# Time-Based Bayesian Optimization for High-Cost Evaluation

Asher Labovich[1] and Lafayette Bussey[2]

[1]Brown University, Providence-USA
[2]Brown University, Providence-USA

May 14, 2024

**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 introduces BayesOptTime, an innovative approach that 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 discusses its advantages over conventional methods 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, $\Omega$, 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 algrithms face an additional challenge: reaching the global optima in the minimum amount of time. Bayesian optimization (BO) 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, 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 $\gamma$. 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.

Our algorithm, BayesOptTime, solves these problems by expanding on the existing package "Bayesian Optimization" (described in [2]) by adding an additional acquisition function, "Expected Improvement over Time". 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 BO Algorithms

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 2.1** (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 [3].

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 [4]. $\xi$ 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, and most definitely better methods than BayesOptTime.

# 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 of conditional independence simplifies our calculations by allowing us to separate expected values.

**Definition 3.1** (EIT from 0)**.** Given these values, the EIT is initially defined and simplified as the following:

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

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

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 greater than zero on the entire real line. Therefore, $\exists \epsilon > 0 \; s.t. \forall y \in [0,1]$, $\frac{1}{\sqrt{2\sigma_t(x,y)^2\pi}} \exp\left(-\frac{(y-\mu_t(x,y))^2}{2\sigma_t(x,y)^2}\right) \leq \epsilon$. In addition, $\frac{1}{y} > 0$ and $\frac{1}{\sqrt{2\pi\sigma_t(x,y)^2}} \exp\left(-\frac{(y-\mu_t(x,y))^2}{2\sigma_t(x,y)^2}\right) > 0$. Therefore,

$$\int_0^\infty \frac{1}{y\sqrt{2\sigma_t(x,y)^2\pi}} \exp\left(-\frac{(y-\mu_t(x,y))^2}{2\sigma_t(x,y)^2}\right) dy \geq$$
$$\int_0^1 \frac{1}{y\sqrt{2\sigma_t(x,y)^2\pi}} \exp\left(-\frac{(y-\mu_t(x,y))^2}{2\sigma_t(x,y)^2}\right) dy \geq$$
$$\int_0^1 \frac{\epsilon}{y} dy =$$
$$\epsilon \ln y|_0^1 = \infty$$

□

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 3.2** (EIT)**.** The true formula for EIT is the following

$$\mathbb{E}_{y_1,y_2}\left[\frac{I(x,y_1)}{T(x,y_2)}\right] = \left[(\mu_y(x)-y_{\min})\Phi\left(\frac{\mu_y(x)-y_{\min}}{\sigma_y(x)}\right)+\sigma_y(x)\phi\left(\frac{\mu_y(x)-y_{\max}}{\sigma_y(x)}\right)\right] \times \int_{t^{\min}}^{\infty}\frac{1}{y\sqrt{2\sigma_t(x,y)^2\pi}}\exp\left(-\frac{(y-\mu_t(x,y))^2}{2\sigma_t(x,y)^2}\right)dy \tag{3}$$

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

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

Since the integrand is always positive, it converges. □

Though the integral converges, there is no currently-known nicely-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. 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 about 0.3 seconds on average, which is fairly negligent on the scale of image recognition and other machine learning models. 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 retangular 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, 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. 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). While this gives a highly accurate estimate of the integral's value, it cannot be vectorized: the implications of this performance bottleneck are discussed below.

We implemented a CNN image recognition model to test the performance of our EIT implementation on.

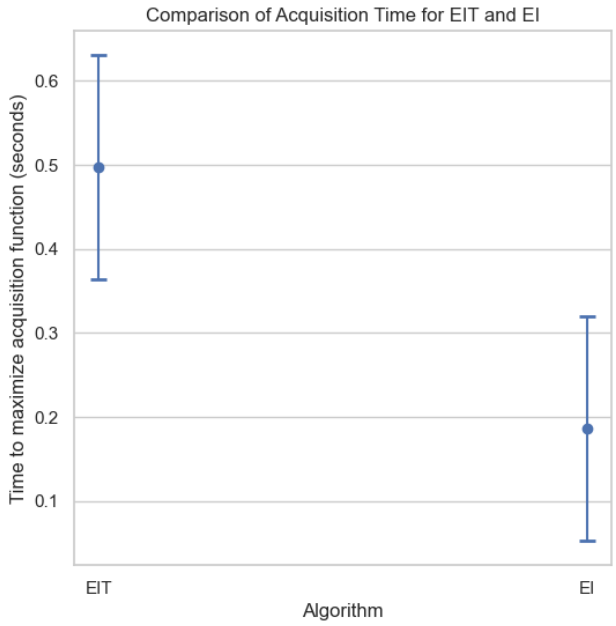


Fig. 1: Average time to maximize each acquisition function

This Model was based on PyTorch's CNN Module 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). It 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).

Bayesian optimization was used to optimize the following hyper parameters 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 bayesian optimization package.

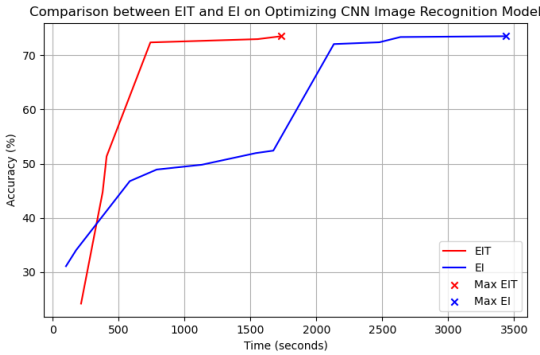


Fig. 2: Comparison of average maximum accuracy over time

# 5 Results

We ran 4 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 (ranging from 1-2 hours) to optimize the CNN. Figure 2 shows the how the the accuracy of the best found hyper parameter configuration improved over time. Figure 3 shows the average time to reach the maximum value, as well as error bounds. A t-test for the average time to reach the maximum value gave a p-value of 0.13, so we cannot reject the null hypothesis that EIT reaches the maximum no faster than EI. However, with only 4 trials, we expected a lack of statistical significance, and expect to gain more concrete confirmation as to the efficiency of EIT as we test each algorithm more times.

Since an acquisition function will not always return a point that improves upon the best known configuration, a graph showing the performance of all suggested points over time is not useful in comparing performance. It would, however, betray EIT's unique approach. EI's suggestions are better than EIT's on average (relative to the best known configuration at any given time), but EIT suggests points far more erratically. While many such recommendations are often worse, causing the model to do no better than guessing randomly, they are very easy to evaluate. Consequently, EIT is able to suggest and evaluate as many configurations as EI in the same time. Figure 4 shows the average number of iterations that EIT and EI test within one hour each.

# 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. Our experimental design, though limited in scope, saw EIT achieve comparable performance to EI in optimizing a CNN architecture while sampling configurations in unique way.

EIT's propensity to sample cheaper-to-evaluate, and thus more, configurations falls in line with what we

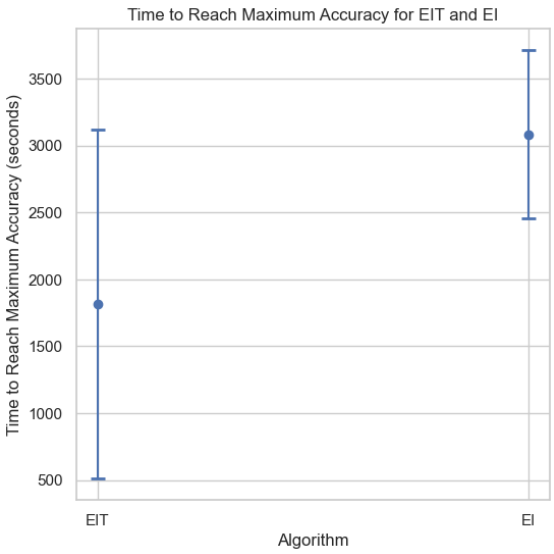


Fig. 3: Comparison of average maximum accuracy over time

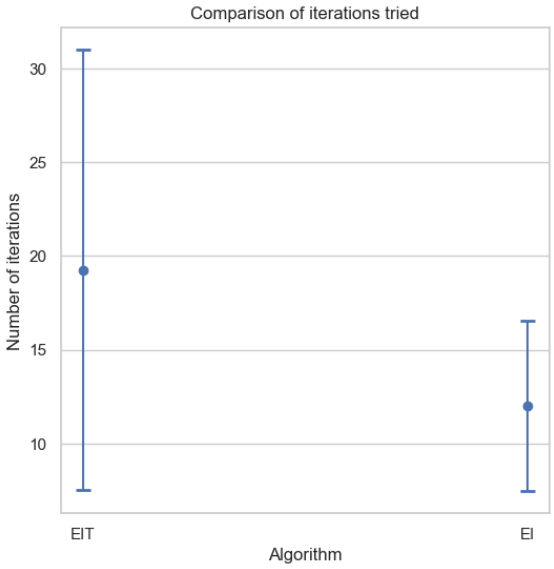


Fig. 4: Comparison of iterations tried by EIT and EI

predicted. This results in a better informed Gaussian process, and thus a more thoroughly explored domain earlier on in optimization. Conversely, EIT did not leverage this improvement to find better configurations than EI regardless of time.

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 it takes at most a second more to evaluate EIT, the discrepancy did dramatically affect performance on simpler functions used in our preliminary testing (the Ackley function, for instance).

Moreover, our implementation would benefit from a larger domain to sample from. We ultimately capped several hyper parameters below the upper limit of what could be expected to improve performance in service of having more full trials, though still many more trials would be desirable. Though the range of performances observed in our trials is still enough to demonstrate some merits of EIT, no configuration sampled by either acquisition function is in the same stratosphere as current top-performing models on cifar-10. While this is a known aspect of PyTorch CNN dummy-architectures, we still elected to use the module and dataset on account of their familiarity and convenience.

Expanding the domain of the hyper parameters we optimized as part of the experiment is only half the battle, however. A major upshot of EIT succeeding, and indeed using bayesian optimization for hyper parameters as a whole, is that one could delegate nearly every arbitrary aspect of model architecture to it. For instance: while we had the model select a static learning rate, recent results demonstrate the virtue in a

dynamic one that changes non-lenearly over the course of training. This varying schedule could be tuned by varying learning rate as a function of epochs, test accuracy, or some combination of the two. The coefficients of this function could then be optimized. Just within the scope of cifar-10 image recognition, there are many more qualities that we did not optimize in our trials, such as the number of output channels, Relu threshold, stride and padding for filters, and more (we removed two qualities from those being optimized as they proved to be red herrings within the allowed domain: number of convolution layers and unique kernel sizes for the remaining convolution layers).

Additional research that improves on our implementation and continues to increase the predictive power, scope, and scale of the models being optimized will be necessary to determine the precise strengths and weaknesses of EIT, and furthermore, if it can contend as a useful acquisition function for Bayesian optimization in research or commercial contexts. Our preliminary inquiry by no means rules this out. Within our scope, EIT balances exploration and exploitation differently than the status quo without major sacrifices in performance. This contrast begs the question: perhaps the next breakthrough in bayesian optimization will synthesize elements of both.

## Acknowledgments

The authors thank Professor Yu Cheng for his support. Neither author reports a conflict of interest.

## References

[1] J. Larson, M. Menickelly, and S. M. Wild, "Derivative-free optimization methods," *Acta Numerica*, vol. 28, p. 287â404, May 2019.

[2] F. Nogueira, "Bayesian Optimization: Open source constrained global optimization tool for Python," 2014–.

[3] E. Brochu, V. M. Cora, and N. de Freitas, "A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," 2010.

[4] J. Wang, "An intuitive tutorial to gaussian process regression," 2023.

## Additional Figures

**Algorithm 1** Computation of EIT Integral

**procedure** ARGMAXEIT($A, N, t^{\min}, x_1, ..., x_M$)
  $t^{\max} \leftarrow \max(\mu_t(x_i) + A\sigma_t(x_i))$
  $\text{EIT}_{best} = 0$
  $x_{best} = \text{None}$
  **for** $k \leftarrow 1$ **to** M **do**
    EIT $\leftarrow 0$
    **for** $i \leftarrow 1$ **to** N **do**
      $y_i \leftarrow \frac{(N-i)t^{\min}+it^{\max}}{N}$
      EIT $\leftarrow$ EIT $+ \frac{1}{\sqrt{2\sigma_t(x_k,y_i)^2\pi}} \exp\left(-\frac{(y_i-\mu_t(x_k,y_i))^2}{2\sigma_t(x_k,y_i)^2}\right)$
    **end for**
    EIT $\leftarrow$ EIT * $\left((\mu(x) - y_{\min} - \xi)\Phi\left(\frac{\mu(x)-y_{\min}-\xi}{\sigma(x)}\right) + \sigma(x)\phi\left(\frac{\mu(x)-y_{\max}-\xi}{\sigma(x)}\right)\right)$
    **if** EIT>$\text{EIT}_{best}$ **then**
      $\text{EIT}_{best} \leftarrow$ EIT
      $x_{best} \leftarrow x_k$
    **end if**
  **end for**
  **return** $x_{\text{best}}$
**end procedure**