\theta}}}\right] =$$

$$\frac{1}{2\sigma_{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D})}\nabla_{\boldsymbol{\theta}}\left[k_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{x}) - \boldsymbol{a}_{\boldsymbol{\theta}}^{\intercal}[\boldsymbol{K}_{\boldsymbol{\theta}}+\sigma_n^2\mathbf{I}]^{-1}\boldsymbol{a}_{\boldsymbol{\theta}}\right] = \frac{1}{2\sigma_{\boldsymbol{\theta}}(\boldsymbol{x})}\left[\nabla_{\boldsymbol{\theta}}k_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{x}) - \nabla_{\boldsymbol{\theta}}\left[\boldsymbol{a}_{\boldsymbol{\theta}}^{\intercal}[\boldsymbol{K}_{\boldsymbol{\theta}}+\sigma_n^2\mathbf{I}]^{-1}\boldsymbol{a}_{\boldsymbol{\theta}}\right]\right].$$

Let us study the second gradient expression. Using our notation we have

$$\boldsymbol{a}_{\boldsymbol{\theta}}^{\intercal}[\boldsymbol{K}_{\boldsymbol{\theta}}+\sigma_n^2\mathbf{I}]^{-1}\boldsymbol{a}_{\boldsymbol{\theta}} = \boldsymbol{a}_{\boldsymbol{\theta}}^{\intercal}\boldsymbol{B}_{\boldsymbol{\theta}}\boldsymbol{a}_{\boldsymbol{\theta}} = \sum_{i=1}^{N}\sum_{j=1}^{N}[\boldsymbol{a}_{\boldsymbol{\theta}}]_i[\boldsymbol{a}_{\boldsymbol{\theta}}]_j\left[\boldsymbol{B}_{\boldsymbol{\theta}}\right]_{ij}.$$

Hence, for the gradient,

$$\nabla_{\boldsymbol{\theta}}\left[\boldsymbol{a}_{\boldsymbol{\theta}}^{\intercal}\boldsymbol{B}_{\boldsymbol{\theta}}\boldsymbol{a}_{\boldsymbol{\theta}}\right] = \sum_{i=1}^{N}\sum_{j=1}^{N}\nabla_{\boldsymbol{\theta}}\left[[\boldsymbol{a}_{\boldsymbol{\theta}}]_i[\boldsymbol{a}_{\boldsymbol{\theta}}]_j\left[\boldsymbol{B}_{\boldsymbol{\theta}}\right]_{ij}\right] =$$

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\nabla_{\boldsymbol{\theta}}\left[[\boldsymbol{a}_{\boldsymbol{\theta}}]_i\right][\boldsymbol{a}_{\boldsymbol{\theta}}]_j\left[\boldsymbol{B}_{\boldsymbol{\theta}}\right]_{ij} + \sum_{i=1}^{N}\sum_{j=1}^{N}\nabla_{\boldsymbol{\theta}}\left[[\boldsymbol{a}_{\boldsymbol{\theta}}]_j\right][\boldsymbol{a}_{\boldsymbol{\theta}}]_i\left[\boldsymbol{B}_{\boldsymbol{\theta}}\right]_{ij} +$$

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\nabla_{\boldsymbol{\theta}}\left[\left[\boldsymbol{B}_{\boldsymbol{\theta}}\right]_{ij}\right][\boldsymbol{a}_{\boldsymbol{\theta}}]_i[\boldsymbol{a}_{\boldsymbol{\theta}}]_j.$$

and for the norm of the above expression we have

$$\left|\left|\nabla_{\boldsymbol{\theta}}\left[\boldsymbol{a}_{\boldsymbol{\theta}}^{\intercal}\boldsymbol{B}_{\boldsymbol{\theta}}\boldsymbol{a}_{\boldsymbol{\theta}}\right]\right|\right|_2 = \sum_{i=1}^{N}\sum_{j=1}^{N}\nabla_{\boldsymbol{\theta}}\left[[\boldsymbol{a}_{\boldsymbol{\theta}}]_i[\boldsymbol{a}_{\boldsymbol{\theta}}]_j\left[\boldsymbol{B}_{\boldsymbol{\theta}}\right]_{ij}\right] =$$

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\left|\left|\nabla_{\boldsymbol{\theta}}\left[\left[\boldsymbol{B}_{\boldsymbol{\theta}}\right]_{ij}\right]\right|\right|_2|[\boldsymbol{a}_{\boldsymbol{\theta}}]_i[\boldsymbol{a}_{\boldsymbol{\theta}}]_j| + \sum_{i=1}^{N}\sum_{j=1}^{N}||\nabla_{\boldsymbol{\theta}}||_2\left|[[\boldsymbol{a}_{\boldsymbol{\theta}}]_i]\left[\boldsymbol{a}_{\boldsymbol{\theta}}\right]_j\left[\boldsymbol{B}_{\boldsymbol{\theta}}\right]_{ij}\right| +$$

$$\sum_{i=1}^{N}\sum_{j=1}^{N}||\nabla_{\boldsymbol{\theta}}\left[[\boldsymbol{a}_{\boldsymbol{\theta}}]_j\right]||_2\left|[\boldsymbol{a}_{\boldsymbol{\theta}}]_i\left[\boldsymbol{B}_{\boldsymbol{\theta}}\right]_{ij}\right|.$$

Let us now bound each term in this expression:

1. The first term:

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\left|\left|\nabla_{\boldsymbol{\theta}}\left[\left[\boldsymbol{B}_{\boldsymbol{\theta}}\right]_{ij}\right]\right|\right|_2|[\boldsymbol{a}_{\boldsymbol{\theta}}]_i[\boldsymbol{a}_{\boldsymbol{\theta}}]_j| \leq$$

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\left|\left|\nabla_{\boldsymbol{\theta}}\left[\left[\boldsymbol{B}_{\boldsymbol{\theta}}\right]_{ij}\right]\right|\right|_2||\boldsymbol{a}_{\boldsymbol{\theta}}||_2||\boldsymbol{a}_{\boldsymbol{\theta}}||_2 \leq M_1^2\sum_{i=1}^{N}\sum_{j=1}^{N}\left|\left|\nabla_{\boldsymbol{\theta}}\left[\left[\boldsymbol{B}_{\boldsymbol{\theta}}\right]_{ij}\right]\right|\right|_2$$

Using the previous bound for $\left|\left|\nabla_{\boldsymbol{\theta}}\left[[\boldsymbol{B}_{\boldsymbol{\theta}}]_{ij}\right]\right|\right|_2$ we have:

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\left|\left|\nabla_{\boldsymbol{\theta}}\left[[\boldsymbol{B}_{\boldsymbol{\theta}}]_{ij}\right]\right|\right|_2|[\boldsymbol{a}_{\boldsymbol{\theta}}]_i[\boldsymbol{a}_{\boldsymbol{\theta}}]_j| \leq M_1^2\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{r=1}^{p}\left|\frac{\partial}{\partial\theta_r}\left[\boldsymbol{K}_{\boldsymbol{\theta}}+\sigma_n^2\mathbf{I}\right]^{-1}_{ij}\right| =$$
$$NM_1^2\sum_{r=1}^{p}\left|\left|[\boldsymbol{K}_{\boldsymbol{\theta}}+\sigma_n\boldsymbol{I}]^{-1}\frac{\partial}{\partial\theta_r}\boldsymbol{K}_{\boldsymbol{\theta}}\left[\boldsymbol{K}_{\boldsymbol{\theta}}+\sigma_n\boldsymbol{I}\right]^{-1}\right|\right|_F \leq$$
$$N^{\frac{3}{2}}M_1^2\sum_{r=1}^{p}\left|\left|[\boldsymbol{K}_{\boldsymbol{\theta}}+\sigma_n\boldsymbol{I}]^{-1}\frac{\partial}{\partial\theta_r}\boldsymbol{K}_{\boldsymbol{\theta}}\left[\boldsymbol{K}_{\boldsymbol{\theta}}+\sigma_n\boldsymbol{I}\right]^{-1}\right|\right|_2$$

Since $\left|\left|[\boldsymbol{K}_{\boldsymbol{\theta}}+\sigma_n\boldsymbol{I}]^{-1}\right|\right|_2 \leq \frac{1}{\sigma_n^2}$ we have:

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\left|\left|\nabla_{\boldsymbol{\theta}}\left[[\boldsymbol{B}_{\boldsymbol{\theta}}]_{ij}\right]\right|\right|_2|[\boldsymbol{a}_{\boldsymbol{\theta}}]_i[\boldsymbol{a}_{\boldsymbol{\theta}}]_j| \leq \frac{N\sqrt{N}M_1^2}{\sigma_n^4}\sum_{r=1}^{p}\sum_{i=1}^{N}\sum_{j=1}^{N}\left|\frac{\partial}{\partial\theta_r}k_{\boldsymbol{\theta}}(\boldsymbol{x}_i,\boldsymbol{x}_j)\right|.$$

Using $\sum_{r=1}^{p}\sum_{i=1}^{N}\sum_{j=1}^{N}\left|\frac{\partial}{\partial\theta_r}k_{\boldsymbol{\theta}}(\boldsymbol{x}_i,\boldsymbol{x}_j)\right| = \sqrt{p}\sum_{i=1}^{N}\sum_{j=1}^{N}||\nabla_{\boldsymbol{\theta}}k_{\boldsymbol{\theta}}(\boldsymbol{x}_i,\boldsymbol{x}_j)||_2 \leq N^2\sqrt{p}M_2$, we have:

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\left|\left|\nabla_{\boldsymbol{\theta}}\left[[\boldsymbol{B}_{\boldsymbol{\theta}}]_{ij}\right]\right|\right|_2|[\boldsymbol{a}_{\boldsymbol{\theta}}]_i[\boldsymbol{a}_{\boldsymbol{\theta}}]_j| \leq \frac{N^{\frac{7}{2}}\sqrt{p}M_1^2M_2}{\sigma_n^4}$$

2. The second and the third terms are identical with respect to the bounding strategy,

$$\sum_{i=1}^{N}\sum_{j=1}^{N}||\nabla_{\boldsymbol{\theta}}\left[[\boldsymbol{a}_{\boldsymbol{\theta}}]_i\right]||_2\left|[\boldsymbol{a}_{\boldsymbol{\theta}}]_j\left[\boldsymbol{B}_{\boldsymbol{\theta}}\right]_{ij}\right| = \sum_{i=1}^{N}|\boldsymbol{B}_{\boldsymbol{\theta}}(i,:)\boldsymbol{a}_{\boldsymbol{\theta}}|\,||\nabla_{\boldsymbol{\theta}}\left[[\boldsymbol{a}_{\boldsymbol{\theta}}]_i\right]||_2 \leq$$
$$\sum_{i=1}^{N}||\boldsymbol{B}_{\boldsymbol{\theta}}(i,:)||_2\,||\boldsymbol{a}_{\boldsymbol{\theta}}||_2\,||\nabla_{\boldsymbol{\theta}}\left[[\boldsymbol{a}_{\boldsymbol{\theta}}]_i\right]||_2 \leq ||\boldsymbol{B}_{\boldsymbol{\theta}}||_F||\boldsymbol{a}_{\boldsymbol{\theta}}||_2\sum_{i=1}^{N}||\nabla_{\boldsymbol{\theta}}\left[[\boldsymbol{a}_{\boldsymbol{\theta}}]_i\right]||_2,$$

since $||\boldsymbol{B}_{\boldsymbol{\theta}}||_F \leq \sqrt{\text{rank}(\boldsymbol{B}_{\boldsymbol{\theta}})}||\boldsymbol{B}_{\boldsymbol{\theta}}||_2 \leq \frac{\sqrt{N}}{\sigma_n^2}$. Hence,

$$\sum_{i=1}^{N}\sum_{j=1}^{N}||\nabla_{\boldsymbol{\theta}}\left[[\boldsymbol{a}_{\boldsymbol{\theta}}]_i\right]||_2\left|[\boldsymbol{a}_{\boldsymbol{\theta}}]_j\left[\boldsymbol{B}_{\boldsymbol{\theta}}\right]_{ij}\right| \leq \frac{N\sqrt{N}M_1M_2}{\sigma_n^2}.$$

Combining these results and using $||\nabla_{\boldsymbol{\theta}}k_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{x})|| \leq M_2$, $|\sigma_{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D})| \geq k_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{x}) \geq M_0$, we have

$$||\nabla_{\boldsymbol{\theta}}\left[\sigma_{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D})\right]||_2 \leq \frac{N\sqrt{N}M_1M_2}{2\sigma_n^2M_0}\left[\frac{N^2\sqrt{p}M_1}{\sigma_n^2}+2\right] \tag{14}$$

Hence, combining (13) and (14) we have

$$||\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D})||_2 \leq ||\nabla_{\boldsymbol{\theta}}\mu_{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D})||_2 + \sqrt{\frac{\beta\pi}{2}}\,||\nabla_{\boldsymbol{\theta}}\left[\sigma_{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D})\right]||_2 \leq$$
$$\frac{(C+M_4)N^2M_2}{\sigma_n^2}\left[1+\frac{N^2M_1\sqrt{p}}{\sigma_n^2}\right]+\sqrt{\frac{\beta\pi}{2}}\frac{N\sqrt{N}M_1M_2}{2\sigma_n^2M_0}\left[\frac{N^2\sqrt{p}M_1}{\sigma_n^2}+2\right] \triangleq A_1.$$

Now, we are ready to bound the other two terms in the claim:

$$\mu^2_{\boldsymbol{\theta}+\boldsymbol{\epsilon}}(\boldsymbol{x}|\mathcal{D}) \leq 2\left[\boldsymbol{a}^{\mathsf{T}}_{\boldsymbol{\theta}+\boldsymbol{\epsilon}}\boldsymbol{B}_{\boldsymbol{\theta}+\boldsymbol{\epsilon}}[\boldsymbol{y}-\boldsymbol{m}_{\mathcal{D}}]\right]^2 + 2|m|^2 \leq 2\frac{(C+M_4)^2M_1^2}{\sigma_n^4} + 2M_4^2$$

Therefore, for $\mathbb{E}_{\boldsymbol{\epsilon}}\left[\mu^2_{\boldsymbol{\theta}+\boldsymbol{\epsilon}}(\boldsymbol{x}|\mathcal{D})\right]$ we have

$$\mathbb{E}_{\boldsymbol{\epsilon}}\left[\mu^2_{\boldsymbol{\theta}+\boldsymbol{\epsilon}}(\boldsymbol{x}|\mathcal{D})\right] \leq 2\frac{(C+M_4)^2M_1^2}{\sigma_n^4} + 2M_4^2 \triangleq A_2.$$

Finally, for the posterior mean

$$\sigma^2_{\boldsymbol{\theta}+\boldsymbol{\epsilon}}(\boldsymbol{x}|\mathcal{D}) \leq k_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{x}) + \boldsymbol{a}^{\mathsf{T}}_{\boldsymbol{\theta}+\boldsymbol{\epsilon}}\boldsymbol{B}_{\boldsymbol{\theta}+\boldsymbol{\epsilon}}\boldsymbol{a}_{\boldsymbol{\theta}+\boldsymbol{\epsilon}} \leq M_1 + \frac{M_1^2}{\sigma_n^2}$$

Therefore, for $\mathbb{E}_{\boldsymbol{\epsilon}}\left[\sigma^2_{\boldsymbol{\theta}+\boldsymbol{\epsilon}}(\boldsymbol{x}|\mathcal{D})\right]$ we have

$$\mathbb{E}_{\boldsymbol{\epsilon}}\left[\sigma^2_{\boldsymbol{\theta}+\boldsymbol{\epsilon}}(\boldsymbol{x}|\mathcal{D})\right] \leq M_1 + \frac{M_1^2}{\sigma_n^2} \triangleq A_3.$$

This finishes the proof of the claim. □

Equipped with these results, we can further simplify the expressions (7):

$$\sigma_n = \frac{\rho}{4\Phi^{-1}\left(1-\frac{\delta}{8N_\epsilon}\right)}, \quad \sigma_{\boldsymbol{\epsilon}} = \min\left\{1, \frac{\rho}{8\left[2\sqrt{p}+\sqrt{\log\frac{4N_\epsilon}{\delta}}\right][A_1+o(1)]}\right\},$$

with

$$N_\epsilon = \left\lceil \frac{16\left[A_2+\frac{\beta\pi}{2}A_3\right]}{\delta\rho^2} \right\rceil.$$

This finishes the proof of the lemma.

As such, we may now implement robust formulations of acquisition functions using only the GP predictive mean and variance.

Appendix C. Statistical Hypothesis Tests for Heteroscedasticity

In this section we present the full results for the statistical hypothesis testing using Levene's test and the Fligner-Killeen test in Table 5, Table 6, Table 7, Table 8, Table 9 and Table 10 for the Boston, breast cancer, diabetes, digits, iris and wine datasets respectively.

Appendix D. Task-level Results Breakdown

In this section we present the full task-level breakdown of the results with each metric, data set and model combination for each black-box optimiser summarised with the mean and variance achieved across 20 seeds. We show a summary plot in Table 11.

Table 5: Heteroscedasticity tests on tasks involving `Boston` data set.

Data set	Model	Metric	Fligner Statistic	p-value	Levene Statistic	p-value
Boston	DT	mae	73.51	**0.01327**	1.752	**1.900e-03**
	MLP-adam	mae	336.3	**1.737e-44**	14.4	**3.611e-65**
	MLP-SGD	mae	272.6	**8.694e-33**	6.561	**1.480e-29**
	RF	mae	28.79	0.9906	0.6768	0.9537
	SVM	mae	48.08	0.5106	0.9612	0.5508
	ada	mae	218.7	**2.692e-23**	13.59	**5.542e-62**
	kNN	mae	33.15	0.9597	0.619	0.98
	lasso	mae	30.4	0.983	0.6091	0.983
	linear	mae	16.17	1	0.251	1
	DT	mse	60.75	0.1211	1.33	0.07387
	MLP-adam	mse	387	**4.504e-54**	15.32	**1.147e-68**
	MLP-SGD	mse	353.2	**1.185e-47**	8.239	**3.548e-38**
	RF	mse	35.59	0.9242	0.8985	0.6692
	SVM	mse	25.01	0.9983	0.4491	0.9996
	ada	mse	249.1	**1.398e-28**	14.4	**3.682e-65**
	kNN	mse	27.75	0.9938	0.8247	0.7951
	lasso	mse	31.38	0.9764	0.5397	0.9955
	linear	mse	16.67	1	0.1726	1

Table 6: Heteroscedasticity tests on tasks involving `Breast cancer` (BC) data set.

Data set	Model	Metric	Fligner Statistic	p-value	Levene Statistic	p-value
BC	DT	acc	97.79	**4.302e-05**	4.62	**6.650e-19**
	MLP-adam	acc	133	**1.113e-09**	2.939	**1.923e-09**
	MLP-SGD	acc	116.8	**1.854e-07**	2.469	**6.495e-07**
	RF	acc	154.9	**6.469e-13**	6.661	**4.353e-30**
	SVM	acc	20.7	0.9999	0.3995	0.9999
	ada	acc	272.5	**9.178e-33**	13.57	**6.582e-62**
	kNN	acc	33.16	0.9596	0.5519	0.9941
	lasso	acc	20.78	0.9999	0.4291	0.9998
	linear	acc	21.15	0.9998	0.4545	0.9995
	DT	nll	260.5	**1.280e-30**	9.52	**2.294e-44**
	MLP-adam	nll	166.6	**1.008e-14**	3.643	**2.247e-13**
	MLP-SGD	nll	141.2	**7.115e-11**	2.669	**5.661e-08**
	RF	nll	185.8	**8.495e-18**	7.553	**1.013e-34**
	SVM	nll	76.98	**6.526e-03**	1.707	**2.970e-03**
	ada	nll	142	**5.458e-11**	4.283	**5.274e-17**
	kNN	nll	125.7	**1.155e-08**	4.337	**2.635e-17**
	lasso	nll	71.41	**0.02**	1.011	0.4565
	linear	nll	18.55	1	0.2714	1

Table 7: Heteroscedasticity tests on tasks involving `diabetes` data set.

Dataset	Model	Metric	Fligner Statistic	p-value	Levene Statistic	p-value
Diabetes	DT	mae	56.52	0.2146	1.131	0.2601
	MLP-adam	mae	74.64	**0.01059**	2.573	**1.847e-07**
	MLP-SGD	mae	191.3	**1.062e-18**	17.87	**8.498e-78**
	RF	mae	79.38	**3.898e-03**	1.558	**0.01174**
	SVM	mae	2.436	1	1.810e-04	1
	ada	mae	179.8	**7.883e-17**	7.542	**1.154e-34**
	kNN	mae	67.48	**0.04106**	2.101	**4.747e-05**
	lasso	mae	176.2	**2.950e-16**	4.75	**1.225e-19**
	linear	mae	206	**3.792e-21**	5.714	**5.490e-25**
	DT	mse	44.52	0.6551	0.8264	0.7925
	MLP-adam	mse	100.4	**2.109e-05**	3.582	**4.951e-13**
	MLP-SGD	mse	202.9	**1.257e-20**	14.31	**7.960e-65**
	RF	mse	37.1	0.8938	0.8063	0.8224
	SVM	mse	4.004	1	4.740e-04	1
	ada	mse	189	**2.510e-18**	7.348	**1.138e-33**
	kNN	mse	88.62	**4.545e-04**	2.964	**1.407e-09**
	lasso	mse	257.6	**4.341e-30**	10.86	**1.637e-50**
	linear	mse	278.2	**8.540e-34**	10.01	**1.216e-46**

Table 8: Heteroscedasticity tests on tasks involving `digits` data set.

Data set	Model	Metric	Fligner Statistic	p-value	Levene Statistic	p-value
Digits	DT	acc	205	**5.670e-21**	14.29	**9.219e-65**
	MLP-adam	acc	256.7	**6.239e-30**	7.342	**1.219e-33**
	MLP-SGD	acc	210	**8.188e-22**	6.53	**2.167e-29**
	RF	acc	184.3	**1.458e-17**	15.61	**9.379e-70**
	SVM	acc	91.72	**2.093e-04**	2.187	**1.790e-05**
	ada	acc	99.34	**2.832e-05**	2.305	**4.601e-06**
	kNN	acc	35	0.9343	0.7042	0.9349
	lasso	acc	22.97	0.9994	0.4292	0.9998
	linear	acc	17.3	1	0.2963	1
	DT	nll	249.6	**1.140e-28**	15.71	**3.892e-70**
	MLP-adam	nll	339.8	**3.816e-45**	6.882	**3.012e-31**
	MLP-SGD	nll	244.8	**7.740e-28**	6.104	**4.129e-27**
	RF	nll	144	**2.791e-11**	7.435	**4.059e-34**
	SVM	nll	4.373	1	0.06091	1
	ada	nll	135.1	**5.444e-10**	3.294	**2.061e-11**
	kNN	nll	108.2	**2.326e-06**	3.059	**4.211e-10**
	lasso	nll	88.4	**4.799e-04**	2.116	**3.995e-05**
	linear	nll	103	**1.024e-05**	3.328	**1.335e-11**

Table 9: Heteroscedasticity tests on tasks involving `iris` data set.

Data set	Model	Metric	Fligner Statistic	p-value	Levene Statistic	p-value
Iris	DT	acc	207.1	**2.440e-21**	6.523	**2.355e-29**
	MLP-adam	acc	83.81	**1.436e-03**	1.838	**7.989e-04**
	MLP-SGD	acc	68.52	**0.03413**	1.409	**0.04082**
	RF	acc	155.5	**5.311e-13**	6.138	**2.726e-27**
	SVM	acc	198.4	**6.990e-20**	3.345	**1.065e-11**
	ada	acc	155.7	**4.788e-13**	5.018	**3.858e-21**
	kNN	acc	55.68	0.2378	1.124	0.2701
	lasso	acc	19.72	0.9999	0.4045	0.9999
	linear	acc	106.4	**3.965e-06**	2.959	**1.502e-09**
	DT	nll	322.2	**7.375e-42**	6.118	**3.506e-27**
	MLP-adam	nll	106.3	**4.070e-06**	3.123	**1.869e-10**
	MLP-SGD	nll	155.6	**4.966e-13**	6.386	**1.264e-28**
	RF	nll	321.3	**1.066e-41**	8.339	**1.136e-38**
	SVM	nll	188.4	**3.217e-18**	4.736	**1.470e-19**
	ada	nll	74.04	**0.01194**	1.414	**0.03938**
	kNN	nll	212.6	**2.863e-22**	8.838	**4.118e-41**
	lasso	nll	45.45	0.6177	0.5045	0.998
	linear	nll	36.64	0.9037	0.733	0.9101

Table 10: Heteroscedasticity tests on tasks involving the `wine` data set.

Data set	Model	Metric	Fligner Statistic	p-value	Levene Statistic	p-value
Wine	DT	acc	127.3	**6.912e-09**	3.553	**7.195e-13**
	MLP-adam	acc	85.37	**9.945e-04**	1.874	**5.544e-04**
	MLP-SGD	acc	109	**1.845e-06**	2.48	**5.701e-07**
	RF	acc	128.5	**4.717e-09**	5.069	**2.014e-21**
	SVM	acc	28.73	0.9908	0.5136	0.9975
	ada	acc	156.6	**3.527e-13**	3.968	**3.215e-15**
	kNN	acc	37.67	0.8807	0.6869	0.9473
	lasso	acc	29.8	0.9862	0.5981	0.9859
	linear	acc	21.28	0.9998	0.3839	1
	DT	nll	349.2	**6.614e-47**	10.46	**1.115e-48**
	MLP-adam	nll	57.19	0.1971	1.21	0.1646
	MLP-SGD	nll	110.1	**1.362e-06**	2.597	**1.380e-07**
	RF	nll	258	**3.660e-30**	6.468	**4.597e-29**
	SVM	nll	57.18	0.1975	1.006	0.4663
	ada	nll	152.8	**1.323e-12**	3.072	**3.555e-10**
	kNN	nll	178.2	**1.410e-16**	5.446	**1.635e-23**
	lasso	nll	83.94	**1.394e-03**	1.782	**1.416e-03**
	linear	nll	185.8	**8.404e-18**	5.01	**4.312e-21**

Table 11: Number of tasks for which each optimiser performed best.

HEBO	TuRBO	PySOT	Skopt	Nevergrad (1+1)	BOHB-BB	Opentuner	Hyperopt	TuRBO+
71 (65.7%)	14 (13.0%)	7 (6.5 %)	5 (4.6 %)	4 (3.7 %)	3 (2.8%)	2 (1.9%)	1 (0.9%)	1 (0.9%)

We now present sequentially the full results for each of the 6 datasets: Boston, Breast cancer, Diabetes, Digits, Iris and Wine. For each optimiser we give the mean and variance of the performance metric across all 18 tasks (2 metrics x 9 models) for a given data set.

Table 12: Boston with MAE loss for tuning DT model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Boston	DT	MAE	106.658094	3.514569
PySOT	Boston	DT	MAE	104.811444	4.220520
Skopt	Boston	DT	MAE	104.303237	4.311549
TuRBO	Boston	DT	MAE	103.863640	6.184650
Nevergrad (1+1)	Boston	DT	MAE	102.938925	17.667195
Hyperopt	Boston	DT	MAE	99.231898	5.681478
TuRBO+	Boston	DT	MAE	98.464758	35.071496
Random-search	Boston	DT	MAE	95.273667	21.448475
BOHB-BB	Boston	DT	MAE	93.360381	37.490514
Opentuner	Boston	DT	MAE	86.421660	313.239599

Table 13: Boston with MAE loss for tuning MLP-adam model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is TuRBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
TuRBO	Boston	MLP-adam	MAE	104.423509	17.216156
PySOT	Boston	MLP-adam	MAE	103.369026	15.037764
HEBO	Boston	MLP-adam	MAE	101.377994	31.981786
TuRBO+	Boston	MLP-adam	MAE	98.759781	46.065189
Nevergrad (1+1)	Boston	MLP-adam	MAE	96.279956	45.560329
Hyperopt	Boston	MLP-adam	MAE	95.255250	28.728828
Random-search	Boston	MLP-adam	MAE	92.866456	23.342156
BOHB-BB	Boston	MLP-adam	MAE	92.001680	17.523725
Opentuner	Boston	MLP-adam	MAE	89.595926	47.161525
Skopt	Boston	MLP-adam	MAE	86.257482	24.245633

Table 14: Boston with MAE loss for tuning MLP-SGD model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is PySOT.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
PySOT	Boston	MLP-SGD	MAE	96.325329	34.592455
HEBO	Boston	MLP-SGD	MAE	95.885011	4.520840
TuRBO	Boston	MLP-SGD	MAE	94.936543	21.232934
Hyperopt	Boston	MLP-SGD	MAE	93.605852	4.184636
Skopt	Boston	MLP-SGD	MAE	91.084570	8.855305
Random-search	Boston	MLP-SGD	MAE	89.854633	9.131124
BOHB-BB	Boston	MLP-SGD	MAE	89.622364	20.485508
TuRBO+	Boston	MLP-SGD	MAE	89.288302	18.617165
Nevergrad (1+1)	Boston	MLP-SGD	MAE	87.149541	115.487770
Opentuner	Boston	MLP-SGD	MAE	85.130342	26.218217

Table 15: Boston with MAE loss for tuning RF model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Boston	RF	MAE	101.726637	0.282878
TuRBO	Boston	RF	MAE	100.095822	2.339170
PySOT	Boston	RF	MAE	99.773979	1.147489
Skopt	Boston	RF	MAE	99.352605	0.805112
Hyperopt	Boston	RF	MAE	98.142703	1.649156
Nevergrad (1+1)	Boston	RF	MAE	96.790676	20.349353
TuRBO+	Boston	RF	MAE	95.817383	7.005061
Opentuner	Boston	RF	MAE	94.012783	32.962245
BOHB-BB	Boston	RF	MAE	93.869871	6.026597
Random-search	Boston	RF	MAE	93.223774	8.626423

Table 16: Boston with MAE loss for tuning SVM model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Boston	SVM	MAE	102.469764	0.015274
Nevergrad (1+1)	Boston	SVM	MAE	102.236442	0.135306
TuRBO	Boston	SVM	MAE	101.826793	0.169932
PySOT	Boston	SVM	MAE	101.262466	0.082329
Skopt	Boston	SVM	MAE	100.914194	0.281085
Hyperopt	Boston	SVM	MAE	98.943800	9.092074
Opentuner	Boston	SVM	MAE	97.906378	12.428820
TuRBO+	Boston	SVM	MAE	90.935974	182.645481
Random-search	Boston	SVM	MAE	88.908494	80.396927
BOHB-BB	Boston	SVM	MAE	86.714806	46.398841

Table 17: Boston with MAE loss for tuning Ada model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Boston	Ada	MAE	103.445454	19.376575
Opentuner	Boston	Ada	MAE	101.293346	54.800061
TuRBO	Boston	Ada	MAE	99.774003	84.126539
Nevergrad (1+1)	Boston	Ada	MAE	97.261376	394.018233
Skopt	Boston	Ada	MAE	97.141270	48.739222
TuRBO+	Boston	Ada	MAE	97.035052	178.220595
Hyperopt	Boston	Ada	MAE	93.394811	59.032286
PySOT	Boston	Ada	MAE	91.618187	25.548574
Random-search	Boston	Ada	MAE	85.889844	39.688171
BOHB-BB	Boston	Ada	MAE	84.104672	99.526333

Table 18: Boston with MAE loss for tuning Knn model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Boston	Knn	MAE	100.000000	0.000000
Opentuner	Boston	Knn	MAE	100.000000	0.000000
Skopt	Boston	Knn	MAE	100.000000	0.000000
TuRBO	Boston	Knn	MAE	100.000000	0.000000
TuRBO+	Boston	Knn	MAE	99.689888	1.923386
PySOT	Boston	Knn	MAE	99.069665	5.162772
Hyperopt	Boston	Knn	MAE	98.023553	9.829689
Nevergrad (1+1)	Boston	Knn	MAE	97.379434	28.707287
BOHB-BB	Boston	Knn	MAE	97.134442	18.411664
Random-search	Boston	Knn	MAE	96.297696	45.566838

Table 19: Boston with MAE loss for tuning Lasso model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Boston	Lasso	MAE	100.031658	0.000141
PySOT	Boston	Lasso	MAE	100.004491	0.001119
Nevergrad (1+1)	Boston	Lasso	MAE	99.989103	0.005199
Skopt	Boston	Lasso	MAE	99.979143	0.001668
TuRBO	Boston	Lasso	MAE	99.977645	0.002141
Hyperopt	Boston	Lasso	MAE	99.967041	0.002263
BOHB-BB	Boston	Lasso	MAE	99.926445	0.006747
TuRBO+	Boston	Lasso	MAE	99.921337	0.031836
Random-search	Boston	Lasso	MAE	99.917670	0.002902
Opentuner	Boston	Lasso	MAE	99.306403	0.956136

Table 20: Boston with MAE loss for tuning Linear model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Boston	Linear	MAE	99.964657	0.002542
Hyperopt	Boston	Linear	MAE	99.922142	0.006819
TuRBO	Boston	Linear	MAE	99.695751	1.213154
PySOT	Boston	Linear	MAE	99.394823	0.799652
BOHB-BB	Boston	Linear	MAE	98.547907	5.197760
Skopt	Boston	Linear	MAE	98.327627	6.150494
Random-search	Boston	Linear	MAE	97.843101	5.252852
TuRBO+	Boston	Linear	MAE	95.874716	153.782738
Nevergrad (1+1)	Boston	Linear	MAE	80.996813	1523.062377
Opentuner	Boston	Linear	MAE	45.221227	2080.382131

Table 21: Boston with MSE loss for tuning DT model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is Skopt.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
Skopt	Boston	DT	MSE	105.587343	12.872983
PySOT	Boston	DT	MSE	105.072075	4.900371
HEBO	Boston	DT	MSE	104.838117	6.188087
TuRBO	Boston	DT	MSE	104.030707	10.799242
Hyperopt	Boston	DT	MSE	102.229770	11.149492
TuRBO+	Boston	DT	MSE	100.450728	12.162198
Nevergrad (1+1)	Boston	DT	MSE	97.629442	407.356319
Random-search	Boston	DT	MSE	95.359894	28.675740
BOHB-BB	Boston	DT	MSE	94.520125	20.675992
Opentuner	Boston	DT	MSE	87.826986	205.761429

Table 22: Boston with MSE loss for tuning MLP-adam model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Boston	MLP-adam	MSE	101.227304	15.854379
TuRBO	Boston	MLP-adam	MSE	100.906498	4.709893
PySOT	Boston	MLP-adam	MSE	98.878397	7.952874
TuRBO+	Boston	MLP-adam	MSE	98.090398	17.299141
Nevergrad (1+1)	Boston	MLP-adam	MSE	97.631386	45.709714
Hyperopt	Boston	MLP-adam	MSE	95.833644	15.829315
BOHB-BB	Boston	MLP-adam	MSE	95.267050	21.466780
Random-search	Boston	MLP-adam	MSE	93.691552	18.033615
Opentuner	Boston	MLP-adam	MSE	92.489905	52.692896
Skopt	Boston	MLP-adam	MSE	87.610428	15.511744

Table 23: Boston with MSE loss for tuning MLP-SGD model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is TuRBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
TuRBO	Boston	MLP-SGD	MSE	104.817056	17.856834
HEBO	Boston	MLP-SGD	MSE	103.285788	4.467401
Hyperopt	Boston	MLP-SGD	MSE	102.049189	6.848005
PySOT	Boston	MLP-SGD	MSE	99.992157	17.117778
TuRBO+	Boston	MLP-SGD	MSE	98.973335	45.162334
Skopt	Boston	MLP-SGD	MSE	98.293397	6.746134
BOHB-BB	Boston	MLP-SGD	MSE	94.731025	17.820451
Random-search	Boston	MLP-SGD	MSE	94.524667	28.931228
Nevergrad (1+1)	Boston	MLP-SGD	MSE	94.267722	207.308935
Opentuner	Boston	MLP-SGD	MSE	88.380591	58.962581

Table 24: Boston with MSE loss for tuning RF model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Boston	RF	MSE	103.461058	0.440697
PySOT	Boston	RF	MSE	101.853905	0.492860
TuRBO	Boston	RF	MSE	101.839078	0.800142
Skopt	Boston	RF	MSE	101.472976	0.606579
Nevergrad (1+1)	Boston	RF	MSE	100.814244	2.514980
Hyperopt	Boston	RF	MSE	100.547643	0.964402
TuRBO+	Boston	RF	MSE	98.915015	3.362617
Random-search	Boston	RF	MSE	98.167459	4.278077
Opentuner	Boston	RF	MSE	98.049981	11.073665
BOHB-BB	Boston	RF	MSE	97.839608	6.645583

Table 25: Boston with MSE loss for tuning SVM model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Boston	SVM	MSE	104.852372	0.005667
Nevergrad (1+1)	Boston	SVM	MSE	104.578053	0.288069
TuRBO	Boston	SVM	MSE	103.821417	1.334006
PySOT	Boston	SVM	MSE	103.398778	0.101558
Hyperopt	Boston	SVM	MSE	101.271084	3.903429
Skopt	Boston	SVM	MSE	100.753282	76.344135
TuRBO+	Boston	SVM	MSE	97.946333	50.658268
Opentuner	Boston	SVM	MSE	91.009871	467.963686
BOHB-BB	Boston	SVM	MSE	90.350500	127.546283
Random-search	Boston	SVM	MSE	83.204838	135.308788

Table 26: Boston with MSE loss for tuning Ada model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Boston	Ada	MSE	96.942494	18.658049
Opentuner	Boston	Ada	MSE	94.274205	34.779775
TuRBO	Boston	Ada	MSE	94.131115	43.476954
Nevergrad (1+1)	Boston	Ada	MSE	92.333218	117.061847
TuRBO+	Boston	Ada	MSE	91.017786	86.796910
Hyperopt	Boston	Ada	MSE	89.721591	41.637778
PySOT	Boston	Ada	MSE	88.996278	29.924305
Skopt	Boston	Ada	MSE	87.835681	55.855654
BOHB-BB	Boston	Ada	MSE	82.947338	67.514235
Random-search	Boston	Ada	MSE	80.119051	17.165168

Table 27: Boston with MSE loss for tuning Knn model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Boston	Knn	MSE	100.000000	0.000000
Opentuner	Boston	Knn	MSE	100.000000	0.000000
PySOT	Boston	Knn	MSE	100.000000	0.000000
TuRBO	Boston	Knn	MSE	100.000000	0.000000
TuRBO+	Boston	Knn	MSE	99.704490	1.253051
Hyperopt	Boston	Knn	MSE	99.319467	6.373596
Random-search	Boston	Knn	MSE	99.068697	7.272026
Skopt	Boston	Knn	MSE	99.014267	19.433403
BOHB-BB	Boston	Knn	MSE	96.129465	62.658638
Nevergrad (1+1)	Boston	Knn	MSE	88.882774	227.331571

Table 28: Boston with MSE loss for tuning Lasso model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Boston	Lasso	MSE	100.037610	0.000010
PySOT	Boston	Lasso	MSE	100.008571	0.000269
Hyperopt	Boston	Lasso	MSE	100.007350	0.000123
TuRBO	Boston	Lasso	MSE	99.982366	0.021257
Skopt	Boston	Lasso	MSE	99.806671	0.433679
BOHB-BB	Boston	Lasso	MSE	99.679375	0.206443
Nevergrad (1+1)	Boston	Lasso	MSE	99.372815	4.020588
Random-search	Boston	Lasso	MSE	99.325799	0.769118
TuRBO+	Boston	Lasso	MSE	98.971017	2.235443
Opentuner	Boston	Lasso	MSE	97.388970	6.008643

Table 29: Boston with MSE loss for tuning Linear model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Boston	Linear	MSE	100.000001	3.207627e-10
Hyperopt	Boston	Linear	MSE	99.999348	1.340314e-06
PySOT	Boston	Linear	MSE	99.994196	1.378338e-04
Skopt	Boston	Linear	MSE	99.986643	6.872575e-04
Nevergrad (1+1)	Boston	Linear	MSE	99.976999	5.017296e-03
TuRBO	Boston	Linear	MSE	99.889263	2.451020e-01
BOHB-BB	Boston	Linear	MSE	99.837413	2.455636e-01
Random-search	Boston	Linear	MSE	99.802681	2.731228e-01
TuRBO+	Boston	Linear	MSE	99.332841	1.085780e+00
Opentuner	Boston	Linear	MSE	98.545654	1.344348e+00

Table 30: Breast cancer dataset with ACC loss for tuning DT model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Breast cancer dataset	DT	ACC	106.694915	52.457703
PySOT	Breast cancer dataset	DT	ACC	103.220339	32.930646
TuRBO	Breast cancer dataset	DT	ACC	99.830508	32.930646
Nevergrad (1+1)	Breast cancer dataset	DT	ACC	99.661017	47.052420
Skopt	Breast cancer dataset	DT	ACC	99.322034	27.033974
Opentuner	Breast cancer dataset	DT	ACC	97.457627	19.201984
Hyperopt	Breast cancer dataset	DT	ACC	97.288136	17.357384
TuRBO+	Breast cancer dataset	DT	ACC	96.610169	26.913017
Random-search	Breast cancer dataset	DT	ACC	93.813559	29.415322
BOHB-BB	Breast cancer dataset	DT	ACC	93.305085	22.823145

Table 31: Breast cancer dataset with ACC loss for tuning MLP-adam model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is Hyperopt.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
Hyperopt	Breast cancer dataset	MLP-adam	ACC	99.078947	14.779851
HEBO	Breast cancer dataset	MLP-adam	ACC	98.684211	20.046654
Nevergrad (1+1)	Breast cancer dataset	MLP-adam	ACC	96.973684	53.415221
PySOT	Breast cancer dataset	MLP-adam	ACC	96.578947	11.736405
TuRBO	Breast cancer dataset	MLP-adam	ACC	96.447368	17.695728
TuRBO+	Breast cancer dataset	MLP-adam	ACC	96.447368	18.424697
Skopt	Breast cancer dataset	MLP-adam	ACC	95.921053	20.757399
Opentuner	Breast cancer dataset	MLP-adam	ACC	95.000000	21.067211
Random-search	Breast cancer dataset	MLP-adam	ACC	94.078947	13.759294
BOHB-BB	Breast cancer dataset	MLP-adam	ACC	93.684211	11.955095

Table 32: Breast cancer dataset with ACC loss for tuning MLP-SGD model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Breast cancer dataset	MLP-SGD	ACC	100.847458	1.209574
TuRBO	Breast cancer dataset	MLP-SGD	ACC	99.491525	1.391010
Skopt	Breast cancer dataset	MLP-SGD	ACC	99.237288	0.824022
PySOT	Breast cancer dataset	MLP-SGD	ACC	99.067797	1.504407
Hyperopt	Breast cancer dataset	MLP-SGD	ACC	98.771186	1.396680
Nevergrad (1+1)	Breast cancer dataset	MLP-SGD	ACC	98.771186	4.647409
TuRBO+	Breast cancer dataset	MLP-SGD	ACC	98.559322	2.502306
Random-search	Breast cancer dataset	MLP-SGD	ACC	96.864407	1.973117
BOHB-BB	Breast cancer dataset	MLP-SGD	ACC	96.355932	3.787478
Opentuner	Breast cancer dataset	MLP-SGD	ACC	84.703390	190.241386

Table 33: Breast cancer dataset with ACC loss for tuning RF model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is Skopt.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
Skopt	Breast cancer dataset	RF	ACC	98.846154	5.985365
HEBO	Breast cancer dataset	RF	ACC	98.356643	8.250307
PySOT	Breast cancer dataset	RF	ACC	97.377622	5.964775
Hyperopt	Breast cancer dataset	RF	ACC	96.888112	3.988099
TuRBO	Breast cancer dataset	RF	ACC	96.853147	8.416317
Nevergrad (1+1)	Breast cancer dataset	RF	ACC	96.083916	13.249908
TuRBO+	Breast cancer dataset	RF	ACC	95.419580	6.047137
BOHB-BB	Breast cancer dataset	RF	ACC	94.335664	2.980457
Opentuner	Breast cancer dataset	RF	ACC	93.636364	3.289313
Random-search	Breast cancer dataset	RF	ACC	93.321678	2.495296

Table 34: Breast cancer dataset with ACC loss for tuning SVM model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is TuRBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
TuRBO	Breast cancer dataset	SVM	ACC	89.285714	40.279270
Hyperopt	Breast cancer dataset	SVM	ACC	86.428571	53.168636
TuRBO+	Breast cancer dataset	SVM	ACC	86.428571	96.133190
PySOT	Breast cancer dataset	SVM	ACC	85.714286	42.964554
HEBO	Breast cancer dataset	SVM	ACC	84.285714	40.816327
BOHB-BB	Breast cancer dataset	SVM	ACC	80.714286	113.319012
Opentuner	Breast cancer dataset	SVM	ACC	78.571429	182.599356
Nevergrad (1+1)	Breast cancer dataset	SVM	ACC	77.142857	481.203007
Random-search	Breast cancer dataset	SVM	ACC	76.428571	48.872180
Skopt	Breast cancer dataset	SVM	ACC	76.428571	349.624060

Table 35: Breast cancer dataset with ACC loss for tuning Ada model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Breast cancer dataset	Ada	ACC	99.06250	4.214638
Opentuner	Breast cancer dataset	Ada	ACC	98.59375	6.656044
TuRBO	Breast cancer dataset	Ada	ACC	98.43750	3.597862
TuRBO+	Breast cancer dataset	Ada	ACC	97.96875	8.506373
Nevergrad (1+1)	Breast cancer dataset	Ada	ACC	97.65625	12.207031
PySOT	Breast cancer dataset	Ada	ACC	97.03125	5.628084
Hyperopt	Breast cancer dataset	Ada	ACC	96.87500	8.223684
Random-search	Breast cancer dataset	Ada	ACC	95.31250	6.681743
BOHB-BB	Breast cancer dataset	Ada	ACC	94.53125	3.983347
Skopt	Breast cancer dataset	Ada	ACC	93.12500	9.868421

Table 36: Breast cancer dataset with ACC loss for tuning Knn model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Breast cancer dataset	Knn	ACC	100.000	0.000000
Opentuner	Breast cancer dataset	Knn	ACC	100.000	0.000000
Hyperopt	Breast cancer dataset	Knn	ACC	97.500	26.315789
Nevergrad (1+1)	Breast cancer dataset	Knn	ACC	95.625	53.865132
Random-search	Breast cancer dataset	Knn	ACC	95.625	53.865132
Skopt	Breast cancer dataset	Knn	ACC	95.000	55.921053
BOHB-BB	Breast cancer dataset	Knn	ACC	95.000	39.473684
TuRBO+	Breast cancer dataset	Knn	ACC	92.500	55.921053
TuRBO	Breast cancer dataset	Knn	ACC	91.875	37.417763
PySOT	Breast cancer dataset	Knn	ACC	91.250	34.539474

Table 37: Breast cancer dataset with ACC loss for tuning Lasso model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is PySOT.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
PySOT	Breast cancer dataset	Lasso	ACC	95.000000	32.163743
TuRBO	Breast cancer dataset	Lasso	ACC	93.888889	32.163743
Opentuner	Breast cancer dataset	Lasso	ACC	93.888889	32.163743
HEBO	Breast cancer dataset	Lasso	ACC	93.333333	31.189084
TuRBO+	Breast cancer dataset	Lasso	ACC	92.222222	27.290448
Hyperopt	Breast cancer dataset	Lasso	ACC	90.555556	16.569201
Random-search	Breast cancer dataset	Lasso	ACC	90.555556	16.569201
BOHB-BB	Breast cancer dataset	Lasso	ACC	90.000000	11.695906
Nevergrad (1+1)	Breast cancer dataset	Lasso	ACC	88.333333	58.154646
Skopt	Breast cancer dataset	Lasso	ACC	87.777778	180.636777

Table 38: Breast cancer dataset with ACC loss for tuning Linear model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Breast cancer dataset	Linear	ACC	107.0	95.789474
Nevergrad (1+1)	Breast cancer dataset	Linear	ACC	103.0	390.526316
TuRBO	Breast cancer dataset	Linear	ACC	99.0	188.421053
PySOT	Breast cancer dataset	Linear	ACC	98.0	164.210526
TuRBO+	Breast cancer dataset	Linear	ACC	98.0	164.210526
Opentuner	Breast cancer dataset	Linear	ACC	98.0	290.526316
Hyperopt	Breast cancer dataset	Linear	ACC	96.0	109.473684
Skopt	Breast cancer dataset	Linear	ACC	95.0	205.263158
Random-search	Breast cancer dataset	Linear	ACC	89.0	146.315789
BOHB-BB	Breast cancer dataset	Linear	ACC	85.0	78.947368

Table 39: Breast cancer dataset with NLL loss for tuning DT model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Breast cancer dataset	DT	NLL	111.047413	51.627976
TuRBO+	Breast cancer dataset	DT	NLL	98.912425	121.134669
PySOT	Breast cancer dataset	DT	NLL	98.696805	115.036349
Hyperopt	Breast cancer dataset	DT	NLL	98.201008	75.868808
Skopt	Breast cancer dataset	DT	NLL	97.647135	165.617171
TuRBO	Breast cancer dataset	DT	NLL	95.406797	230.498379
Opentuner	Breast cancer dataset	DT	NLL	91.076386	92.789945
Nevergrad (1+1)	Breast cancer dataset	DT	NLL	90.579282	92.638289
BOHB-BB	Breast cancer dataset	DT	NLL	90.418909	43.016544
Random-search	Breast cancer dataset	DT	NLL	87.752435	64.953742

Table 40: Breast cancer dataset with NLL loss for tuning MLP-adam model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Breast cancer dataset	MLP-adam	NLL	100.303210	0.252546
TuRBO	Breast cancer dataset	MLP-adam	NLL	100.260440	0.386150
PySOT	Breast cancer dataset	MLP-adam	NLL	99.965212	0.410841
Hyperopt	Breast cancer dataset	MLP-adam	NLL	99.778153	0.295595
Opentuner	Breast cancer dataset	MLP-adam	NLL	99.614006	0.559840
Skopt	Breast cancer dataset	MLP-adam	NLL	99.422393	0.444736
TuRBO+	Breast cancer dataset	MLP-adam	NLL	99.298544	1.362840
Nevergrad (1+1)	Breast cancer dataset	MLP-adam	NLL	98.865542	5.064746
Random-search	Breast cancer dataset	MLP-adam	NLL	98.538089	1.227034
BOHB-BB	Breast cancer dataset	MLP-adam	NLL	98.396155	0.577588

Table 41: Breast cancer dataset with NLL loss for tuning MLP-SGD model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Breast cancer dataset	MLP-SGD	NLL	100.721473	0.061946
TuRBO	Breast cancer dataset	MLP-SGD	NLL	100.162853	1.622656
PySOT	Breast cancer dataset	MLP-SGD	NLL	100.091236	0.153482
Hyperopt	Breast cancer dataset	MLP-SGD	NLL	100.049807	0.198361
TuRBO+	Breast cancer dataset	MLP-SGD	NLL	99.717751	0.569165
Skopt	Breast cancer dataset	MLP-SGD	NLL	99.682763	0.194010
Random-search	Breast cancer dataset	MLP-SGD	NLL	99.084890	0.573484
BOHB-BB	Breast cancer dataset	MLP-SGD	NLL	99.061649	0.934630
Nevergrad (1+1)	Breast cancer dataset	MLP-SGD	NLL	98.762368	4.114520
Opentuner	Breast cancer dataset	MLP-SGD	NLL	97.518858	6.031878

Table 42: Breast cancer dataset with NLL loss for tuning RF model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Breast cancer dataset	RF	NLL	104.284930	1.565493
PySOT	Breast cancer dataset	RF	NLL	102.796116	4.187314
Skopt	Breast cancer dataset	RF	NLL	101.998789	4.898058
Hyperopt	Breast cancer dataset	RF	NLL	98.198100	10.076521
TuRBO	Breast cancer dataset	RF	NLL	97.942753	25.132499
Nevergrad (1+1)	Breast cancer dataset	RF	NLL	96.820234	31.448689
TuRBO+	Breast cancer dataset	RF	NLL	96.029192	12.874949
Opentuner	Breast cancer dataset	RF	NLL	92.790267	10.042108
BOHB-BB	Breast cancer dataset	RF	NLL	92.660352	5.982402
Random-search	Breast cancer dataset	RF	NLL	92.434443	6.196296

Table 43: Breast cancer dataset with NLL loss for tuning SVM model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Breast cancer dataset	SVM	NLL	100.999289	0.870906
TuRBO	Breast cancer dataset	SVM	NLL	100.331139	0.502305
PySOT	Breast cancer dataset	SVM	NLL	100.038823	0.614796
Hyperopt	Breast cancer dataset	SVM	NLL	99.238507	1.374858
Opentuner	Breast cancer dataset	SVM	NLL	99.130932	3.472981
TuRBO+	Breast cancer dataset	SVM	NLL	98.787566	7.629457
Nevergrad (1+1)	Breast cancer dataset	SVM	NLL	98.258990	41.913612
Skopt	Breast cancer dataset	SVM	NLL	98.153794	16.959059
BOHB-BB	Breast cancer dataset	SVM	NLL	97.569333	1.733710
Random-search	Breast cancer dataset	SVM	NLL	97.191463	1.506776

Table 44: Breast cancer dataset with NLL loss for tuning Ada model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Breast cancer dataset	Ada	NLL	103.298980	1.970935
TuRBO	Breast cancer dataset	Ada	NLL	100.445760	7.172282
PySOT	Breast cancer dataset	Ada	NLL	100.402760	4.365384
Skopt	Breast cancer dataset	Ada	NLL	98.923004	5.054022
Nevergrad (1+1)	Breast cancer dataset	Ada	NLL	98.874005	6.293192
Hyperopt	Breast cancer dataset	Ada	NLL	98.645011	1.948260
TuRBO+	Breast cancer dataset	Ada	NLL	98.474514	3.187387
Random-search	Breast cancer dataset	Ada	NLL	97.922922	2.325453
BOHB-BB	Breast cancer dataset	Ada	NLL	97.671007	2.051746
Opentuner	Breast cancer dataset	Ada	NLL	96.538817	164.478434

Table 45: Breast cancer dataset with NLL loss for tuning Knn model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Breast cancer dataset	Knn	NLL	100.000000	0.000000
Opentuner	Breast cancer dataset	Knn	NLL	100.000000	0.000000
TuRBO	Breast cancer dataset	Knn	NLL	99.835048	0.257772
Hyperopt	Breast cancer dataset	Knn	NLL	99.815764	0.321563
Random-search	Breast cancer dataset	Knn	NLL	99.248073	0.729147
BOHB-BB	Breast cancer dataset	Knn	NLL	99.121241	2.900580
PySOT	Breast cancer dataset	Knn	NLL	98.175743	5.063927
Nevergrad (1+1)	Breast cancer dataset	Knn	NLL	97.334954	8.601850
TuRBO+	Breast cancer dataset	Knn	NLL	95.752544	121.788536
Skopt	Breast cancer dataset	Knn	NLL	85.667024	457.645212

Table 46: Breast cancer dataset with NLL loss for tuning Lasso model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is PySOT.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
PySOT	Breast cancer dataset	Lasso	NLL	92.444124	41.420464
Opentuner	Breast cancer dataset	Lasso	NLL	92.364922	52.804059
Hyperopt	Breast cancer dataset	Lasso	NLL	90.483697	40.153553
TuRBO+	Breast cancer dataset	Lasso	NLL	89.669648	32.541741
TuRBO	Breast cancer dataset	Lasso	NLL	89.275892	40.406114
Random-search	Breast cancer dataset	Lasso	NLL	89.267387	86.949148
HEBO	Breast cancer dataset	Lasso	NLL	87.648453	9.938537
Nevergrad (1+1)	Breast cancer dataset	Lasso	NLL	84.512735	81.507309
BOHB-BB	Breast cancer dataset	Lasso	NLL	81.089709	45.062520
Skopt	Breast cancer dataset	Lasso	NLL	80.270494	104.357765

Table 47: Breast cancer dataset with NLL loss for tuning Linear model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Breast cancer dataset	Linear	NLL	97.097724	4.317349
TuRBO	Breast cancer dataset	Linear	NLL	96.073657	20.501975
PySOT	Breast cancer dataset	Linear	NLL	95.497805	3.788872
Opentuner	Breast cancer dataset	Linear	NLL	94.970576	5.694999
Skopt	Breast cancer dataset	Linear	NLL	93.683985	8.175137
TuRBO+	Breast cancer dataset	Linear	NLL	93.460437	18.265925
Hyperopt	Breast cancer dataset	Linear	NLL	92.065782	7.999446
Nevergrad (1+1)	Breast cancer dataset	Linear	NLL	91.268968	86.801548
Random-search	Breast cancer dataset	Linear	NLL	90.477323	8.865931
BOHB-BB	Breast cancer dataset	Linear	NLL	89.401005	15.561542

Table 48: Diabetes with MAE loss for tuning DT model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Diabetes	DT	MAE	96.638714	19.424434
Skopt	Diabetes	DT	MAE	96.283909	12.675138
TuRBO	Diabetes	DT	MAE	96.161335	22.916642
PySOT	Diabetes	DT	MAE	95.330459	14.421246
Hyperopt	Diabetes	DT	MAE	92.694296	30.442937
TuRBO+	Diabetes	DT	MAE	92.417202	24.654829
Random-search	Diabetes	DT	MAE	89.414992	33.396944
BOHB-BB	Diabetes	DT	MAE	88.578413	21.742304
Nevergrad (1+1)	Diabetes	DT	MAE	87.192577	491.678538
Opentuner	Diabetes	DT	MAE	84.459098	199.389559

Table 49: Diabetes with MAE loss for tuning MLP-adam model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is PySOT.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
PySOT	Diabetes	MLP-adam	MAE	100.011743	0.039479
TuRBO	Diabetes	MLP-adam	MAE	100.003598	0.132742
HEBO	Diabetes	MLP-adam	MAE	99.905117	0.071792
Nevergrad (1+1)	Diabetes	MLP-adam	MAE	99.783143	0.276722
Hyperopt	Diabetes	MLP-adam	MAE	99.747348	0.024272
Skopt	Diabetes	MLP-adam	MAE	99.691162	0.057345
Opentuner	Diabetes	MLP-adam	MAE	99.564832	0.075922
TuRBO+	Diabetes	MLP-adam	MAE	99.518025	0.125512
BOHB-BB	Diabetes	MLP-adam	MAE	99.343161	0.055340
Random-search	Diabetes	MLP-adam	MAE	99.288591	0.104943

Table 50: Diabetes with MAE loss for tuning MLP-SGD model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Diabetes	MLP-SGD	MAE	99.305259	0.571598
TuRBO	Diabetes	MLP-SGD	MAE	99.173074	2.151207
PySOT	Diabetes	MLP-SGD	MAE	98.493992	3.290314
TuRBO+	Diabetes	MLP-SGD	MAE	98.017532	2.468709
Hyperopt	Diabetes	MLP-SGD	MAE	97.991732	1.875066
Skopt	Diabetes	MLP-SGD	MAE	97.323238	1.402552
BOHB-BB	Diabetes	MLP-SGD	MAE	94.355532	9.155887
Random-search	Diabetes	MLP-SGD	MAE	94.196391	42.702734
Opentuner	Diabetes	MLP-SGD	MAE	93.018054	116.857957
Nevergrad (1+1)	Diabetes	MLP-SGD	MAE	87.052894	249.800579

Table 51: Diabetes with MAE loss for tuning RF model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is TuRBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
TuRBO	Diabetes	RF	MAE	100.689329	1.625789
PySOT	Diabetes	RF	MAE	100.621188	2.067673
HEBO	Diabetes	RF	MAE	100.420933	1.415993
Skopt	Diabetes	RF	MAE	99.206462	1.447476
Nevergrad (1+1)	Diabetes	RF	MAE	98.695634	22.941598
Hyperopt	Diabetes	RF	MAE	98.321835	1.287212
TuRBO+	Diabetes	RF	MAE	97.985211	5.657556
BOHB-BB	Diabetes	RF	MAE	96.181909	6.997435
Random-search	Diabetes	RF	MAE	95.677899	5.078577
Opentuner	Diabetes	RF	MAE	93.512496	46.716147

Table 52: Diabetes with MAE loss for tuning SVM model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is Nevergrad.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
Nevergrad (1+1)	Diabetes	SVM	MAE	114.952221	1.997569e-23
HEBO	Diabetes	SVM	MAE	114.952206	4.667718e-10
TuRBO	Diabetes	SVM	MAE	114.828240	9.438118e-03
PySOT	Diabetes	SVM	MAE	114.410275	1.494253e-01
Skopt	Diabetes	SVM	MAE	113.977223	2.466076e-01
Opentuner	Diabetes	SVM	MAE	113.263069	9.959103e+00
TuRBO+	Diabetes	SVM	MAE	106.842965	1.060824e+02
Hyperopt	Diabetes	SVM	MAE	104.021953	8.292988e+01
Random-search	Diabetes	SVM	MAE	76.950614	2.326428e+02
BOHB-BB	Diabetes	SVM	MAE	70.832961	2.115443e+02

Table 53: Diabetes with MAE loss for tuning Ada model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is Opentuner.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
Opentuner	Diabetes	Ada	MAE	82.820466	729.771421
HEBO	Diabetes	Ada	MAE	81.642682	638.761474
TuRBO	Diabetes	Ada	MAE	80.090059	159.113760
PySOT	Diabetes	Ada	MAE	79.899689	271.945194
Hyperopt	Diabetes	Ada	MAE	79.323266	146.085370
BOHB-BB	Diabetes	Ada	MAE	77.689133	209.828285
TuRBO+	Diabetes	Ada	MAE	76.467861	135.190360
Skopt	Diabetes	Ada	MAE	75.430634	172.038737
Nevergrad (1+1)	Diabetes	Ada	MAE	73.932751	141.078137
Random-search	Diabetes	Ada	MAE	73.572795	183.641961

Table 54: Diabetes with MAE loss for tuning Knn model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Diabetes	Knn	MAE	100.000000	0.000000
Opentuner	Diabetes	Knn	MAE	100.000000	0.000000
PySOT	Diabetes	Knn	MAE	99.918437	0.133050
TuRBO	Diabetes	Knn	MAE	99.276294	10.475009
Hyperopt	Diabetes	Knn	MAE	97.929753	28.990634
Nevergrad (1+1)	Diabetes	Knn	MAE	97.515979	58.516868
Random-search	Diabetes	Knn	MAE	95.595510	46.440649
BOHB-BB	Diabetes	Knn	MAE	95.543615	56.425722
Skopt	Diabetes	Knn	MAE	95.429465	66.513444
TuRBO+	Diabetes	Knn	MAE	91.604083	134.441613

Table 55: Diabetes with MAE loss for tuning Lasso model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is Skopt.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
Skopt	Diabetes	Lasso	MAE	100.024636	0.001070
HEBO	Diabetes	Lasso	MAE	100.020048	0.000540
TuRBO	Diabetes	Lasso	MAE	100.011646	0.000588
PySOT	Diabetes	Lasso	MAE	100.006646	0.000432
Hyperopt	Diabetes	Lasso	MAE	100.004698	0.000154
TuRBO+	Diabetes	Lasso	MAE	99.997964	0.000615
BOHB-BB	Diabetes	Lasso	MAE	99.995219	0.000595
Nevergrad (1+1)	Diabetes	Lasso	MAE	99.993918	0.000879
Random-search	Diabetes	Lasso	MAE	99.993704	0.000037
Opentuner	Diabetes	Lasso	MAE	99.813361	0.285529

Table 56: Diabetes with MAE loss for tuning Linear model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Diabetes	Linear	MAE	99.999997	7.405442e-11
TuRBO	Diabetes	Linear	MAE	99.999955	5.421510e-08
PySOT	Diabetes	Linear	MAE	99.999884	8.825730e-08
Nevergrad (1+1)	Diabetes	Linear	MAE	99.999798	1.823753e-07
Hyperopt	Diabetes	Linear	MAE	99.999756	1.834449e-07
Skopt	Diabetes	Linear	MAE	99.999461	3.064474e-07
BOHB-BB	Diabetes	Linear	MAE	99.999386	4.929089e-07
Random-search	Diabetes	Linear	MAE	99.999226	7.463938e-07
TuRBO+	Diabetes	Linear	MAE	99.998033	5.543436e-05
Opentuner	Diabetes	Linear	MAE	99.990706	1.793190e-04

Table 57: Diabetes with MSE loss for tuning DT model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is TuRBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
TuRBO	Diabetes	DT	MSE	99.159752	20.394596
HEBO	Diabetes	DT	MSE	99.105320	27.539740
PySOT	Diabetes	DT	MSE	98.463863	12.943768
TuRBO+	Diabetes	DT	MSE	98.146099	29.207851
Skopt	Diabetes	DT	MSE	98.011055	9.114515
Nevergrad (1+1)	Diabetes	DT	MSE	95.965807	89.742185
Hyperopt	Diabetes	DT	MSE	94.106805	17.465775
BOHB-BB	Diabetes	DT	MSE	91.746558	29.504056
Random-search	Diabetes	DT	MSE	90.512956	24.763756
Opentuner	Diabetes	DT	MSE	85.951624	189.632259

Table 58: Diabetes with MSE loss for tuning MLP-adam model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is TuRBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
TuRBO	Diabetes	MLP-adam	MSE	100.080222	0.011338
PySOT	Diabetes	MLP-adam	MSE	100.021190	0.010076
Nevergrad (1+1)	Diabetes	MLP-adam	MSE	100.017846	0.019891
HEBO	Diabetes	MLP-adam	MSE	99.977968	0.004366
Hyperopt	Diabetes	MLP-adam	MSE	99.937035	0.011698
Opentuner	Diabetes	MLP-adam	MSE	99.886863	0.005519
Skopt	Diabetes	MLP-adam	MSE	99.838359	0.008621
TuRBO+	Diabetes	MLP-adam	MSE	99.825080	0.033512
Random-search	Diabetes	MLP-adam	MSE	99.777438	0.009177
BOHB-BB	Diabetes	MLP-adam	MSE	99.742448	0.011606

Table 59: Diabetes with MSE loss for tuning MLP-SGD model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Diabetes	MLP-SGD	MSE	99.502940	0.179089
TuRBO	Diabetes	MLP-SGD	MSE	99.498069	0.391187
Hyperopt	Diabetes	MLP-SGD	MSE	98.948022	0.324618
TuRBO+	Diabetes	MLP-SGD	MSE	98.704370	1.720076
Skopt	Diabetes	MLP-SGD	MSE	98.452683	1.161588
PySOT	Diabetes	MLP-SGD	MSE	98.357894	3.117919
Random-search	Diabetes	MLP-SGD	MSE	97.569960	3.685257
BOHB-BB	Diabetes	MLP-SGD	MSE	95.630189	18.807287
Opentuner	Diabetes	MLP-SGD	MSE	94.462201	34.471258
Nevergrad (1+1)	Diabetes	MLP-SGD	MSE	92.620769	179.226746

Table 60: Diabetes with MSE loss for tuning RF model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Diabetes	RF	MSE	99.567950	1.137694
PySOT	Diabetes	RF	MSE	99.197939	1.240447
TuRBO	Diabetes	RF	MSE	99.154475	1.995345
Skopt	Diabetes	RF	MSE	98.665889	0.909053
Hyperopt	Diabetes	RF	MSE	98.271427	1.306007
TuRBO+	Diabetes	RF	MSE	97.396962	3.233997
Nevergrad (1+1)	Diabetes	RF	MSE	96.904507	16.420320
Random-search	Diabetes	RF	MSE	96.478596	4.042486
BOHB-BB	Diabetes	RF	MSE	96.269017	3.544861
Opentuner	Diabetes	RF	MSE	95.846844	22.993955

Table 61: Diabetes with MSE loss for tuning SVM model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is Nevergrad.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
Nevergrad (1+1)	Diabetes	SVM	MSE	102.789567	2.184625e-24
HEBO	Diabetes	SVM	MSE	102.789540	1.530581e-09
TuRBO	Diabetes	SVM	MSE	102.673445	8.390893e-03
PySOT	Diabetes	SVM	MSE	102.020190	4.231205e-01
Skopt	Diabetes	SVM	MSE	101.868102	1.554345e-01
Opentuner	Diabetes	SVM	MSE	100.897226	1.367135e+01
Hyperopt	Diabetes	SVM	MSE	95.268658	4.223803e+01
TuRBO+	Diabetes	SVM	MSE	94.606843	4.791899e+01
BOHB-BB	Diabetes	SVM	MSE	74.069393	1.232001e+02
Random-search	Diabetes	SVM	MSE	65.895981	1.918664e+02

Table 62: Diabetes with MSE loss for tuning Ada model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is Nevergrad.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
Nevergrad (1+1)	Diabetes	Ada	MSE	68.161267	269.888332
Skopt	Diabetes	Ada	MSE	67.321751	190.562037
Opentuner	Diabetes	Ada	MSE	65.700020	63.006486
PySOT	Diabetes	Ada	MSE	63.875149	105.453227
Hyperopt	Diabetes	Ada	MSE	63.247637	100.721372
HEBO	Diabetes	Ada	MSE	63.117246	55.910472
Random-search	Diabetes	Ada	MSE	63.112054	51.487703
TuRBO	Diabetes	Ada	MSE	62.719630	59.718734
BOHB-BB	Diabetes	Ada	MSE	62.520126	91.727211
TuRBO+	Diabetes	Ada	MSE	61.558612	66.945920

Table 63: Diabetes with MSE loss for tuning Knn model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Diabetes	Knn	MSE	100.000000	0.000000
PySOT	Diabetes	Knn	MSE	100.000000	0.000000
Opentuner	Diabetes	Knn	MSE	99.019554	19.225491
Nevergrad (1+1)	Diabetes	Knn	MSE	96.078216	64.759548
Hyperopt	Diabetes	Knn	MSE	95.868028	72.584707
BOHB-BB	Diabetes	Knn	MSE	94.460734	99.952329
Skopt	Diabetes	Knn	MSE	93.696947	98.818821
TuRBO	Diabetes	Knn	MSE	91.661607	114.292743
Random-search	Diabetes	Knn	MSE	89.280338	126.210687
TuRBO+	Diabetes	Knn	MSE	88.510080	118.833506

Table 64: Diabetes with MSE loss for tuning Lasso model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is TuRBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
TuRBO	Diabetes	Lasso	MSE	100.001427	0.000065
HEBO	Diabetes	Lasso	MSE	99.993203	0.000066
Skopt	Diabetes	Lasso	MSE	99.988906	0.000109
Hyperopt	Diabetes	Lasso	MSE	99.988425	0.000162
PySOT	Diabetes	Lasso	MSE	99.984161	0.000170
TuRBO+	Diabetes	Lasso	MSE	99.982936	0.000182
BOHB-BB	Diabetes	Lasso	MSE	99.969386	0.000650
Random-search	Diabetes	Lasso	MSE	99.968212	0.000182
Nevergrad (1+1)	Diabetes	Lasso	MSE	99.889782	0.015621
Opentuner	Diabetes	Lasso	MSE	99.549967	2.274095

Table 65: Diabetes with MSE loss for tuning Linear model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Diabetes	Linear	MSE	99.999999	5.900912e-12
TuRBO	Diabetes	Linear	MSE	99.999987	3.453784e-09
Hyperopt	Diabetes	Linear	MSE	99.999900	1.682948e-07
Skopt	Diabetes	Linear	MSE	99.999887	5.424762e-08
PySOT	Diabetes	Linear	MSE	99.999856	1.702504e-07
Random-search	Diabetes	Linear	MSE	99.999833	4.783733e-08
BOHB-BB	Diabetes	Linear	MSE	99.999821	1.116438e-07
TuRBO+	Diabetes	Linear	MSE	99.999698	2.978636e-07
Nevergrad (1+1)	Diabetes	Linear	MSE	99.999633	5.677318e-07
Opentuner	Diabetes	Linear	MSE	99.989516	7.327061e-04

Table 66: Digits with ACC loss for tuning DT model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Digits	DT	ACC	109.992773	0.333759
TuRBO	Digits	DT	ACC	109.229157	2.411464
PySOT	Digits	DT	ACC	106.415805	30.276129
Skopt	Digits	DT	ACC	102.201342	234.180142
Hyperopt	Digits	DT	ACC	92.812052	64.914274
TuRBO+	Digits	DT	ACC	75.463155	434.854745
Opentuner	Digits	DT	ACC	71.263614	642.656253
BOHB-BB	Digits	DT	ACC	68.750324	284.398998
Random-search	Digits	DT	ACC	65.051159	432.111162
Nevergrad (1+1)	Digits	DT	ACC	61.284002	1994.275954

Table 67: Digits with ACC loss for tuning MLP-adam model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is TuRBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
TuRBO	Digits	MLP-adam	ACC	102.902359	5.829126
HEBO	Digits	MLP-adam	ACC	101.875822	12.570464
TuRBO+	Digits	MLP-adam	ACC	100.266068	18.108832
Hyperopt	Digits	MLP-adam	ACC	100.263270	4.310725
PySOT	Digits	MLP-adam	ACC	99.686892	13.193611
Nevergrad (1+1)	Digits	MLP-adam	ACC	96.867646	36.871957
BOHB-BB	Digits	MLP-adam	ACC	95.526339	22.930349
Skopt	Digits	MLP-adam	ACC	95.215922	17.997087
Opentuner	Digits	MLP-adam	ACC	94.365035	11.136649
Random-search	Digits	MLP-adam	ACC	94.005964	6.753444

Table 68: Digits with ACC loss for tuning MLP-SGD model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Digits	MLP-SGD	ACC	101.197199	0.137161
TuRBO	Digits	MLP-SGD	ACC	101.186343	0.329266
PySOT	Digits	MLP-SGD	ACC	100.810497	0.305607
Hyperopt	Digits	MLP-SGD	ACC	100.374369	0.547541
Skopt	Digits	MLP-SGD	ACC	100.221920	0.490938
Opentuner	Digits	MLP-SGD	ACC	100.009681	0.631299
Nevergrad (1+1)	Digits	MLP-SGD	ACC	99.638780	5.425609
TuRBO+	Digits	MLP-SGD	ACC	99.213829	1.526439
Random-search	Digits	MLP-SGD	ACC	98.434458	1.856923
BOHB-BB	Digits	MLP-SGD	ACC	98.135339	0.652457

Table 69: Digits with ACC loss for tuning RF model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Digits	RF	ACC	106.100300	1.232976
PySOT	Digits	RF	ACC	104.238234	40.163502
Skopt	Digits	RF	ACC	104.091452	1.113028
Hyperopt	Digits	RF	ACC	99.371979	7.584547
TuRBO	Digits	RF	ACC	90.437346	1074.875086
TuRBO+	Digits	RF	ACC	86.244306	105.700384
BOHB-BB	Digits	RF	ACC	81.694197	146.883857
Opentuner	Digits	RF	ACC	78.227366	197.260382
Random-search	Digits	RF	ACC	77.402425	236.222327
Nevergrad (1+1)	Digits	RF	ACC	62.720269	2400.500537

Table 70: Digits with ACC loss for tuning SVM model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Digits	SVM	ACC	131.797994	0.000000
Hyperopt	Digits	SVM	ACC	130.140907	54.918774
Opentuner	Digits	SVM	ACC	128.483819	104.056624
TuRBO	Digits	SVM	ACC	126.826732	147.413550
Random-search	Digits	SVM	ACC	125.169645	184.989553
PySOT	Digits	SVM	ACC	118.541295	277.484329
TuRBO+	Digits	SVM	ACC	116.889961	285.935578
BOHB-BB	Digits	SVM	ACC	111.935960	276.844235
Skopt	Digits	SVM	ACC	108.610278	358.177641
Nevergrad (1+1)	Digits	SVM	ACC	93.702245	838.899480

Table 71: Digits with ACC loss for tuning Ada model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Digits	Ada	ACC	113.013741	8.281988
Opentuner	Digits	Ada	ACC	106.626931	57.406496
Skopt	Digits	Ada	ACC	95.538424	505.125378
PySOT	Digits	Ada	ACC	89.816945	200.472104
Hyperopt	Digits	Ada	ACC	85.187450	354.000313
Random-search	Digits	Ada	ACC	80.936224	197.588714
TuRBO	Digits	Ada	ACC	79.578370	286.188817
TuRBO+	Digits	Ada	ACC	77.540995	279.358366
Nevergrad (1+1)	Digits	Ada	ACC	69.909418	239.320766
BOHB-BB	Digits	Ada	ACC	69.669304	174.603038

Table 72: Digits with ACC loss for tuning Knn model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Digits	Knn	ACC	100.040728	0.000000
BOHB-BB	Digits	Knn	ACC	100.039388	0.000036
Random-search	Digits	Knn	ACC	100.036708	0.000096
Hyperopt	Digits	Knn	ACC	100.031348	0.000172
Opentuner	Digits	Knn	ACC	99.657488	2.937454
PySOT	Digits	Knn	ACC	99.656148	2.936409
TuRBO+	Digits	Knn	ACC	98.884308	7.890293
Nevergrad (1+1)	Digits	Knn	ACC	98.499729	9.868794
TuRBO	Digits	Knn	ACC	97.355369	14.097037
Skopt	Digits	Knn	ACC	97.351349	14.074483

Table 73: Digits with ACC loss for tuning Lasso model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Digits	Lasso	ACC	99.812506	0.000015
TuRBO	Digits	Lasso	ACC	99.579356	19.648246
TuRBO+	Digits	Lasso	ACC	99.567177	9.142503
PySOT	Digits	Lasso	ACC	98.824228	9.445856
Skopt	Digits	Lasso	ACC	98.573678	4.903416
Hyperopt	Digits	Lasso	ACC	98.320519	8.122499
Nevergrad (1+1)	Digits	Lasso	ACC	98.077799	42.798357
BOHB-BB	Digits	Lasso	ACC	97.067772	9.136628
Opentuner	Digits	Lasso	ACC	97.067772	14.404687
Random-search	Digits	Lasso	ACC	95.815894	9.460591

Table 74: Digits with ACC loss for tuning Linear model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Digits	Linear	ACC	92.457319	0.000058
PySOT	Digits	Linear	ACC	92.439986	0.000736
Opentuner	Digits	Linear	ACC	92.422652	0.001039
TuRBO	Digits	Linear	ACC	92.104461	2.529241
Hyperopt	Digits	Linear	ACC	92.092080	2.520398
Skopt	Digits	Linear	ACC	91.351699	6.751230
TuRBO+	Digits	Linear	ACC	90.680651	9.996151
BOHB-BB	Digits	Linear	ACC	90.663318	9.987309
Random-search	Digits	Linear	ACC	90.296841	11.103892
Nevergrad (1+1)	Digits	Linear	ACC	87.836744	43.931074

Table 75: Digits with NLL loss for tuning DT model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is TuRBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
TuRBO	Digits	DT	NLL	105.314498	15.866589
HEBO	Digits	DT	NLL	104.981244	18.690339
PySOT	Digits	DT	NLL	100.712129	25.343726
Skopt	Digits	DT	NLL	100.695390	158.169850
Hyperopt	Digits	DT	NLL	91.082975	73.272063
TuRBO+	Digits	DT	NLL	85.781199	240.286219
BOHB-BB	Digits	DT	NLL	83.167442	110.410120
Opentuner	Digits	DT	NLL	80.876262	324.737850
Random-search	Digits	DT	NLL	75.654077	204.351641
Nevergrad (1+1)	Digits	DT	NLL	62.949280	1582.133768

Table 76: Digits with NLL loss for tuning MLP-adam model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is TuRBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
TuRBO	Digits	MLP-adam	NLL	104.936214	3.807905
HEBO	Digits	MLP-adam	NLL	103.888110	2.816804
PySOT	Digits	MLP-adam	NLL	102.476835	6.207587
Hyperopt	Digits	MLP-adam	NLL	101.270862	5.540137
TuRBO+	Digits	MLP-adam	NLL	100.550974	8.449016
Nevergrad (1+1)	Digits	MLP-adam	NLL	99.537718	15.391778
Opentuner	Digits	MLP-adam	NLL	95.718504	17.973634
BOHB-BB	Digits	MLP-adam	NLL	95.628722	17.000823
Random-search	Digits	MLP-adam	NLL	95.303772	10.415455
Skopt	Digits	MLP-adam	NLL	94.395163	12.085910

Table 77: Digits with NLL loss for tuning MLP-SGD model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Digits	MLP-SGD	NLL	100.316022	0.033631
TuRBO	Digits	MLP-SGD	NLL	100.247280	0.020780
PySOT	Digits	MLP-SGD	NLL	99.817072	0.191541
Hyperopt	Digits	MLP-SGD	NLL	99.559253	0.118122
Opentuner	Digits	MLP-SGD	NLL	99.166567	0.476585
Skopt	Digits	MLP-SGD	NLL	99.120810	0.255960
TuRBO+	Digits	MLP-SGD	NLL	98.906896	0.367730
BOHB-BB	Digits	MLP-SGD	NLL	98.642683	0.487494
Nevergrad (1+1)	Digits	MLP-SGD	NLL	98.611544	3.487067
Random-search	Digits	MLP-SGD	NLL	98.378973	0.516273

Table 78: Digits with NLL loss for tuning RF model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Digits	RF	NLL	124.468725	0.297160
PySOT	Digits	RF	NLL	124.073856	0.336978
TuRBO	Digits	RF	NLL	121.787294	21.428797
Skopt	Digits	RF	NLL	120.230526	2.533880
Hyperopt	Digits	RF	NLL	99.580977	158.609928
Nevergrad (1+1)	Digits	RF	NLL	84.573865	3268.993667
TuRBO+	Digits	RF	NLL	81.415754	885.535630
Opentuner	Digits	RF	NLL	80.271515	997.408185
BOHB-BB	Digits	RF	NLL	69.251670	441.233007
Random-search	Digits	RF	NLL	67.481493	469.513384

Table 79: Digits with NLL loss for tuning SVM model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Digits	SVM	NLL	101.325277	5.161457
Opentuner	Digits	SVM	NLL	100.363949	7.971372
TuRBO	Digits	SVM	NLL	99.999409	4.000423
TuRBO+	Digits	SVM	NLL	99.759626	5.784031
Skopt	Digits	SVM	NLL	99.695385	9.680923
Hyperopt	Digits	SVM	NLL	98.923652	5.243537
Nevergrad (1+1)	Digits	SVM	NLL	98.668450	15.533980
PySOT	Digits	SVM	NLL	98.531429	3.305281
Random-search	Digits	SVM	NLL	97.508303	4.801569
BOHB-BB	Digits	SVM	NLL	96.832038	8.596550

Table 80: Digits with NLL loss for tuning Ada model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Digits	Ada	NLL	111.973166	5.026622
Opentuner	Digits	Ada	NLL	108.973046	9.183543
Skopt	Digits	Ada	NLL	107.670995	53.503481
TuRBO	Digits	Ada	NLL	106.933065	66.341873
Nevergrad (1+1)	Digits	Ada	NLL	103.648617	104.403755
PySOT	Digits	Ada	NLL	103.613606	18.883842
Hyperopt	Digits	Ada	NLL	102.669338	21.172717
TuRBO+	Digits	Ada	NLL	101.012913	48.423642
BOHB-BB	Digits	Ada	NLL	94.250499	161.887190
Random-search	Digits	Ada	NLL	93.337020	91.626773

Table 81: Digits with NLL loss for tuning Knn model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Digits	Knn	NLL	99.032850	0.000000
Opentuner	Digits	Knn	NLL	99.032850	0.000000
Random-search	Digits	Knn	NLL	98.292524	10.961651
PySOT	Digits	Knn	NLL	97.818694	8.793321
Hyperopt	Digits	Knn	NLL	97.009257	12.931355
TuRBO+	Digits	Knn	NLL	97.009257	12.931355
TuRBO	Digits	Knn	NLL	96.604538	14.483117
BOHB-BB	Digits	Knn	NLL	96.002435	31.845710
Nevergrad (1+1)	Digits	Knn	NLL	89.407410	309.374772
Skopt	Digits	Knn	NLL	79.757469	260.984575

Table 82: Digits with NLL loss for tuning Lasso model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Digits	Lasso	NLL	100.137158	0.000255
TuRBO	Digits	Lasso	NLL	100.129152	0.000276
PySOT	Digits	Lasso	NLL	100.041641	0.002026
Skopt	Digits	Lasso	NLL	100.012443	0.016183
Opentuner	Digits	Lasso	NLL	99.993744	0.034008
Hyperopt	Digits	Lasso	NLL	99.916651	0.018303
Random-search	Digits	Lasso	NLL	99.336829	0.333326
TuRBO+	Digits	Lasso	NLL	99.205388	5.479509
BOHB-BB	Digits	Lasso	NLL	98.596349	2.171619
Nevergrad (1+1)	Digits	Lasso	NLL	97.693951	18.423865

Table 83: Digits with NLL loss for tuning Linear model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Digits	Linear	NLL	100.615794	0.000224
TuRBO	Digits	Linear	NLL	100.608559	0.000513
PySOT	Digits	Linear	NLL	100.585723	0.001391
Hyperopt	Digits	Linear	NLL	100.428061	0.021496
Opentuner	Digits	Linear	NLL	100.323235	0.175515
TuRBO+	Digits	Linear	NLL	100.316800	0.117834
Skopt	Digits	Linear	NLL	99.709029	2.042606
BOHB-BB	Digits	Linear	NLL	98.930379	4.622083
Random-search	Digits	Linear	NLL	98.655079	3.097357
Nevergrad (1+1)	Digits	Linear	NLL	98.269646	48.924742

Table 84: Iris with ACC loss for tuning DT model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Iris	DT	ACC	99.247213	2.809044
PySOT	Iris	DT	ACC	99.018466	3.525075
Skopt	Iris	DT	ACC	98.789719	2.478568
Nevergrad (1+1)	Iris	DT	ACC	98.560972	1.872696
TuRBO	Iris	DT	ACC	98.446598	1.253054
TuRBO+	Iris	DT	ACC	98.217851	1.032737
Random-search	Iris	DT	ACC	97.874730	1.046507
Hyperopt	Iris	DT	ACC	97.760356	0.261627
BOHB-BB	Iris	DT	ACC	97.645982	0.000000
Opentuner	Iris	DT	ACC	97.645982	1.652379

Table 85: Iris with ACC loss for tuning MLP-adam model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Iris	MLP-adam	ACC	97.523595	2.387333
TuRBO	Iris	MLP-adam	ACC	97.404537	1.477162
Opentuner	Iris	MLP-adam	ACC	97.166421	1.954629
PySOT	Iris	MLP-adam	ACC	97.047363	1.551766
Nevergrad (1+1)	Iris	MLP-adam	ACC	97.047363	2.745433
Skopt	Iris	MLP-adam	ACC	96.928305	1.119062
TuRBO+	Iris	MLP-adam	ACC	96.928305	2.312729
Hyperopt	Iris	MLP-adam	ACC	96.213958	2.670829
Random-search	Iris	MLP-adam	ACC	96.094900	4.416566
BOHB-BB	Iris	MLP-adam	ACC	95.142437	4.177833

Table 86: Iris with ACC loss for tuning MLP-SGD model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is TuRBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
TuRBO	Iris	MLP-SGD	ACC	100.859022	1.177745
HEBO	Iris	MLP-SGD	ACC	100.716068	0.962632
PySOT	Iris	MLP-SGD	ACC	100.644591	0.516272
Skopt	Iris	MLP-SGD	ACC	100.358684	1.419747
Nevergrad (1+1)	Iris	MLP-SGD	ACC	100.001300	1.823085
Opentuner	Iris	MLP-SGD	ACC	98.643240	16.563718
TuRBO+	Iris	MLP-SGD	ACC	97.070749	76.989020
Hyperopt	Iris	MLP-SGD	ACC	95.784166	16.778831
BOHB-BB	Iris	MLP-SGD	ACC	93.854291	42.221346
Random-search	Iris	MLP-SGD	ACC	92.853615	32.799388

Table 87: Iris with ACC loss for tuning RF model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is Nevergrad.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
Nevergrad (1+1)	Iris	RF	ACC	99.724330	0.835050
TuRBO	Iris	RF	ACC	99.585230	0.488810
TuRBO+	Iris	RF	ACC	99.585230	0.896151
Opentuner	Iris	RF	ACC	99.446131	1.120189
HEBO	Iris	RF	ACC	99.376581	0.707755
PySOT	Iris	RF	ACC	99.376581	0.707755
Skopt	Iris	RF	ACC	99.307031	0.896151
Hyperopt	Iris	RF	ACC	99.028832	0.529544
BOHB-BB	Iris	RF	ACC	98.681082	0.911426
Random-search	Iris	RF	ACC	98.541983	0.667021

Table 88: Iris with ACC loss for tuning SVM model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is TuRBO+.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
TuRBO+	Iris	SVM	ACC	97.042816	78.603995
Hyperopt	Iris	SVM	ACC	95.047170	95.372847
PySOT	Iris	SVM	ACC	94.049347	100.613114
Opentuner	Iris	SVM	ACC	91.055878	103.757273
Random-search	Iris	SVM	ACC	88.062409	88.036474
TuRBO	Iris	SVM	ACC	88.062409	88.036474
HEBO	Iris	SVM	ACC	87.064586	78.603995
BOHB-BB	Iris	SVM	ACC	85.068940	53.450717
Skopt	Iris	SVM	ACC	85.068940	53.450717
Nevergrad (1+1)	Iris	SVM	ACC	83.073295	19.913012

Table 89: Iris with ACC loss for tuning Ada model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is BOHB.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
BOHB-BB	Iris	Ada	ACC	101.593252	0.000000
HEBO	Iris	Ada	ACC	101.593252	0.000000
Hyperopt	Iris	Ada	ACC	101.593252	0.000000
PySOT	Iris	Ada	ACC	101.593252	0.000000
Random-search	Iris	Ada	ACC	101.593252	0.000000
TuRBO	Iris	Ada	ACC	101.593252	0.000000
TuRBO+	Iris	Ada	ACC	101.593252	0.000000
Skopt	Iris	Ada	ACC	100.304592	33.212881
Nevergrad (1+1)	Iris	Ada	ACC	90.070291	788.023678
Opentuner	Iris	Ada	ACC	83.552015	146.835896

Table 90: Iris with ACC loss for tuning Knn model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is BOHB.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
BOHB-BB	Iris	Knn	ACC	74.768089	0.000000
HEBO	Iris	Knn	ACC	74.768089	0.000000
Hyperopt	Iris	Knn	ACC	74.768089	0.000000
Opentuner	Iris	Knn	ACC	74.768089	0.000000
PySOT	Iris	Knn	ACC	74.768089	0.000000
Random-search	Iris	Knn	ACC	74.768089	0.000000
Skopt	Iris	Knn	ACC	74.768089	0.000000
TuRBO	Iris	Knn	ACC	74.768089	0.000000
TuRBO+	Iris	Knn	ACC	74.768089	0.000000
Nevergrad (1+1)	Iris	Knn	ACC	62.012987	239.757557

Table 91: Iris with ACC loss for tuning Lasso model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is BOHB.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
BOHB-BB	Iris	Lasso	ACC	101.755486	0.000000
HEBO	Iris	Lasso	ACC	101.755486	0.000000
Hyperopt	Iris	Lasso	ACC	101.755486	0.000000
Opentuner	Iris	Lasso	ACC	101.755486	0.000000
PySOT	Iris	Lasso	ACC	101.755486	0.000000
TuRBO+	Iris	Lasso	ACC	100.893417	14.863258
Skopt	Iris	Lasso	ACC	100.031348	28.161963
Random-search	Iris	Lasso	ACC	100.031348	28.161963
TuRBO	Iris	Lasso	ACC	100.031348	28.161963
Nevergrad (1+1)	Iris	Lasso	ACC	91.410658	75.098567

Table 92: Iris with ACC loss for tuning Linear model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Iris	Linear	ACC	102.086438	0.000000
PySOT	Iris	Linear	ACC	102.086438	0.000000
Opentuner	Iris	Linear	ACC	101.061848	20.995700
Skopt	Iris	Linear	ACC	100.037258	39.781326
Hyperopt	Iris	Linear	ACC	99.012668	56.356879
TuRBO	Iris	Linear	ACC	99.012668	56.356879
TuRBO+	Iris	Linear	ACC	99.012668	56.356879
BOHB-BB	Iris	Linear	ACC	97.988077	70.722358
Random-search	Iris	Linear	ACC	91.840537	110.503685
Nevergrad (1+1)	Iris	Linear	ACC	87.742176	92.823095

Table 93: Iris with NLL loss for tuning DT model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is Skopt.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
Skopt	Iris	DT	NLL	90.203906	10.555548
BOHB-BB	Iris	DT	NLL	89.526762	6.488902
Random-search	Iris	DT	NLL	89.082026	5.704656
HEBO	Iris	DT	NLL	88.967765	17.831672
TuRBO+	Iris	DT	NLL	88.562978	52.676524
PySOT	Iris	DT	NLL	87.994255	81.937308
Opentuner	Iris	DT	NLL	87.961800	83.095049
TuRBO	Iris	DT	NLL	86.393584	167.928590
Hyperopt	Iris	DT	NLL	86.051753	41.891055
Nevergrad (1+1)	Iris	DT	NLL	72.442142	312.271956

Table 94: Iris with NLL loss for tuning MLP-adam model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Iris	MLP-adam	NLL	105.645493	6.054549
PySOT	Iris	MLP-adam	NLL	102.039797	13.350468
TuRBO	Iris	MLP-adam	NLL	100.140611	19.185383
Skopt	Iris	MLP-adam	NLL	99.621989	17.712939
Nevergrad (1+1)	Iris	MLP-adam	NLL	98.212997	53.506634
TuRBO+	Iris	MLP-adam	NLL	96.282661	47.747129
Hyperopt	Iris	MLP-adam	NLL	96.243907	16.679246
Opentuner	Iris	MLP-adam	NLL	96.133265	28.635379
BOHB-BB	Iris	MLP-adam	NLL	93.098113	9.215071
Random-search	Iris	MLP-adam	NLL	92.646785	10.125014

Table 95: Iris with NLL loss for tuning MLP-SGD model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Iris	MLP-SGD	NLL	102.090160	4.808520
TuRBO	Iris	MLP-SGD	NLL	100.931431	18.282403
PySOT	Iris	MLP-SGD	NLL	100.898255	17.026358
Skopt	Iris	MLP-SGD	NLL	95.476812	11.295356
Opentuner	Iris	MLP-SGD	NLL	93.021987	32.453837
TuRBO+	Iris	MLP-SGD	NLL	91.060150	90.564666
Nevergrad (1+1)	Iris	MLP-SGD	NLL	89.963939	334.753752
Hyperopt	Iris	MLP-SGD	NLL	83.627458	120.346759
Random-search	Iris	MLP-SGD	NLL	82.555939	76.757676
BOHB-BB	Iris	MLP-SGD	NLL	80.606157	122.102085

Table 96: Iris with NLL loss for tuning RF model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Iris	RF	NLL	101.145768	0.577105
PySOT	Iris	RF	NLL	100.876231	0.403809
Skopt	Iris	RF	NLL	100.444269	0.670743
TuRBO	Iris	RF	NLL	100.137104	0.354499
Nevergrad (1+1)	Iris	RF	NLL	99.974386	2.958654
Hyperopt	Iris	RF	NLL	99.128911	0.638326
Random-search	Iris	RF	NLL	98.705274	1.341589
BOHB-BB	Iris	RF	NLL	98.532111	0.918847
TuRBO+	Iris	RF	NLL	97.755200	23.505634
Opentuner	Iris	RF	NLL	97.597173	30.398967

Table 97: Iris with NLL loss for tuning SVM model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Iris	SVM	NLL	96.905678	2.563359
TuRBO	Iris	SVM	NLL	96.560351	5.102559
Nevergrad (1+1)	Iris	SVM	NLL	96.415399	4.986315
PySOT	Iris	SVM	NLL	95.705867	1.884251
Opentuner	Iris	SVM	NLL	95.169892	3.064513
TuRBO+	Iris	SVM	NLL	95.156637	3.952362
Hyperopt	Iris	SVM	NLL	92.938343	3.272897
Skopt	Iris	SVM	NLL	91.588572	15.500143
Random-search	Iris	SVM	NLL	89.166348	7.772873
BOHB-BB	Iris	SVM	NLL	88.699355	6.977295

Table 98: Iris with NLL loss for tuning Ada model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is Opentuner.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
Opentuner	Iris	Ada	NLL	111.286667	635.556925
BOHB-BB	Iris	Ada	NLL	92.632477	293.408333
Skopt	Iris	Ada	NLL	90.629536	341.650262
Nevergrad (1+1)	Iris	Ada	NLL	90.294262	381.283394
Hyperopt	Iris	Ada	NLL	87.128984	169.159183
Random-search	Iris	Ada	NLL	86.929736	170.566677
TuRBO+	Iris	Ada	NLL	86.754284	416.384095
TuRBO	Iris	Ada	NLL	86.711063	128.643690
HEBO	Iris	Ada	NLL	83.893227	159.359904
PySOT	Iris	Ada	NLL	80.149037	9.698680

Table 99: Iris with NLL loss for tuning Knn model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Iris	Knn	NLL	91.554078	0.000000
Hyperopt	Iris	Knn	NLL	91.554078	0.000000
Opentuner	Iris	Knn	NLL	91.554078	0.000000
BOHB-BB	Iris	Knn	NLL	90.979918	3.123091
PySOT	Iris	Knn	NLL	90.405758	5.552162
Random-search	Iris	Knn	NLL	90.242912	7.487026
Skopt	Iris	Knn	NLL	90.118678	6.506440
TuRBO+	Iris	Knn	NLL	89.544518	7.894481
TuRBO	Iris	Knn	NLL	87.307469	98.518070
Nevergrad (1+1)	Iris	Knn	NLL	85.145932	80.598094

Table 100: Iris with NLL loss for tuning Lasso model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is TuRBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
TuRBO	Iris	Lasso	NLL	101.656336	0.547832
Skopt	Iris	Lasso	NLL	101.385957	0.236491
Opentuner	Iris	Lasso	NLL	101.322010	0.200262
HEBO	Iris	Lasso	NLL	101.311644	0.108916
PySOT	Iris	Lasso	NLL	101.302536	0.136747
Nevergrad (1+1)	Iris	Lasso	NLL	101.259029	0.404304
TuRBO+	Iris	Lasso	NLL	101.016469	0.178279
Hyperopt	Iris	Lasso	NLL	100.925838	0.094444
Random-search	Iris	Lasso	NLL	100.729706	0.122522
BOHB-BB	Iris	Lasso	NLL	100.523347	0.052615

Table 101: Iris with NLL loss for tuning Linear model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is TuRBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
TuRBO	Iris	Linear	NLL	101.260741	0.000089
PySOT	Iris	Linear	NLL	101.243390	0.000561
Opentuner	Iris	Linear	NLL	101.230232	0.000970
HEBO	Iris	Linear	NLL	101.225588	0.001167
Nevergrad (1+1)	Iris	Linear	NLL	101.210259	0.008063
Hyperopt	Iris	Linear	NLL	101.166301	0.001982
TuRBO+	Iris	Linear	NLL	101.127954	0.005476
Skopt	Iris	Linear	NLL	101.008577	0.277775
BOHB-BB	Iris	Linear	NLL	100.753901	0.263597
Random-search	Iris	Linear	NLL	100.630954	0.813351

Table 102: Wine with ACC loss for tuning DT model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is PySOT.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
PySOT	Wine	DT	ACC	102.640360	4.994593
Nevergrad (1+1)	Wine	DT	ACC	102.412606	12.231128
Skopt	Wine	DT	ACC	101.067267	5.199240
TuRBO	Wine	DT	ACC	99.968220	12.856860
HEBO	Wine	DT	ACC	99.608051	8.503162
TuRBO+	Wine	DT	ACC	99.152542	9.570280
Hyperopt	Wine	DT	ACC	98.697034	4.909287
Random-search	Wine	DT	ACC	97.534428	1.557289
BOHB-BB	Wine	DT	ACC	97.086864	0.453885
Opentuner	Wine	DT	ACC	97.086864	1.206916

Table 103: Wine with ACC loss for tuning MLP-adam model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Wine	MLP-adam	ACC	102.288087	3.576365
PySOT	Wine	MLP-adam	ACC	101.781214	16.573344
TuRBO	Wine	MLP-adam	ACC	100.990836	41.217673
Skopt	Wine	MLP-adam	ACC	100.727377	8.173123
TuRBO+	Wine	MLP-adam	ACC	97.943872	29.486961
Hyperopt	Wine	MLP-adam	ACC	96.540664	18.874205
Nevergrad (1+1)	Wine	MLP-adam	ACC	92.328179	161.347003
Random-search	Wine	MLP-adam	ACC	90.816151	63.213250
Opentuner	Wine	MLP-adam	ACC	89.183849	103.560060
BOHB-BB	Wine	MLP-adam	ACC	87.542955	57.246663

Table 104: Wine with ACC loss for tuning MLP-SGD model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Wine	MLP-SGD	ACC	98.576865	76.332793
Skopt	Wine	MLP-SGD	ACC	96.955860	72.699785
TuRBO	Wine	MLP-SGD	ACC	95.133181	291.308493
PySOT	Wine	MLP-SGD	ACC	94.414003	344.889994
Hyperopt	Wine	MLP-SGD	ACC	87.423896	76.509456
Nevergrad (1+1)	Wine	MLP-SGD	ACC	84.821157	393.474837
TuRBO+	Wine	MLP-SGD	ACC	84.303653	163.191728
Random-search	Wine	MLP-SGD	ACC	79.113394	53.744005
BOHB-BB	Wine	MLP-SGD	ACC	78.576865	147.437959
Opentuner	Wine	MLP-SGD	ACC	53.089802	324.782282

Table 105: Wine with ACC loss for tuning RF model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is Skopt.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
Skopt	Wine	RF	ACC	100.599500	0.495957
HEBO	Wine	RF	ACC	100.118651	0.875418
PySOT	Wine	RF	ACC	99.950042	2.588995
Hyperopt	Wine	RF	ACC	99.835554	1.065562
TuRBO	Wine	RF	ACC	99.054954	1.575060
TuRBO+	Wine	RF	ACC	99.029975	3.693421
Nevergrad (1+1)	Wine	RF	ACC	98.986261	1.535684
Opentuner	Wine	RF	ACC	98.576187	1.296012
Random-search	Wine	RF	ACC	97.531224	2.672463
BOHB-BB	Wine	RF	ACC	97.191923	1.483359

Table 106: Wine with ACC loss for tuning SVM model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Wine	SVM	ACC	101.089494	23.739951
Skopt	Wine	SVM	ACC	96.050584	42.094346
PySOT	Wine	SVM	ACC	95.603113	30.537137
TuRBO	Wine	SVM	ACC	95.486381	43.769737
Opentuner	Wine	SVM	ACC	92.743191	54.413749
Hyperopt	Wine	SVM	ACC	91.031128	20.295538
Nevergrad (1+1)	Wine	SVM	ACC	90.408560	17.088987
TuRBO+	Wine	SVM	ACC	89.435798	134.768764
Random-search	Wine	SVM	ACC	83.229572	31.251121
BOHB-BB	Wine	SVM	ACC	82.140078	30.374578

Table 107: Wine with ACC loss for tuning Ada model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Wine	Ada	ACC	102.249135	61.472091
Opentuner	Wine	Ada	ACC	87.525952	89.791159
TuRBO	Wine	Ada	ACC	87.456747	186.588749
Nevergrad (1+1)	Wine	Ada	ACC	87.024221	445.327648
PySOT	Wine	Ada	ACC	85.121107	95.696071
TuRBO+	Wine	Ada	ACC	83.044983	134.639949
Hyperopt	Wine	Ada	ACC	80.536332	97.458944
Skopt	Wine	Ada	ACC	77.595156	548.313094
Random-search	Wine	Ada	ACC	73.702422	43.178551
BOHB-BB	Wine	Ada	ACC	72.698962	81.226341

Table 108: Wine with ACC loss for tuning Knn model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Wine	Knn	ACC	100.000000	0.000000
Nevergrad (1+1)	Wine	Knn	ACC	100.000000	0.000000
Opentuner	Wine	Knn	ACC	100.000000	0.000000
Hyperopt	Wine	Knn	ACC	89.612546	219.840273
TuRBO+	Wine	Knn	ACC	89.077491	317.790364
Random-search	Wine	Knn	ACC	83.994465	285.251803
Skopt	Wine	Knn	ACC	83.791513	325.106208
BOHB-BB	Wine	Knn	ACC	82.416974	287.574559
TuRBO	Wine	Knn	ACC	70.313653	597.435894
PySOT	Wine	Knn	ACC	70.036900	236.259826

Table 109: Wine with ACC loss for tuning Lasso model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is PySOT.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
PySOT	Wine	Lasso	ACC	98.390805	51.790197
HEBO	Wine	Lasso	ACC	95.172414	139.015792
Hyperopt	Wine	Lasso	ACC	87.068966	264.079243
TuRBO+	Wine	Lasso	ACC	85.517241	597.089235
TuRBO	Wine	Lasso	ACC	83.908046	708.707957
Opentuner	Wine	Lasso	ACC	79.195402	685.469123
BOHB-BB	Wine	Lasso	ACC	79.022989	358.508737
Skopt	Wine	Lasso	ACC	67.701149	545.285131
Random-search	Wine	Lasso	ACC	67.643678	327.273296
Nevergrad (1+1)	Wine	Lasso	ACC	66.091954	818.782986

Table 110: Wine with ACC loss for tuning Linear model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Wine	Linear	ACC	100.000000	0.000000
PySOT	Wine	Linear	ACC	98.372093	53.001622
Opentuner	Wine	Linear	ACC	98.313953	56.855057
Hyperopt	Wine	Linear	ACC	96.686047	104.078307
TuRBO	Wine	Linear	ACC	93.372093	300.603456
Random-search	Wine	Linear	ACC	90.116279	240.030173
TuRBO+	Wine	Linear	ACC	85.058140	634.094532
Nevergrad (1+1)	Wine	Linear	ACC	81.918605	1182.356750
BOHB-BB	Wine	Linear	ACC	81.686047	289.043153
Skopt	Wine	Linear	ACC	75.116279	910.933364

Table 111: Wine with NLL loss for tuning DT model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is PySOT.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
PySOT	Wine	DT	NLL	100.735933	7.443055
Skopt	Wine	DT	NLL	100.221883	3.394850
HEBO	Wine	DT	NLL	99.832918	3.739025
TuRBO+	Wine	DT	NLL	99.674754	3.371033
BOHB-BB	Wine	DT	NLL	98.178211	4.342658
TuRBO	Wine	DT	NLL	98.058048	83.877777
Random-search	Wine	DT	NLL	97.376968	10.077564
Opentuner	Wine	DT	NLL	97.049151	36.266865
Hyperopt	Wine	DT	NLL	96.334956	34.297016
Nevergrad (1+1)	Wine	DT	NLL	89.297835	389.090524

Table 112: Wine with NLL loss for tuning MLP-adam model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is TuRBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
TuRBO	Wine	MLP-adam	NLL	100.249989	0.565669
Skopt	Wine	MLP-adam	NLL	99.981264	0.167045
PySOT	Wine	MLP-adam	NLL	99.926231	0.303040
HEBO	Wine	MLP-adam	NLL	99.565384	0.763738
Hyperopt	Wine	MLP-adam	NLL	99.449852	0.504218
Opentuner	Wine	MLP-adam	NLL	98.831651	1.222083
TuRBO+	Wine	MLP-adam	NLL	98.571847	2.666428
Random-search	Wine	MLP-adam	NLL	98.111233	0.855310
BOHB-BB	Wine	MLP-adam	NLL	97.819061	3.117717
Nevergrad (1+1)	Wine	MLP-adam	NLL	97.151962	33.698618

Table 113: Wine with NLL loss for tuning MLP-SGD model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Wine	MLP-SGD	NLL	100.065123	0.406916
Hyperopt	Wine	MLP-SGD	NLL	99.764357	0.625817
TuRBO	Wine	MLP-SGD	NLL	99.560475	0.802736
Skopt	Wine	MLP-SGD	NLL	99.420960	0.566385
Nevergrad (1+1)	Wine	MLP-SGD	NLL	99.358015	0.910527
PySOT	Wine	MLP-SGD	NLL	99.104753	0.965483
TuRBO+	Wine	MLP-SGD	NLL	98.940260	0.886137
Random-search	Wine	MLP-SGD	NLL	98.692717	0.489974
BOHB-BB	Wine	MLP-SGD	NLL	98.358461	0.496547
Opentuner	Wine	MLP-SGD	NLL	98.254971	0.292497

Table 114: Wine with NLL loss for tuning RF model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Wine	RF	NLL	104.649110	2.581885
PySOT	Wine	RF	NLL	104.355692	1.566376
Skopt	Wine	RF	NLL	104.170679	0.665546
TuRBO	Wine	RF	NLL	104.031688	4.984665
Nevergrad (1+1)	Wine	RF	NLL	101.465863	15.172791
Hyperopt	Wine	RF	NLL	99.738821	6.443466
TuRBO+	Wine	RF	NLL	99.334611	7.498471
BOHB-BB	Wine	RF	NLL	97.437084	5.523676
Random-search	Wine	RF	NLL	97.361000	2.798394
Opentuner	Wine	RF	NLL	97.308388	2.661822

Table 115: Wine with NLL loss for tuning SVM model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Wine	SVM	NLL	93.675487	32.073349
PySOT	Wine	SVM	NLL	93.111866	33.743562
TuRBO	Wine	SVM	NLL	93.062091	34.028682
TuRBO+	Wine	SVM	NLL	91.816538	141.971101
Opentuner	Wine	SVM	NLL	91.608167	31.115479
Skopt	Wine	SVM	NLL	89.843405	31.673409
BOHB-BB	Wine	SVM	NLL	86.866351	32.611321
Hyperopt	Wine	SVM	NLL	86.472718	10.534871
Random-search	Wine	SVM	NLL	83.877620	47.933648
Nevergrad (1+1)	Wine	SVM	NLL	80.529537	219.352311

Table 116: Wine with NLL loss for tuning Ada model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Wine	Ada	NLL	99.727993	0.273233
Skopt	Wine	Ada	NLL	98.863208	4.590542
TuRBO	Wine	Ada	NLL	98.310428	3.476209
Opentuner	Wine	Ada	NLL	97.429313	5.993673
TuRBO+	Wine	Ada	NLL	97.067541	12.860766
PySOT	Wine	Ada	NLL	96.701820	9.045832
Hyperopt	Wine	Ada	NLL	95.813459	4.407929
Nevergrad (1+1)	Wine	Ada	NLL	95.011185	61.697984
Random-search	Wine	Ada	NLL	93.567555	13.596675
BOHB-BB	Wine	Ada	NLL	92.183239	14.787665

Table 117: Wine with NLL loss for tuning Knn model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Wine	Knn	NLL	100.000000	0.000000
Opentuner	Wine	Knn	NLL	100.000000	0.000000
PySOT	Wine	Knn	NLL	99.973148	0.014421
Hyperopt	Wine	Knn	NLL	99.946296	0.027323
TuRBO+	Wine	Knn	NLL	99.919444	0.038708
Skopt	Wine	Knn	NLL	99.897748	0.209108
TuRBO	Wine	Knn	NLL	99.892592	0.048574
Random-search	Wine	Knn	NLL	99.675015	0.449657
BOHB-BB	Wine	Knn	NLL	99.517722	0.706332
Nevergrad (1+1)	Wine	Knn	NLL	97.615849	8.701040

Table 118: Wine with NLL loss for tuning Lasso model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Wine	Lasso	NLL	99.827097	0.033778
Nevergrad (1+1)	Wine	Lasso	NLL	99.820084	0.022893
TuRBO	Wine	Lasso	NLL	99.755145	0.024582
TuRBO+	Wine	Lasso	NLL	99.708446	0.317074
PySOT	Wine	Lasso	NLL	99.582104	0.021060
Hyperopt	Wine	Lasso	NLL	99.469131	0.030572
Opentuner	Wine	Lasso	NLL	99.448682	0.204748
BOHB-BB	Wine	Lasso	NLL	99.393568	0.047506
Skopt	Wine	Lasso	NLL	99.381857	1.114643
Random-search	Wine	Lasso	NLL	99.110200	0.317902

Table 119: Wine with NLL loss for tuning Linear model, averaged over 20 seeds. Optimiser for this task with highest mean normalised score is HEBO.

Algorithm	Dataset	Model	Metric	Normalised Score	Variance
HEBO	Wine	Linear	NLL	99.532690	0.620171
TuRBO	Wine	Linear	NLL	99.437383	0.618966
Opentuner	Wine	Linear	NLL	98.930409	2.544485
PySOT	Wine	Linear	NLL	98.501752	1.017364
Nevergrad (1+1)	Wine	Linear	NLL	97.886057	34.425637
TuRBO+	Wine	Linear	NLL	97.200873	3.720058
Hyperopt	Wine	Linear	NLL	96.307423	2.606512
Skopt	Wine	Linear	NLL	95.972270	38.830724
Random-search	Wine	Linear	NLL	94.013264	4.158990
BOHB-BB	Wine	Linear	NLL	93.128692	5.936852

References

Abdolshah, M., Shilton, A., Rana, S., Gupta, S., & Venkatesh, S. (2019). Multi-objective Bayesian optimisation with preferences over objectives. In *Advances in Neural Information Processing Systems*, Vol. 32, pp. 12235–12245.

Abdullah, M. A., Ren, H., Ammar, H. B., Milenkovic, V., Luo, R., Zhang, M., & Wang, J. (2019). Wasserstein robust reinforcement learning. arXiv preprint arXiv:1907.13196.

Ansel, J., Kamil, S., Veeramachaneni, K., Ragan-Kelley, J., Bosboom, J., O'Reilly, U.-M., & Amarasinghe, S. (2014). Opentuner: An extensible framework for program autotuning. In *Proceedings of the 23rd International Conference on Parallel Architectures and Compilation*, pp. 303–316.

Balandat, M., Karrer, B., Jiang, D., Daulton, S., Letham, B., Wilson, A. G., & Bakshy, E. (2020). BoTorch: A framework for efficient Monte-Carlo Bayesian optimization. *Advances in Neural Information Processing Systems*, *33*.

Bergstra, J., Komer, B., Eliasmith, C., Yamins, D., & Cox, D. D. (2015). Hyperopt: a Python library for model selection and hyperparameter optimization. *Computational Science & Discovery*, *8*(1), 014008.

Bertsimas, D., Nohadani, O., & Teo, K. M. (2010). Nonconvex robust optimization for problems with constraints. *INFORMS Journal on Computing*, *22*(1), 44–58.

Blank, J., & Deb, K. (2020). Pymoo: Multi-objective optimization in Python. *IEEE Access*, *8*, 89497–89509.

Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-Based Systems*, *46*, 109–132.

Bogunovic, I., Scarlett, J., Jegelka, S., & Cevher, V. (2018). Adversarially robust optimization with Gaussian processes. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 5765–5775.

Box, G. E., & Cox, D. R. (1964). An analysis of transformations. *Journal of the Royal Statistical Society: Series B (Methodological)*, *26*(2), 211–243.

Calandra, R. (2017). *Bayesian modeling for optimization and control in robotics*. Ph.D. thesis, Darmstadt, Technische Universität.

Calandra, R., Peters, J., Rasmussen, C. E., & Deisenroth, M. P. (2016). Manifold Gaussian processes for regression. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 3338–3345. IEEE.

Cowen-Rivers, A. I., Palenicek, D., Moens, V., Abdullah, M., Sootla, A., Wang, J., & Bou-Ammar, H. (2020). SAMBA: Safe model-based & active reinforcement learning. In *arXiv preprint arXiv:2006.09436*.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197.

Dua, D., & Graff, C. (2017). UCI machine learning repository. http://archive.ics.uci.edu/ml.

Eriksson, D., Bindel, D., & Shoemaker, C. A. (2019a). pySOT and POAP: An event-driven asynchronous framework for surrogate optimization. arXiv preprint arXiv:1908.00420.

Eriksson, D., Pearce, M., Gardner, J., Turner, R. D., & Poloczek, M. (2019b). Scalable global optimization via local Bayesian optimization. *Advances in Neural Information Processing Systems*, *32*, 5496–5507.

Eriksson, D., & Poloczek, M. (2021). Scalable constrained Bayesian optimization. In *International Conference on Artificial Intelligence and Statistics*, pp. 730–738. PMLR.

Falkner, S., Klein, A., & Hutter, F. (2018). Bohb: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning*, pp. 1437–1446. PMLR.

Fatima, M., & Pasha, M. (2017). Survey of machine learning algorithms for disease diagnostic. *Journal of Intelligent Learning Systems and Applications*, *09*, 1–16.

Fligner, M. A., & Killeen, T. J. (1976). Distribution-free two-sample tests for scale. *Journal of the American Statistical Association*, *71*(353), 210–213.

Gabillon, V., Tutunov, R., Valko, M., & Bou-Ammar, H. (2020). Derivative-free & order-robust optimisation. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, Vol. 108, pp. 2293–2303.

Griffiths, R.-R., Aldrick, A. A., Garcia-Ortegon, M., Lalchand, V. R., & Lee, A. A. (2021a). Achieving robustness to aleatoric uncertainty with heteroscedastic Bayesian optimisation. arXiv preprint arXiv:1910.07779.

Griffiths, R.-R., Jiang, J., Buisson, D. J. K., Wilkins, D., Gallo, L. C., Ingram, A., Lee, A. A., Grupe, D., Kara, E., Parker, M. L., & et al. (2021b). Modeling the multiwavelength variability of mrk 335 using Gaussian processes. *The Astrophysical Journal*, *914*(2), 144.

Hansen, N. (2016). The CMA evolution strategy: A tutorial. arXiv preprint arXiv:1604.00772.

Hoffman, M., Brochu, E., & de Freitas, N. (2011a). Portfolio allocation for Bayesian optimization. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, pp. 327–336.

Hoffman, M. D., Brochu, E., & de Freitas, N. (2011b). Portfolio allocation for bayesian optimization.. In *UAI*, pp. 327–336. Citeseer.

Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In *International Conference on Learning and Intelligent Optimization*, pp. 507–523.

Kandasamy, K., Dasarathy, G., Schneider, J., & Póczos, B. (2017). Multi-fidelity Bayesian optimisation with continuous approximations. In *International Conference on Machine Learning*, pp. 1799–1808.

Kandasamy, K., Neiswanger, W., Schneider, J., Póczos, B., & Xing, E. P. (2018). Neural architecture search with Bayesian optimisation and optimal transport. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 2020–2029.

Kandasamy, K., Vysyaraju, K. R., Neiswanger, W., Paria, B., Collins, C. R., Schneider, J., Poczos, B., & Xing, E. P. (2020). Tuning hyperparameters without grad students: Scalable and robust bayesian optimisation with dragonfly. *Journal of Machine Learning Research*, *21*(81), 1–27.

Kersting, K., Plagemann, C., Pfaff, P., & Burgard, W. (2007). Most likely heteroscedastic Gaussian process regression. In *Proceedings of the 24th International Conference on Machine Learning*, pp. 393–400.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y., & LeCun, Y. (Eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kingma, D. P., & Welling, M. (2014). Auto-encoding variational Bayes. In Bengio, Y., & LeCun, Y. (Eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Kirschner, J., Bogunovic, I., Jegelka, S., & Krause, A. (2020). Distributionally robust Bayesian optimization. In *International Conference on Artificial Intelligence and Statistics*, pp. 2174–2184.

Klein, A., Falkner, S., Mansur, N., & Hutter, F. (2017). RoBo: A flexible and robust Bayesian optimization framework in Python. In *In Neural Information Processing Systems 2017 Workshop on Bayesian Optimization*.

Knudde, N., van der Herten, J., Dhaene, T., & Couckuyt, I. (2017). GpflowOpt: a Bayesian optimization library using TensorFlow. In *Neural Information Processing Systems 2017 Workshop on Bayesian Optimization*.

Kuindersma, S. R., Grupen, R. A., & Barto, A. G. (2013). Variable risk control via stochastic optimization. *The International Journal of Robotics Research*, *32*(7), 806–825.

Kumaraswamy, P. (1980). A generalized probability density function for double-bounded random processes. *Journal of Hydrology*, *46*(1-2), 79–88.

Lázaro-Gredilla, M., & Titsias, M. K. (2011). Variational heteroscedastic Gaussian process regression. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pp. 841–848.

Levene, H. (1960). Contributions to probability and statistics. *Essays in Honor of Harold Hotelling*, *1*(1), 278–292.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2017). Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, *18*(1), 6765–6816.

Lin, T., Jin, C., & Jordan, M. (2020). On gradient descent ascent for nonconvex-concave minimax problems. In *International Conference on Machine Learning*, pp. 6083–6093.

Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., Van Der Laak, J. A., Van Ginneken, B., & Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. *Medical Image Analysis*, *42*, 60–88.

Liu, D. C., & Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Math. Programming*, *45*(3), 503–528.

Loshchilov, I., & Hutter, F. (2016). CMA-ES for hyperparameter optimization of deep neural networks. arXiv preprint arXiv:1604.07269.

Lyu, W., Yang, F., Yan, C., Zhou, D., & Zeng, X. (2018). Batch Bayesian optimization via multi-objective acquisition ensemble for automated analog circuit design. In *International Conference on Machine Learning*, pp. 3306–3314.

Maronas, J., Hamelijnck, O., Knoblauch, J., & Damoulas, T. (2020). Transforming Gaussian processes with normalizing flows. arXiv preprint arXiv:2011.01596.

Nguyen, V., & Osborne, M. A. (2020). Knowing the what but not the where in Bayesian optimization. In *International Conference on Machine Learning*, pp. 7317–7326.

Oh, C., Gavves, E., & Welling, M. (2018). Bock: Bayesian optimization with cylindrical kernels. In *International Conference on Machine Learning*, pp. 3868–3877.

Park, C., Borth, D. J., Wilson, N. S., Hunter, C. N., & Friedersdorf, F. J. (2020). Robust Gaussian process regression with a bias model. arXiv preprint arXiv:2001.04639.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Rapin, J., & Teytaud, O. (2018). Nevergrad - A gradient-free optimization platform. https://GitHub.com/FacebookResearch/Nevergrad.

Rasmussen, C. E., & Williams, C. K. (2006). *Gaussian Processes for Machine Learning*, Vol. 2. MIT press Cambridge, MA.

Rezende, D., & Mohamed, S. (2015). Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning*, Vol. 37, pp. 1530–1538.

Rios, G., & Tobar, F. (2019). Compositionally-warped Gaussian processes. *Neural Networks*, *118*, 235–246.

Sen, R., Kandasamy, K., & Shakkottai, S. (2018). Multi-fidelity black-box optimization with hierarchical partitions. In *International Conference on Machine Learning*, pp. 4538–4547. PMLR.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., & de Freitas, N. (2016). Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, *1*(104), 148–175.

Shahriari, B., Wang, Z., Hoffman, M. W., Bouchard-Côté, A., & de Freitas, N. (2014). An entropy search portfolio for Bayesian optimization. arXiv preprint arXiv:1406.4625.

Snelson, E., Rasmussen, C. E., & Ghahramani, Z. (2004). Warped Gaussian processes. *Advances in Neural Information Processing Systems*, *16*, 337–344.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pp. 2951–2959.

Snoek, J., Swersky, K., Zemel, R., & Adams, R. (2014). Input warping for Bayesian optimization of non-stationary functions. In *International Conference on Machine Learning*, pp. 1674–1682.

Springenberg, J. T., Klein, A., Falkner, S., & Hutter, F. (2016). Bayesian optimization with robust Bayesian neural networks. In *Advances in Neural Information Processing Systems*, pp. 4134–4142.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., & Abbeel, P. (2017). Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30. IEEE.

Tripuraneni, N., Stern, M., Jin, C., Regier, J., & Jordan, M. I. (2017). Stochastic cubic regularization for fast nonconvex optimization. *CoRR*, *abs/1711.02838*.

Turner, R., Eriksson, D., McCourt, M., Kiili, J., Laaksonen, E., Xu, Z., & Guyon, I. (2021). Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. *arXiv preprint arXiv:2104.10201*, *1*, 1.

Williams, C. K., & Rasmussen, C. E. (1996). Gaussian processes for regression. *MIT press Cambridge*, *1*(1), 1.

Wilson, A. C., Roelofs, R., Stern, M., Srebro, N., & Recht, B. (2017). The marginal value of adaptive gradient methods in machine learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 4151–4161.

Yeo, I.-K., & Johnson, R. A. (2000). A new family of power transformations to improve normality or symmetry. *Biometrika*, *87*(4), 954–959.

Zhu, Z. A., & Li, Y. (2016). Even faster SVD decomposition yet without agonizing pain. *CoRR*, *abs/1607.03463*.