Bayesian Hyperparameter Optimization

Joe Whales
Probabilistic Modelling and Reasoning (PMR)

ABSTRACT

Hyperparameters significantly influence the behavior and performance of machine learning models, controlling aspects from model complexity to optimization dynamics. This report examines Bayesian hyperparameter optimization (BHO), a sample-efficient approach to this challenge. We outline the evolution of hyperparameter tuning methods, from grid and random search to probabilistic model-based methods. We discuss the core components of Bayesian optimization frameworks: surrogate models (particularly Gaussian processes and Tree-structured Parzen Estimators), acquisition functions for balancing exploration-exploitation trade-offs and sampling strategies. Through a worked example optimizing a UNet-based image segmentation model, we demonstrate BHO's effectiveness in finding better hyperparameter configurations while requiring 67.6% fewer evaluations and 65.6% less computational time than grid search. Our analysis also reveals that not all hyperparameters contribute equally to performance, supporting the "low effective dimensionality" concept that explains BHO's efficiency. We conclude by discussing future research directions like transfer learning to further improve the efficiency of BHO.

1 INTRODUCTION

Hyperparameters play a critical role in machine learning systems, directly influencing their performance. Unlike model parameters that are learned during training, hyperparameters must be specified before training begins and often require precise tuning to achieve optimal results [7]. As machine learning models become increasingly complex the challenge of hyperparameter selection becomes increasingly important and difficult.

The hyperparameter optimization (HPO) problem can be formally defined as finding the hyperparameter configuration $\lambda^* \in \Lambda$ that minimizes an objective function f:

$$\lambda^* = \arg\min_{\lambda \in \Lambda} f(\lambda)$$

where Λ represents the hyperparameter search space, and f corresponds to a validation loss metric [7].

Several factors make HPO particularly challenging in practice. Function evaluations are typically extremely expensive, especially for large deep learning models or complex machine learning pipelines operating on substantial datasets[7]. The configuration space is often complex being made up of a heterogeneous mix of continuous, categorical and conditional hyperparameters. Determining which hyperparameters to optimize and their appropriate ranges isn't always straightforward. Unlike conventional optimization problems, we generally do not have access to gradients of the objective function with respect to the hyperparameters, and standard optimization assumptions such as convexity and smoothness cannot generally be made [7, 9].

This report explores Bayesian Hyperparameter Optimization (BHO) as a principled approach to hyperparameter tuning within the context of probabilistic modeling and reasoning. We provide background on the history of hyperparameter optimization techniques and show the theoretical foundations of Bayesian optimization.

This report explores Bayesian Hyperparameter Optimization (BHO) as a principled approach to hyperparameter tuning within the context of probabilistic modeling and reasoning. We provide background on the history of hyperparameter optimization techniques. We then examine the core components of BHO: surrogate models that provide probabilistic approximations of objective functions, acquisition functions and sampling strategies for initialization. Finally we demonstrate BHO's practical effectiveness on a worked example on an image segmentation task when compared to grid search. Finally, we discuss limitations of current approaches and promising directions for future research, including transfer learning across optimization tasks.

2 BACKGROUND

Historically, hyperparameter tuning was primarily done through manual search, where hyperparameter values were selected based on intuition, experience, or trial and error [3]. While manual search offers insight into how hyperparameters affect model performance, it suffers from poor reproducibility, requires manual labor and often very time consuming.

Grid search was the first systematic approach, testing hyperparameters at fixed intervals throughout the search space. While simple to implement and easily parallelized, grid search becomes impractical as dimensions increase because the number of required evaluations grows exponentially with each additional hyperparameter [3]. This "curse of dimensionality" severely limits its applicability to problems with more than a few hyperparameters.

Random search, proposed by Bergstra and Bengio [3], samples hyperparameter configurations uniformly from the search space. Random search is able to consistently find better solutions than grid search with fewer evaluations, especially in high-dimensional spaces where only a small fraction of hyperparameters significantly impacts model performance [3]. This efficiency stems from what they termed "low effective dimensionality" - the observation that typically only 1-4 hyperparameters meaningfully influence model performance on a given dataset, with different hyperparameters being important for different datasets [3]. While grid search allocates many trials to dimensions that do not matter while providing inadequate coverage in dimensions that are most important [3].

Sequential optimization methods emerged as an improvement to random search, offering more efficient hyperparameter tuning approaches. Some methods focused on better sampling strategies, such as low-discrepancy sequences [5] and Latin hypercube sampling [17], which provided more uniform coverage of the search space. Others introduced adaptation, including evolutionary strategies [11] and coordinate descent methods, which iteratively refined the search using information from previous evaluations. More advanced approaches like SMAC [14] and TPE [2] used simple surrogate models to approximate the objective function. Although these methods used information from previous trials, they struggled to systematically balance exploration of unknown regions with exploitation of promising areas. This limitation motivated the development of Bayesian optimization [24], which applies Bayesian inference principles to create a principled method for reasoning about the unknown objective function and making better sampling decisions.

BHO provides a systematic framework for efficient hyperparameter tuning through sequential decision-making under uncertainty. Building on early Bayesian optimization techniques from Močkus [19], BHO became prominent in machine learning following implementations by Jones et al. [15] with the Efficient Global Optimization algorithm and further refinements by Snoek et al. [24] in their Spearmint system. BHO works by constructing a probabilistic surrogate model that approximates the relationship between hyperparameters and model performance based on observed evaluations. An acquisition function then uses this model to determine which hyperparameter configuration to evaluate next, balancing exploration of uncertain regions with exploitation of promising areas. This approach allows BHO to find high-performing hyperparameter settings with substantially fewer evaluations than traditional methods like grid or random search. Some key advancements in BHO include techniques for handling heterogeneous hyperparameter spaces (continuous, discrete and categorical) [10] and multi-fidelity evaluations to manage computational costs

3 THEORETICAL BACKGROUND

BHO is made up of three main components: surrogate models to approximate the objective function probabilistically, acquisition functions to guide the selection of the next configuration to evaluate and sampling strategies to initialize and update the process. Together, these components create a system that balances exploration of uncertain regions with exploitation of promising areas in the hyperparameter space.

3.1 Surrogate Models

Surrogate models provide a probabilistic approximation of the objective function. These models allow us to reason about the performance of hyperparameter configurations without training and evaluating our entire model.

Gaussian processes (GPs) are a common choice for surrogate models [24]. They define a distribution over functions through a mean function $\mu(\lambda)$ and covariance kernel k(λ, λ'). For any set of hyperparameter configurations, the corresponding function values follow a multivariate Gaussian distribution:

$$[f(\lambda_1), f(\lambda_2), ..., f(\lambda_n)] \sim \mathcal{N}([\mu(\lambda_1), \mu(\lambda_2), ..., \mu(\lambda_n)], K) \quad (1)$$

where K is the covariance matrix with entries K_{ij} = k(λ_i, λ_j)[20]. When new observations are collected, the posterior distribution over f at a new configuration λ^* can be computed analytically:

$$p(f(\lambda_*)|\mathcal{D}) = \mathcal{N}(\mu_n(\lambda_*), \sigma_n^2(\lambda_*)) \quad (2)$$

This formulation provides both mean predictions and uncertainty estimates, enabling effective exploration-exploitation tradeoffs[24]. The squared exponential kernel is commonly used:

$$k_{SE}(\lambda, \lambda') = \sigma_f^2 \exp\left(-\frac{1}{2l^2}||\lambda - \lambda'||^2\right) \quad (3)$$

While GPs provide solid theoretical foundations, they have practical limitations: they become computationally expensive as the number of data points increases (O(n^3) complexity) and perform poorly with many dimensions or when dealing with different types of hyperparameters [25].

The Tree-structured Parzen Estimator (TPE) [2] is another strong choice of surrogate model. Instead of modeling p(y|λ) directly, TPE models p(λ|y) and p(y) separately. It divides the configurations that it observes into two groups:

$$p(\lambda|y) = \begin{cases} l(\lambda) & \text{if } y < y^* \\ g(\lambda) & \text{if } y \geq y^* \end{cases} \quad (4)$$

where y^* is a threshold performance value, l(λ) is the density of configurations that performed well and g(λ) is the density of configurations that performed poorly. These densities are estimated using kernel density estimation. TPE then selects new configurations by maximizing the improvement ratio l(λ)/g(λ), effectively sampling from places where the configurations are more likely to perform well[2, 4].

TPE offers several advantages for hyperparameter optimization: it can handle mixed parameter types (continuous, discrete and categorical), it scales better with the number of observations and it handles dependencies between parameters well[4]. It is for these reasons that TPE was selected as the surrogate model in our implementation.

Other alternative surrogate models include Random Forests [14], which work well with categorical parameters but struggle when parameter spaces are highly conditional. Bayesian Neural Networks [25] can capture more complex relationships between parameters and performance, but their training process is more computationally expensive and they require more samples to be effective than simpler models [26].

3.2 Acquisition Functions

Acquisition functions use the surrogate model's probabilistic predictions to create a utility function that guides the selection of the next hyperparameter configuration to evaluate. They balance finding configurations that are likely to perform well (exploitation) with trying new areas where performance is uncertain (exploration).

Expected Improvement (EI) is one of the most widely used acquisition functions [15, 19]. EI measures the expected improvement value compared to the best observed performance so far ($f(\lambda^+)$):

$$EI(\lambda) = \mathbb{E}[\max(f(\lambda) - f(\lambda^+), 0)] \quad (5)$$

For a Gaussian process surrogate with posterior mean $\mu(\lambda)$ and posterior variance $\sigma^2(\lambda)$, this has a closed-form expression:

$$EI(\lambda) = \begin{cases} (\mu(\lambda) - f(\lambda^+))\Phi(Z) + \sigma(\lambda)\phi(Z) & \text{if } \sigma(\lambda) > 0 \\ 0 & \text{if } \sigma(\lambda) = 0 \end{cases} \quad (6)$$

where $Z = \frac{\mu(\lambda)-f(\lambda^+)}{\sigma(\lambda)}$, and Φ and ϕ are the cumulative distribution function and probability density function of the standard normal distribution, respectively[15].

Other common acquisition functions include Upper Confidence Bound (UCB), which balances exploration and exploitation through a weighted sum [27], and Probability of Improvement (PI), which evaluates the probability that a new point will outperform the current best observation [16]. More advanced approaches include Entropy Search [12] and Predictive Entropy Search [13], which choose points that provide the most information about where the best possible solution is located. Unlike EI or UCB, these methods directly target uncertainty reduction about the optimum's position rather than just balancing exploration and exploitation.

Finding the next point to evaluate requires solving a smaller optimization problem. We typically use simpler methods like random search or gradient-based approaches to find where the acquisition function reaches its highest value[23].

In TPE, acquisition function optimization works differently. Since TPE models p(λ|y) directly, it approximates the expected improvement by sampling many configurations from l(λ) (the distribution of well performing configurations) and selects the one with the highest value of l(λ)/g(λ) [2, 4]. This naturally handles conditional and non-continuous parameters[6].

3.3 Sampling Strategies

Before the surrogate model can make predictions, it must be initialized with some observations. The initial sampling strategy significantly affects the optimization process's efficiency and outcome.

Random sampling, is a simple baseline approach and can be surprisingly effective in early stages when little is known about the objective function's structure. In some cases, it can even outperform structured approaches during initialization, as demonstrated by Bergstra and Bengio [3]. More structured approaches can provide better coverage of the parameter space. Latin hypercube sampling (LHS) [17] distributes samples more evenly across dimensions. LHS creates uniform intervals over the range of each parameter before randomly sampling one point from each interval. In our case, random sampling is used for both the initial sampling and the subsequent sampling of candidate points to run through the acquisition function.

4 WORKED EXAMPLE

This section shows the practical applicability of BHO by applying it to the task of optimizing parameters for a simple UNet-based segmentation model.

4.1 Methodology

To evaluate the effectiveness of BHO, we chose semantic image segmentation as our test case. This task involves detecting object boundaries in images using a small UNet-based model. The model performance is highly dependent on both model architecture and training parameters, making this a good benchmark for comparing optimization methods.

We built a synthetic dataset for our experiments containing 300 grayscale images of 128x128 pixels, split into 70% training, 15% validation and 15% testing sets. Each image contains a rotated square with noise, blur and random elliptical shapes added to increase the difficulty of the task. Figure 1 shows sample images from the dataset.

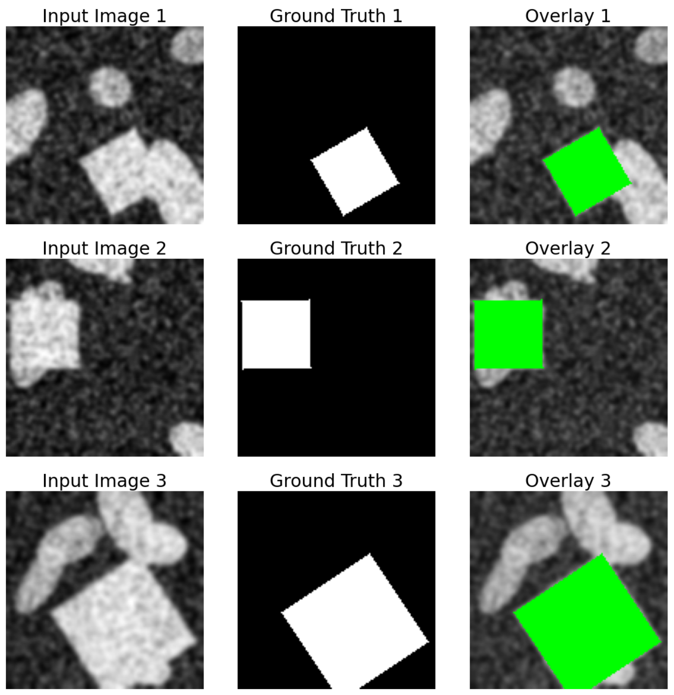

Figure 1: Examples from the synthetic segmentation dataset. Each row shows an input image (left), the corresponding ground truth mask (center) and the ground truth overlaid on the input image (right).

For the segmentation model, we implemented a lightweight U-Net architecture [21]. U-Net has an encoder-decoder structure with skip connections and is widely used for segmentation tasks. Our implementation has several hyperparameters:

- base_filters: Number of filters in the first convolutional layer (range: 8-16)
- depth: Number of downsampling/upsampling levels (range: 2-4)
- learning_rate: Learning rate for the Adam optimizer (range: 0.00005-0.005)
- weight_decay: L2 regularization strength (range: 1e-6-1e-4)
- batch_size: Training batch size (options: 8, 16, 32)

All the models were trained using the Dice loss function [18], which measures the overlap between predicted and ground truth segmentation masks. Each model was trained for a maximum of 50 epochs with early stopping, using a patience of 10, and learning rate reduction on plateau.

For the hyperparameter optimization, two approaches were compared:

1. **Grid Search**: We evaluated all combinations from a predefined set of hyperparameter values:

Table 1: Grid Search Hyperparameter Space

Hyperparameter	Values
base_filters	{8, 16}
depth	{2, 4}
learning_rate	{0.00005, 0.0001, 0.001}
weight_decay	$\{10^{-6}, 10^{-5}, 10^{-4}\}$
batch_size	{8, 16, 32}
Total configurations	2 × 2 × 3 × 3 × 3 = 108

2. **Bayesian Optimization**: We used the TPE algorithm [2] using the Optuna framework [1] for our BHO method. We used the categorical values from the grid search as guide for the hyperparameter space used in our BHO implementation:

Table 2: Bayesian Optimization Hyperparameter Space

Hyperparameter	Range/Values	Type
base_filters	[8, 16]	Integer (rounded)
depth	[2, 4]	Integer (rounded)
learning_rate	[0.00005, 0.005]	Continuous (log scale)
weight_decay	$[10^{-6}, 10^{-4}]$	Continuous (log scale)
batch_size	{8, 16, 32}	Categorical

We evaluated both methods on the Intersection over Union (IoU) score on the test set as well as the computational efficiency of the method.

4.2 Results and Discussion

Our experiments show that HBO outperforms the grid search in terms of both efficiency and performance quality.

BHO found superior hyperparameter configurations while also requiring less computational time. As shown in Figure 2, Bayesian optimization achieved its best IoU score of 0.9112 while requiring only 35 evaluations, compared to grid search's best score of 0.9008 after evaluating 108 configurations. The result is a 1.15% improvement in performance while as well as 67.6% evaluations needed. Moreover, Bayesian optimization completed in 19.07 minutes compared to grid search's 55.40 minutes, a 65.6% reduction in optimization time.

Notably, the best configurations found by each method were significantly different. Grid search selected base_filters=16, depth=4, learning_rate=0.001, weight_decay=1e-6 and batch_size=32. In contrast, Bayesian optimization identified base_filters=14, depth=3, learning_rate=0.00035, weight_decay=3e-6, and batch_size=8. A significant benefit of Bayesian methods becomes apparent here through it's ability to sample anywhere within continuous ranges, and not just at fixed points in a grid.

Figure 3 displays segmentation results from the best model found through Bayesian optimization. The model is generally able to identify the target square but still struggles to accurately outline it in situations where the square is highly occluded.

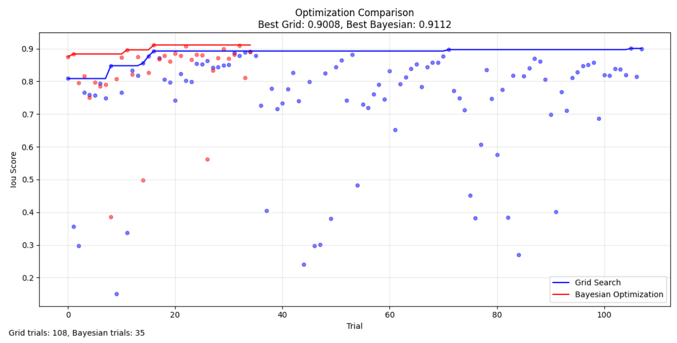

Figure 2: Comparison of optimization trajectories for grid search (blue) and Bayesian optimization (red). The solid lines represent the best score found up to each trial, while the individual points show the performance of each configuration.

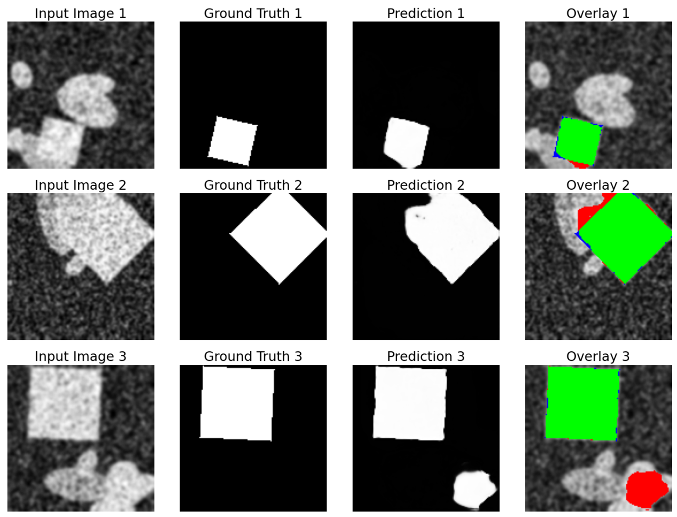

Figure 3: Segmentation performance of the optimal model found through Bayesian optimization. The columns show input images, ground truth masks, model predictions, and color-coded overlays where green areas represent correctly segmented regions, while red areas indicate false positives and blue areas represent false negatives.

Analysis of the optimization results for the BHO optimization revealed that not all hyperparameters contributed equally to the model performance, as shown in Figure 4. Batch size was the most influential parameter with a strong negative correlation to performance, meaning smaller batch sizes resulted in better performance in our segmentation task. Network depth and base filters followed in importance, both showing positive correlations. Learning rate showed the smallest positive effect interestingly. These varying contributions support the "low effective dimensionality" concept described by Bergstra and Bengio [3], which states that typically only a few hyperparameters significantly impact model performance.

This explains why Bayesian optimization performed efficiently: it concentrated sampling in dimensions with the greatest impact on performance while spending fewer resources exploring parameters with less influence on performance.

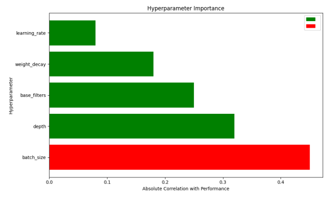

Figure 4: Plot showing the correlation of hyperparameters with the performance of the model.

The code implementation can be found at: code

5 CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

This report demonstrated the effectiveness of Bayesian hyperparameter optimization for machine learning model tuning through a detailed comparison with traditional grid search. Through experimentation we showed that Bayesian optimization finds better-performing settings while using far fewer evaluations, reducing optimization time by 65.6% in our case. This efficiency comes from its ability to focus computing resources on promising areas of the search space using probability-based models. Additionally, Bayesian optimizations ability to explore any point within continuous parameter ranges enables it to find better performing configurations that grid search misses due to its fixed sampling points. These benefits make Bayesian hyperparameter optimization especially useful when computing resources are limited or when rapid model development is needed, addressing both performance and speed requirements.

5.2 Future Work

As machine learning models continue to grow in complexity, Bayesian optimization faces increasing challenges in high-dimensional hyperparameter spaces. The most interesting research direction in my opinion is developing methods that can efficiently traverse continuous parameter spaces would also be highly valuable, as they can discover optimal configurations that coarser approaches would miss entirely. For widely used model architectures that are repeatedly applied to different datasets (such as convolutional networks for computer vision or transformers for NLP), transfer learning could offer significant promise. Methods like hyperparameter priors learned from past tasks [8] and quantile-based transfer learning [22] can improve BHO efficiency by using previously optimized models to guide new ones, reducing the number of evaluations needed for fine-tuning.

REFERENCES

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2623–2631.

[2] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*. 2546–2554.

[3] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *The journal of machine learning research* 13, 1 (2012), 281–305.

[4] James Bergstra, Daniel Yamins, and David D Cox. 2013. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International Conference on Machine Learning*. 115–123.

[5] Paul Bratley, Bennett L Fox, and Harald Niederreiter. 1992. Implementation and tests of low-discrepancy sequences. *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 2, 3 (1992), 195–213.

[6] Stefan Falkner, Aaron Klein, and Frank Hutter. 2018. BOHB: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning*. 1437–1446.

[7] Matthias Feurer and Frank Hutter. 2019. *Hyperparameter optimization*. Springer International Publishing.

[8] Matthias Feurer, Benjamin Letham, Frank Hutter, and Eytan Bakshy. 2018. Practical Transfer Learning for Bayesian Optimization. *arXiv preprint arXiv:1802.02219* (2018). https://arxiv.org/abs/1802.02219

[9] Luca Franceschi, Michele Donini, Valerio Perrone, Aaron Klein, Cédric Archambeau, Matthias Seeger, Massimiliano Pontil, and Paolo Frasconi. 2024. Hyperparameter Optimization in Machine Learning. arXiv:2410.22854 [stat.ML] https://arxiv.org/abs/2410.22854

[10] Eduardo C Garrido-Merchán and Daniel Hernández-Lobato. 2020. Dealing with categorical and integer-valued variables in Bayesian optimization with Gaussian processes. *Neurocomputing* 380 (2020), 20–35.

[11] Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. 2003. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary computation* 11, 1 (2003), 1–18.

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

[13] José Miguel Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. 2014. Predictive entropy search for efficient global optimization of black-box functions. In *Advances in neural information processing systems*. 918–926.

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

[15] Donald R Jones, Matthias Schonlau, and William J Welch. 1998. Efficient global optimization of expensive black-box functions. *Journal of Global optimization* 13, 4 (1998), 455–492.

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

[17] Michael D McKay, Richard J Beckman, and William J Conover. 1979. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics* 21, 2 (1979), 239–245.

[18] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. 2016. V-net: Fully convolutional neural networks for volumetric medical image segmentation. *2016 Fourth International Conference on 3D Vision (3DV)* (2016), 565–571.

[19] Jonas Močkus. 1975. On Bayesian methods for seeking the extremum. (1975), 400–404.

[20] Carl Edward Rasmussen and Christopher K. I. Williams. 2006. *Gaussian processes for machine learning*. MIT Press, Cambridge, MA.

[21] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015* (2015), 234–241.

[22] David Salinas, Huibin Shen, and Valerio Perrone. 2020. A Quantile-based Approach for Hyperparameter Transfer Learning. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119)*, Hal Daumé III and Aarti Singh (Eds.). PMLR, 8438–8448. https://proceedings.mlr.press/v119/salinas20a.html

[23] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. 2015. Taking the human out of the loop: A review of Bayesian optimization. *Proc. IEEE* 104, 1 (2015), 148–175.

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

[25] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. 2015. Scalable Bayesian optimization using deep neural networks. In *International Conference on Machine Learning*. 2171–2180.

[26] Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. 2016. Bayesian optimization with robust Bayesian neural networks. In *Advances in Neural Information Processing Systems*. 4134–4142.

[27] Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. 2009. Gaussian process optimization in the bandit setting: No regret and experimental design. In *International Conference on Machine Learning*. ICML, 1015–1022.

Stellenbosch
UNIVERSITY
IYUNIVESITHI
UNIVERSITEIT

Plagiarism declaration

1. Plagiarism is the use of ideas, material and other intellectual property of another's work and to present is as my own.

2. I agree that plagiarism is a punishable offence because it constitutes theft.

3. I also understand that direct translations are plagiarism.

4. Accordingly all quotations and contributions from any source whatsoever (including the internet) have been cited fully. I understand that the reproduction of text without quotation marks (even when the source is cited) is plagiarism.

5. I declare that the work contained in this assignment is my original work and that I have not previously (in its entirety or in part) submitted it for grading in this module/assignment or another module/assignment.

Name and surname	Joe Whales
Student number	25345125
Module	Probabilistic Modelling and Reasoning 812
Assessment	Project
Date	27 March 2024
Signature	[illegible]