Time-Based Bayesian Optimization for High-Cost Evaluation

Asher Labovich
Brown University
Providence, RI
asher_labovich@brown.edu

Fate Bussey
Brown University
Providence, RI
lafeyette_bussey@brown.edu

Abstract

In cases where derivatives are not easily computed, such as during hyperparameter optimization, traditional gradient-based optimization techniques cannot be used. Derivative-free optimization (DFO) presents unique challenges, particularly when optimizing computationally expensive functions. Traditional bayesian optimization (BO) methods have excelled in creating efficient smooth surrogate models to minimize the number of function evaluations. However, these methods often overlook the critical aspect of evaluation time, focusing solely on reducing the number of iterations. Our work enhances standard BO by integrating a novel acquisition function, the Expected Improvement over Time (EIT). This function not only considers the potential improvement in function values but also incorporates the time dynamics of function evaluations, thus aiming to optimize both the quality of solutions and the time efficiency simultaneously. By constructing dual Gaussian Processes – one for the function values and another for the evaluation times – we propose a mechanism that better utilizes available data, minimizing overall optimization time without compromising on the quality of solutions. This paper demonstrates the theoretical foundation of our approach, provides algorithmic details for its implementation, and compares it to traditional Bayesian Optimization with a focus on high-cost evaluation scenarios such as image recognition.

Keywords: derivative-free optimization, image recognition, bayesian optimization, time efficiency, gaussian processes.

1 Introduction

In many cases, it is essential to find the global optima of a function whose derivatives are either impossible to ascertain or too computationally intensive to compute. This scenario calls for derivative-free optimization (DFO), a method that optimizes functions without relying on their derivatives. Instead of randomly selecting points from the search space, Ω, DFO algorithms use existing function values to intelligently decide the next point for evaluation, as elaborated in [1]. In the case where the function itself is computationally expensive to evaluate, optimization algorithms face an additional challenge: reaching the global optima in the minimum amount of time. Bayesian optimization (BO), initially discussed in [2], is the most commonly-used algorithm when the target function is computationally expensive. BO begins by constructing a smooth surrogate function, such as a Gaussian Process (discussed in [3]), which is computationally simpler to evaluate and thus easier to optimize. The Gaussian Process, which models the function as $\mathcal{N}(\mu(x), \sigma(x))$ for every $x \in \Omega$, is found by fitting a series of points $(x_i, y_i)_1^n$ and is calculated via a specific kernel function chosen. The Radial Basis Function, $K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma||\mathbf{x} - \mathbf{x}'||^2)$ is commonly used as a kernel function, and is governed by the hyperparameter γ. After creating the Gaussian process surrogate function, BO maximizes an acquisition function – for example, the expected improvement function (detailed in

Section 2). It then probes the actual function at that maximizing point and continues the process again until a set number of evaluations, often pre-determined or after a set number of evaluations without progress made. We notice that existing BO algorithms have two crucial areas for improvement:

1. Existing algorithms do not make use of all their available information. Specifically, they only use the values received from the target function, and not how long it takes to evaluate them.
2. Existing algorithms minimize the number of iterations taken to optimize the target function, rather than minimizing the time taken to do so.

We solve this problem by adding an additional acquisition function, “Expected Improvement over Time", to the existing “Bayesian Optimization" (described in [4]) package. We create an additional Gaussian process over the function evaluation times, and choose points that we expect to have a low cost-to-evaluate with high expected improvement.

2 Existing Acquisition Function

One of the most commonly used BO acquisition functions is expected improvement (EI), which simultaneously keeps track of the probability of improvement and how much the function value would improve *by*. At each point x, the Gaussian process for function values produces a normal distribution over all possible loss values y.

Definition 1 (Expected Improvement)**.** *The expected improvement* $\mathbb{E}_y[I(x,y)]$ *is defined as:*

$$\int_{-\infty}^{\infty} \frac{\max(y_{\min} - y, 0)}{\sqrt{2\pi\sigma(x)^2}} \exp\left(-\frac{(y-\mu(x))}{2\sigma(x)^2}\right) dy$$

and simplifies to

$$(\mu(x) - y_{\min} - \xi)\Phi\left(\frac{\mu(x) - y_{\min} - \xi}{\sigma(x)}\right) + \sigma(x)\phi\left(\frac{\mu(x) - y_{max} - \xi}{\sigma(x)}\right) \tag{1}$$

as shown in [5].

The constants $\mu(x)$ and $\sigma(x)$ are found via the Gaussian process. More detailed descriptions of the math behind Gaussian processes as surrogate functions can be found in [6]. ξ represents a hyperparameter trading off between exploration (choosing unknown points with high variation) and exploitation (choosing well-known points with good results). This function is usually considerably easier to evaluate than the target function, and it can be parallelized with relative ease. For target functions that take less than a second to evaluate, there are likely better methods than BO.

3 Advancement on Acquisition Function

While we do not change the surrogate function from Gaussian processes, we do use it in an additional way. Specifically, we create two Gaussian processes: one for the target function values themselves, and one for the time taken to evaluate the function at each input value. After doing so, we have means $\mu_y(x), \mu_t(x)$ and standard deviations $\sigma_y(x), \sigma_t(x)$ at each value x in the search space. Here, the subscript y refers to statistical measures for the function itself, versus t which refers to time. With these values defined, we create a new acquisition function: **expected improvement over time**, or **EIT**.

In our derivation, we make **one key assumption**: after accounting for known time and function values, the values at other points are independent. Specifically, while two *Gaussian processes* may exhibit correlation – such as a scenario where a lower expected time correlates with a lower expected function value – the knowledge that the time value deviates by N standard deviations from the mean provides no information about the deviation of the function value from its expected level. This assumption is not particularly strong: we have no particular reason to believe that the deviations in time and function values are dependent after accounting for their expected values. This assumption of conditional independence simplifies our calculations by allowing us to separate expected values.

However, this assumption may fail in scenarios where there is an intrinsic link between the complexity of the evaluation and the function value. For example, in a hyperparameter optimization task for a machine learning model, evaluating configurations with high complexity (e.g., deep neural networks with many layers) may inherently take longer and produce different performance metrics compared to simpler configurations. In such cases, the deviations in evaluation time and function value may be correlated even after accounting for their expected values.

Definition 2 (EIT from 0)**.** *Given these values, the EIT is initially defined and simplified as the following:*

$$\mathbb{E}_{y_1,y_2}\left[\frac{I(x,y_1)}{T(x,y_2)}\right] = \mathbb{E}_y[I(x,y)]\mathbb{E}_y\left[\frac{1}{T(x,y)}\right] \tag{2}$$

$$= \left[(\mu_y(x) - y_{\min})\Phi\left(\frac{\mu_y(x) - y_{\min}}{\sigma_y(x)}\right) + \sigma_y(x)\phi\left(\frac{\mu_y(x) - y_{max}}{\sigma_y(x)}\right)\right] \times \int_0^\infty \frac{1}{y\sqrt{2\sigma_t(x,y)^2\pi}}\exp\left(-\frac{(y-\mu_t(x,y))^2}{2\sigma_t(x,y)^2}\right)dy \tag{3}$$

The integral given here is **not** the true integral for EIT, as it **diverges**. The following proof details why:

Divergence of EIT from 0. Since

$$\frac{1}{\sqrt{2\sigma_t(x,y)^2\pi}}\exp\left(-\frac{(y-\mu_t(x,y))^2}{2\sigma_t(x,y)^2}\right)$$

is the pdf of a normal distribution, it is continuous and greater than zero on the entire real line. Therefore, by the Extreme Value Theorem (discussed in [7]),

$$\exists \epsilon > 0 \; s.t. \forall y \in [0,1], \; \frac{1}{\sqrt{2\sigma_t(x,y)^2\pi}}\exp\left(-\frac{(y-\mu_t(x,y))^2}{2\sigma_t(x,y)^2}\right) \le \epsilon$$

In addition, $\frac{1}{y} > 0 \; \forall y > 0$. Therefore,

$$\forall x > 0 \;\; \frac{1}{y\sqrt{2\sigma_t(x,y)^2\pi}}\exp\left(-\frac{(y-\mu_t(x,y))^2}{2\sigma_t(x,y)^2}\right) \ge 0$$

and thus,

$$\begin{aligned}
\int_0^\infty \frac{1}{y\sqrt{2\sigma_t(x,y)^2\pi}}\exp\left(-\frac{(y-\mu_t(x,y))^2}{2\sigma_t(x,y)^2}\right)dy &= \int_0^1 \frac{1}{y\sqrt{2\sigma_t(x,y)^2\pi}}\exp\left(-\frac{(y-\mu_t(x,y))^2}{2\sigma_t(x,y)^2}\right)dy \\
&\quad + \int_1^\infty \frac{1}{y\sqrt{2\sigma_t(x,y)^2\pi}}\exp\left(-\frac{(y-\mu_t(x,y))^2}{2\sigma_t(x,y)^2}\right)dy \\
&\ge \int_0^1 \frac{1}{y\sqrt{2\sigma_t(x,y)^2\pi}}\exp\left(-\frac{(y-\mu_t(x,y))^2}{2\sigma_t(x,y)^2}\right)dy \\
&\quad + \int_1^\infty 0\,dy \\
&= \int_0^1 \frac{1}{y\sqrt{2\sigma_t(x,y)^2\pi}}\exp\left(-\frac{(y-\mu_t(x,y))^2}{2\sigma_t(x,y)^2}\right)dy \\
&\ge \int_0^1 \frac{\epsilon}{y}\,dy \\
&= \epsilon \ln y \Big|_0^1 \\
&= \infty
\end{aligned}$$

□

Even though the original integral diverges, it also assumes that it is possible for time to be 0. Obviously, no event can take 0 time, and EIT is especially useful for evaluations that take significant time. So, we can define the true EIT from a minimum value $t^{\min}$.

Definition 3 (EIT). *The true formula for EIT is the following:*

$$\mathbb{E}_{y_1,y_2}\left[\frac{I(x,y_1)}{T(x,y_2)}\right] = \left[(\mu_y(x) - y_{\min})\Phi\left(\frac{\mu_y(x) - y_{\min}}{\sigma_y(x)}\right) + \sigma_y(x)\phi\left(\frac{\mu_y(x) - y_{max}}{\sigma_y(x)}\right)\right] \times \int_{t^{\min}}^{\infty} \frac{1}{y\sqrt{2\sigma_t(x,y)^2\pi}} \exp\left(-\frac{(y-\mu_t(x,y))^2}{2\sigma_t(x,y)^2}\right) dy \quad (4)$$

We define $t^{\min} = \frac{1}{10}\min t$, where $\min t$ is the smallest time value found so far. Any $t^{\min}$ $s.t.$ $0 < t^{\min} < \min t$ functions in a similar manner.

Convergence of EIT. Since $\frac{1}{y}$ is a monotonically decreasing function on $(0,\infty)$, it has a maximum on $[t^{\min},\infty)$ of $\frac{1}{t^{\min}}$. In addition, the normal distribution's pdf is always ≤ 1 over any integral on the real line. Therefore,

$$\int_{t^{\min}}^{\infty} \frac{1}{y\sqrt{2\sigma_t(x,y)^2\pi}} \exp\left(-\frac{(y-\mu_t(x,y))^2}{2\sigma_t(x,y)^2}\right) dy \leq \int_{t^{\min}}^{\infty} \frac{1}{t^{\min}\sqrt{2\sigma_t(x,y)^2\pi}} \exp\left(-\frac{(y-\mu_t(x,y))^2}{2\sigma_t(x,y)^2}\right) dy < \frac{1}{t^{\min}}$$

Since the integrand is always positive, it converges to a value in $(0, \frac{1}{t^{\min}})$ $\square$

Though the integral converges, there is no currently-known easily-expressed simplification, so it must be numerically calculated. Maximizing this integral by individual numerical calculations is considerably more time-intensive than maximizing EI. We derive the EIT value via SciPy's quad integration algorithm, described in [8]. As described in Section 6, this algorithm does not lend itself well to parallelization. We did, however, find in Figure 1 that EI only outperformed EIT by a statistically insignificant 0.2 seconds on average. Even if this overperformance was statistically significant, it would still be negligent on the scale of ML algorithms like image recognition. Algorithm 1 describes an alternative algorithm more amenable to parallelization, though with a greater error of integration depending on the scale of A and N. It utilizes rectangular Riemannian integration over a bounded interval, which reduces computation without substantially changing the true value, considering that both $\frac{1}{y}$ and $f_{N(\mu,\sigma)}(x)$ drop quite quickly after 0. We do not bound the errors on this algorithm, nor prove its convergence, since we do not utilize it in our research. It is merely a cause for future research into EIT.

Each x_i is chosen randomly from the search space. As $A, N \to \infty$, this procedure becomes the same as taking the integral NM times.

4 Application

As mentioned in the introduction, we implemented EIT by expanding the Bayesian Optimization package in a github repository found here. A user can now elect to use "eit" in addition to the existing acquisition functions (referred to in the package as utility functions). We used the package's existing Gaussian process implementation to track and predict the time required for function evaluations. To evaluate the integral for EIT, we used SciPy's quad() implementation (in the absence of an analytic derivation), described in [8]. While this gives a highly accurate estimate of the integral's value, it cannot be easily vectorized: the implications of this performance bottleneck are discussed below. We implemented a CNN image recognition model (initially described in [9]) to test the performance of our EIT implementation. This model was based on PyTorch's CNN Module (discussed in [10]) and consisted of both Convolutional and feed-forward layers. The model was tested and trained on the cifar-10 dataset (classifying images belonging to 10 categories). The cifar-10 dataset can be

Algorithm 1 Computation of EIT Integral

```
1:  procedure ArgMaxEIT($A, N, t^{\min}, x_1, ..., x_M$)
2:      $t^{\max} \leftarrow \max(\mu_t(x_i) + A\sigma_t(x_i))$
3:      EIT$_{best}$ = 0
4:      $x_{best}$ = None
5:      for $k \leftarrow 1$ to M do
6:          EIT $\leftarrow 0$
7:          for $i \leftarrow 1$ to N do
8:              $y_i \leftarrow \frac{(N-i)t^{\min}+it^{\max}}{N}$
9:              EIT $\leftarrow$ EIT + $\frac{1}{y_i\sqrt{2\sigma_t(x_k,y_i)^2\pi}} \exp\left(-\frac{(y_i-\mu_t(x_k,y_i))^2}{2\sigma_t(x_k,y_i)^2}\right)$
10:         end for
11:         EIT $\leftarrow$ EIT * $\left((\mu(x) - y_{\min} - \xi)\Phi\left(\frac{\mu(x)-y_{\min}-\xi}{\sigma(x)}\right) + \sigma(x)\phi\left(\frac{\mu(x)-y_{\max}-\xi}{\sigma(x)}\right)\right)$
12:         if EIT>EIT$_{best}$ then
13:             EIT$_{best} \leftarrow$ EIT
14:             $x_{best} \leftarrow x_k$
15:         end if
16:     end for
17:     return $x_{\text{best}}$
18: end procedure
```

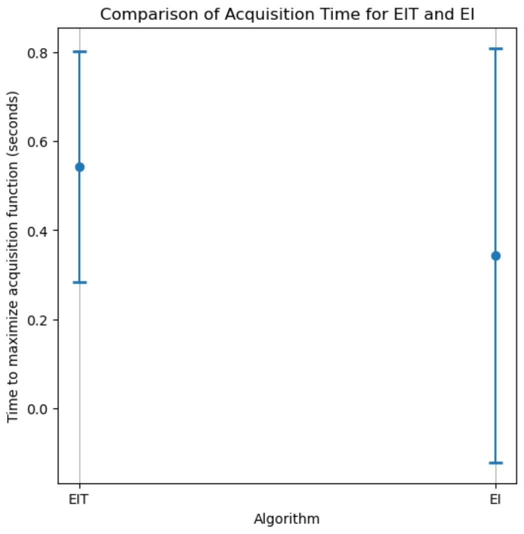

Figure 1: Average time to maximize each acquisition function

found at [11]. We used the PyTorch implementations of cross entropy loss and the Adam optimizer. Performance was measured as the percentage of images classified correctly (top prediction only). BO was used to optimize the following hyperparameters within the respective domains:

1. number of training epochs, $[1, 5]$,
2. batch size, $[2, 20]$,
3. learning rate, $[10^{-3.5}, 10^{-1.5}]$,
4. number of hidden feed-forward layers, $[1, 3]$,
5. size of the hidden feed-forward layers, $[32, 300]$,
6. and kernel size for the first convolution layer, $[2, 7]$.

Since the output of the acquisition function is a point in continuous euclidean space, we rounded the appropriate dimensions for discrete-valued hyper parameters – this being the recommended practice from the makers of the BO package.

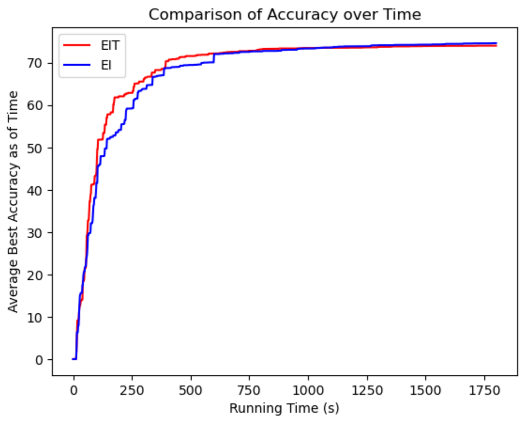

Figure 2: Comparison of average maximum accuracy over time

5 Results

We ran 40 trials comparing EIT to EI specifically, as it is the most ubiquitous and endorsed acquisition function used currently. Both acquisition functions were given equal time (30 minutes) to optimize the CNN. All algorithms were run on a high-performance computing system with 1 GPU and 2 CPUs. Figure 2 shows the how the the accuracy of the best found hyper parameter configuration improved over time. While EIT and EI both eventually reach the maximum found value (of about 74%), EIT generally reaches more accurate values slightly faster. Figure 3 shows the average time to reach the maximum value, as well as error bounds. These error bounds reflect the uncertainty in both initial point selection and the CNN algorithm itself. Our error bars utilize the Central Limit Theorem (discussed in [12], whose assumptions are validated since each sample is independent from all others, and our sample size is 40. Our error bars are calculated by

$$\overline{X} \pm 1.96\frac{S_N}{\sqrt{N}}$$

Where $\overline{X}$ refers to the sample mean and S_N refers to the sample standard deviation. Figure 4 compares EIT and EI's speed to reach various accuracy benchmarks of 70%, 72%, and 74%. We conduct a series of t-tests to compare the effectiveness of EIT to EI on various bechmarks.

We conducted three t-tests on each benchmark and found p-values of 0.13, 0.15, and 0.33, respectively. Consequently, we cannot reject the null hypothesis that, given a specific benchmark, EIT and EI achieve it in the same time. However, when comparing the time to reach the within-run **maximum** accuracy, we observed a p-value of 0.005, significant at the 0.01 level. This indicates that, within a given run, EIT reaches its maximum value faster than EI, even though it may not consistently reach specific benchmarks faster in general. All of our statistical analyses are based on a limit number of trials, and larger-scale studies are necessary to generalize these findings. Despite this, all three benchmarks exhibit a consistent trend where EIT, on average, reaches the benchmarks faster than EI. While EIT showed some promising results in reducing optimization time for CIFAR-10 image recognition tasks, it remains to be seen how it performs in other domains such as natural language processing or reinforcement learning.

6 Discussion

Overall, our implementation suggests EIT warrants further examination as a viable acquisition function for bayesian optimization, specifically in the context of hyper parameter optimization. Within our experimental design, several things stand out as areas with room for improvement. For one, we numerically approximate the integration required for EIT in a way that does not support vectorized operations. While this only minimally impacted our results, as there was no statistically significant difference in time to maximize the two acquisition functions, the discrepancy did dramatically affect performance on simpler functions used in our preliminary testing (the Ackley function, for instance).

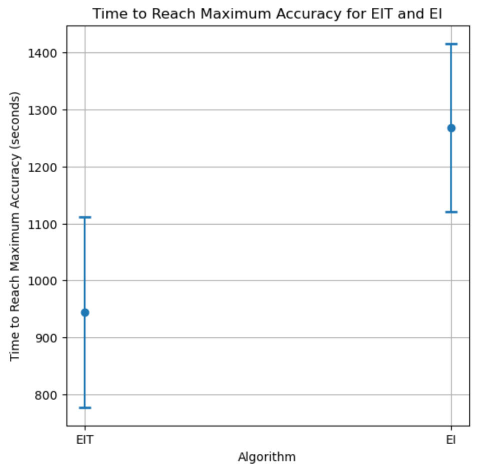

Figure 3: Comparison of time to reach maximum accuracy within each 30-minute run

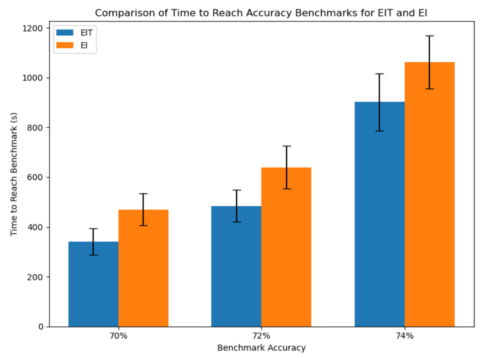

Figure 4: Comparison of time to reach various bechmarks

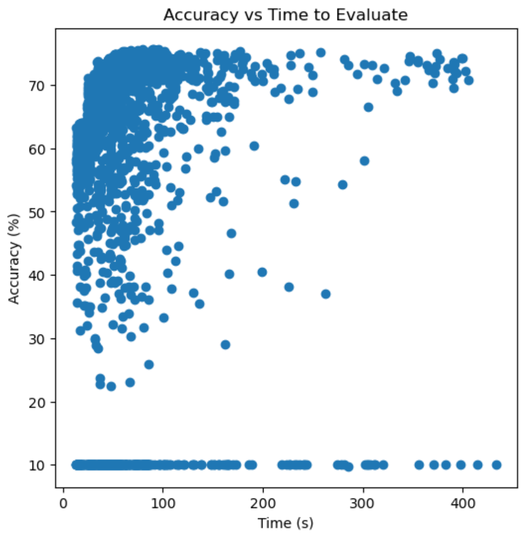

Figure 5: Accuracy vs. time to evaluate function, over all trials

Moreover, our implementation would benefit from a larger domain to sample from. We deliberately capped several hyperparameters below the upper limit that could potentially improve performance to allow for more full trials. This limitation means our findings may not fully represent the potential performance gains of EIT across a broader hyperparameter space. In particular, the capping of hyperparameters caused one significant issue: nearly all points in the search space required a similar amount of evaluation time. Figure 5 illustrates this, showing that the vast majority of points took between 25 and 100 seconds to evaluate. When all points have similar evaluation times, the time-based Gaussian process in EIT becomes relatively flat. Consequently, the EIT formula converges to the EI formula, causing both EIT and EI to test similar points. Uncapping the hyperparameter search space, perhaps by increasing the maximum number of epochs, or selecting machine learning problems with even larger evaluation times would likely address this issue and provide a clearer demonstration of the advantages of EIT.

Although the range of performances observed in our trials is sufficient to demonstrate some potential merits of EIT, no configuration sampled by either acquisition function reaches the performance level of current top-performing models on CIFAR-10. This limitation is a known characteristic of PyTorch

CNN dummy architectures. Nevertheless, we opted to use this module and dataset due to their familiarity and convenience.

However, expanding the domain of the hyperparameters optimized in our experiments is only part of the challenge. A significant advantage of EIT, and Bayesian optimization for hyperparameters in general, is the potential to delegate nearly every arbitrary aspect of model architecture to the optimization process. For instance, while our model selected a static learning rate, recent findings indicate the benefits of a dynamic learning rate that changes non-linearly throughout the training process. This dynamic schedule could be adjusted based on epochs, test accuracy, or a combination of both, with the coefficients of this function subject to optimization. Within the scope of CIFAR-10 image recognition, there are numerous other attributes that were not optimized in our trials, such as the number of output channels, ReLU threshold, stride, and padding for filters. We excluded certain parameters, like the number of convolution layers and unique kernel sizes for the remaining convolution layers, as they proved to be ineffective within the allowed domain.

Further research aimed at enhancing our implementation and continually increasing the predictive power, scope, and scale of the models being optimized will be essential to determine the precise optimal applications of EIT. While EIT showed promising results in reducing optimization time for CIFAR-10 image recognition tasks, it remains to be seen how it performs in other domains such as natural language processing or reinforcement learning. Our preliminary investigation indicates both a strong theoretical foundation and promising initial data-based evidence for the future utilization of EIT in scenarios involving high-cost evaluations.

References

[1] Jeffrey Larson, Matt Menickelly, and Stefan M. Wild. Derivative-free optimization methods. *Acta Numerica*, 28:287–404, may 2019.

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

[3] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2005.

[4] Fernando Nogueira. Bayesian Optimization: Open source constrained global optimization tool for Python, 2014–.

[5] Eric Brochu, Vlad M. Cora, and Nando de Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning, 2010.

[6] Jie Wang. An intuitive tutorial to gaussian process regression, 2023.

[7] Walter Rudin. *Principles of Mathematical Analysis*. McGraw-Hill, 1976.

[8] Robert Piessens, Elise Doncker-Kapenga, Christoph W. Überhuber, and David K. Kahaner. *Quadpack*. Springer, 1983.

[9] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[10] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[11] Alex Krizhevsky. Learning multiple layers of features from tiny images, 2009.

[12] Joseph K. Blitzstein and Jessica Hwang. *Introduction to Probability, Second Edition*. Taylor and Francis Group, LLC, 2019.