Optimizing Neural Architecture Search via Bayesian Optimization using Upper Confidence Bound

PENG LIU, Singapore Management University, Singapore

Neural Architecture Search (NAS) is a time-consuming task that could potentially benefit from a probabilistic optimization approach such as Bayesian Optimization (BO). While the utility of BO in NAS is increasingly acknowledged, the focus has largely been on popular acquisition functions such as Expected Improvement. This study extends to the Upper Confidence Bound (UCB) as the acquisition function for NAS within the Bayesian framework. With its inherent balance between exploration and exploitation, UCB offers a robust and efficient methodology for navigating the complex search space of neural architectures. We demonstrate that our approach achieves superior performance through empirical validation on benchmark datasets.

Additional Key Words and Phrases: Neural Architecture Search, Bayesian Optimization, Upper Confidence Bound

ACM Reference Format:

Peng Liu. 2023. Optimizing Neural Architecture Search via Bayesian Optimization using Upper Confidence Bound. 1, 1 (October 2023), 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Neural Architecture Search (NAS) has emerged as a promising avenue for automating the design of high-performance neural network architectures tailored to specific tasks and datasets [2, 8]. While recent advancements in NAS have achieved state-of-the-art results that surpass human-designed architectures, the process remains computationally expensive, often requiring enormous computational resources and time. This limitation has spurred a quest for more efficient optimization techniques that systematically and effectively explore NAS's complex and high-dimensional search space.

Bayesian Optimization (BO) has garnered attention as a robust technique for optimizing expensive black-box functions, and its application to NAS has shown promising results. However, most existing works in this domain have employed popular acquisition functions like Expected Improvement (EI), such as the recent work by [3], thereby leaving other potentially useful acquisition functions underexplored.

In this study, we explore the use of the Bayesian Optimization framework to conduct NAS more efficiently. We integrate the Weisfeiler-Lehman (WL) subtree graph kernel into the Gaussian Process (GP) model used as a surrogate function in BO. This allows us to capture complex relationships in the search space more effectively. Furthermore, we extend the BO methodology by incorporating the Upper Confidence Bound (UCB) as the acquisition function. UCB is known for its balancing act between exploration and exploitation, and we posit that its properties are particularly well suited for the complex and uncertain search space encountered in NAS.

Our contributions are twofold: 1. We are the first to explore the effectiveness of the UCB acquisition function within the BO framework for NAS, thereby providing a new perspective on its utility. 2. We demonstrate that our method

Author's address: Peng Liu, liupeng@smu.edu.sg, Singapore Management University, 81 Victoria St, Singapore, 188065.

Manuscript submitted to ACM

performs better than existing NAS techniques that use other acquisition functions based on the Diabetes and Wine Quality datasets,

2 METHODOLOGY

2.1 Bayesian Optimization

Bayesian Optimization (BO) offers a structured approach for globally optimizing complex, non-convex, and computationally expensive black-box functions [5]. It employs a balance between exploration and exploitation, making it particularly suitable for optimizing NAS. Given an unknown objective function $f(x)$, which quantifies the validation error, and x representing the graphical architecture of the neural network, BO aims to find $x^* = \arg\min_{x \in \mathcal{X}} f(x)$.

BO comprises two key components: a probabilistic surrogate model, usually a Gaussian Process (GP), and an acquisition function. The GP approximates the unknown objective function, while the acquisition function guides the search for the global optimum.

2.2 Gaussian Processes

A Gaussian Process (GP) is a collection of random variables where any finite subset follows a joint Gaussian distribution [7]. It is characterized by a mean function $m(x)$ and a covariance function $k(x, x')$: $\hat{f}(x) \sim \mathcal{GP}(m(x), k(x, x'))$. The GP posterior is updated in each BO iteration, and the acquisition function is maximized to determine the next sampling location. This iterative process continues until the sampling budget is exhausted. The covariance function, or kernel, encodes assumptions about the function $f(x)$. A popular choice is the squared exponential kernel: $k(x, x') = \sigma^2 \exp\left(-\frac{(x-x')^2}{2l^2}\right)$.

Given a set of observations $D = \{(x_i, y_i)\}_{i=1}^n$, the GP posterior is updated, and the acquisition function is constructed to determine the next query point.

2.3 Acquisition Function

Acquisition functions guide the optimization process by quantifying the utility of evaluating the objective function at a new candidate point. The acquisition function, denoted by $a(x)$, is designed to balance exploration and exploitation. It is derived from the GP posterior and guides the selection of the next query point x_{n+1}. We entertain three choices of acquisition functions, including choices include Expected Improvement (EI) [4], q-Expected Improvement (qEI) [1], and Upper Confidence Bound (UCB) [6].

The Expected Improvement (EI) at a point x is given by: $EI(x) = (\mu(x) - f(x^+))\Phi(Z) + \sigma(x)\phi(Z)$, where $f(x^+)$ is the best-observed function value so far, Φ and ϕ are the CDF and PDF of the standard Normal distribution, respectively, and $Z = \frac{\mu(x) - f(x^+)}{\sigma(x)}$ if $\sigma(x) > 0$ and 0 otherwise.

The q-Expected Improvement (qEI) extends the EI to the case of batch evaluations. For a batch of points $\mathbf{x} = \{x_1, \ldots, x_q\}$, qEI is defined as: $qEI(\mathbf{x}) = \mathbb{E}\left[\min(f(x_1), \ldots, f(x_q)) - f(x^+)\right]$. The expectation is taken over the joint distribution of $f(x_1), \ldots, f(x_q)$.

The Upper Confidence Bound (UCB) acquisition function is given by: $UCB(x) = \mu(x) + \kappa\sigma(x)$, where $\mu(x)$ and $\sigma(x)$ are the mean and standard deviation of the GP posterior at x, respectively, and $\kappa > 0$ is the exploration parameter.

The next query point is then selected by maximizing the chosen acquisition function $x_{n+1} = \arg\max_x a(x)$. This process is repeated iteratively until a stopping criterion is met.

2.4 Weisfeiler-Lehman Kernel

In order to calculate the similarity between different neural network architectures, we adopt the Weisfeiler-Lehman (WL) Kernel as in [3]. The kernel leverages the Weisfeiler-Lehman test of isomorphism to perform efficient graph-based comparisons. This is particularly advantageous in the NAS context, where the search space is highly combinatorial and often requires efficient similarity measures for model selection and performance estimation.

Mathematically, the Weisfeiler-Lehman Kernel operates in an iterative manner to update the labels of each node in a graph based on its neighborhood. Let $G = (V, E)$ be a graph where V is the set of nodes and E the set of edges. Each node v has an initial label $l(v)$. At each iteration i, the label $l_i(v)$ of each node v is updated as follows:

$$l_i(v) = \text{hash}\left(l_{i-1}(v) \oplus \text{sort}\left(\bigoplus_{u \in N(v)} l_{i-1}(u)\right)\right)$$

Here, $\oplus$ denotes concatenation, $N(v)$ is the set of neighbors of v, and the hash function ensures that the new label is a compressed representation of the concatenated string.

After k iterations, the final labels of all nodes are used to compute a histogram $\phi(G)$, which serves as the feature vector of the graph. The kernel between two graphs G_1 and G_2 is then defined as the dot product between their respective feature vectors: $K(G_1, G_2) = \langle \phi(G_1), \phi(G_2) \rangle$.

In the context of NAS, each neural network architecture can be represented as a Directed Acyclic Graph (DAG), and the WL Kernel can be employed to measure the similarity between different architectures. This facilitates more informed search and selection processes in the network architecture.

Our BO-based NAS strategy is shown in Algorithm **??**.

Algorithm 1 Bayesian Optimization in NAS with WL Kernel

1: **Input:** Test size, Acquisition function $a(x)$, β for UCB
2: **Initialize:** Load and preprocess dataset
3: Evaluate initial architectures and initialize GP model with custom WL Kernel
4: best_loss $\leftarrow \infty$
5: **for** $t = 1, 2, \ldots, T$ **do**
6: Optimize acquisition function $a(x)$ to propose the next candidate architecture $\mathbf{x}_t$
7: Evaluate the neural network with architecture $\mathbf{x}_t$, get validation loss L_t
8: Update GP model with new data point $(\mathbf{x}_t, L_t)$
9: best_loss $\leftarrow \min(\text{best\_loss}, L_t)$
10: **end for**
11: **Output:** Architecture with the lowest validation loss

3 EXPERIMENTAL DESIGN

To rigorously evaluate the efficacy of BO in NAS, we conduct experiments on two widely used datasets: the Diabetes dataset and the Boston Housing dataset, both of which are readily accessible in the scikit-learn library. We assess the generalization performance across three different acquisition functions, including EI, qEI, and UCB, along with three distinct test set sizes: 0.1, 0.2, and 0.5.

Each dataset is initially partitioned into training and validation subsets according to the specified test size ratio. Subsequently, feature standardization is performed solely on the basis of the statistics of the training set. For model initialization, the GP model is initialized using four predefined neural network architectures with configurations (50, 50),

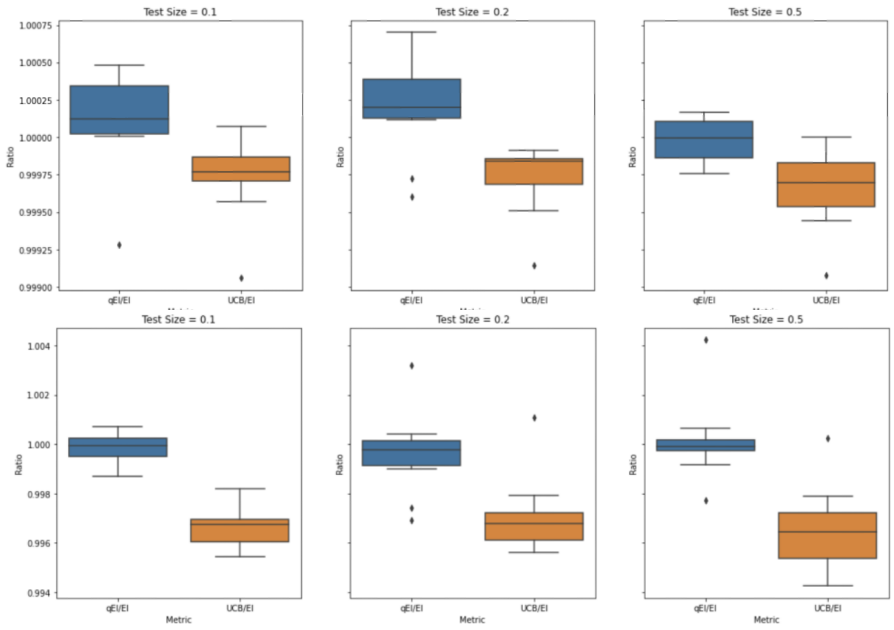

Fig. 1. Comparing the relative performance of validation error across three acquisition functions (EI, qEI, and UCB) and three designs (test size of 0.1, 0.2, and 0.5) in NAS. UCB consistently outperforms other alternatives in locating a network graph with a better validation performance.

$(100, 100)$, $(150, 150)$, and $(200, 200)$. We employ a Scale Kernel furnished with an Radial Basis Function (RBF) kernel as the base, augmented by our custom Weisfeiler-Lehman (WL) Kernel. The latter is specifically designed to compute the topological similarity between neural network graphs. Each optimization run spans 20 iterations for each of the acquisition functions. For the UCB acquisition function, we explore a range of trade-off parameters β within the interval $[0, 2]$, and report the empirically optimal settings for each experimental condition.

Figure 1 illustrates the relative validation error rates for each acquisition function and test size. Notably, the superiority of UCB becomes increasingly evident as the test size expands, implying greater efficacy in more complex learning scenarios. This underscores UCB's potential to outperform its counterparts in NAS, particularly when the optimization problem becomes more challenging.

4 CONCLUSION

In this paper, we empirically explore the use of BO in NAS and proposing a better performing network architecture using the WL kernel as a similarity metric. The WL Kernel enhances the Gaussian Process model's ability to capture intricate relationships in the architecture search space, providing a more nuanced approximation of the objective function. Our empirical findings reveal that integrating the UCB acquisition function within the BO framework exhibits a marked advantage in identifying superior neural network architectures, particularly under more complex and challenging optimization scenarios when there is less training data available.

5 ACKNOWLEDGEMENTS

Peng Liu has been supported by ASEAN Business Research Initiative (ABRI).

REFERENCES

[1] Janis Janusevskis, Rodolphe Le Riche, and David Ginsbourger. 2011. Parallel expected improvements for global optimization: summary, bounds and speed-up. (2011).

[2] Peng Liu, Haowei Wang, and Wei Qiyu. 2023. Bayesian Optimization with Switching Cost: Regret Analysis and Lookahead Variants. *International Joint Conference on Artificial Intelligence* (2023).

[3] Binxin Ru, Xingchen Wan, Xiaowen Dong, and Michael Osborne. 2020. Interpretable neural architecture search via bayesian optimisation with weisfeiler-lehman kernels. *arXiv preprint arXiv:2006.07556* (2020).

[4] Ilya O Ryzhov. 2016. On the convergence rates of expected improvement methods. *Operations Research* 64, 6 (2016), 1515–1528.

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

[6] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. 2009. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995* (2009).

[7] Christopher KI Williams and Carl Edward Rasmussen. 2006. *Gaussian processes for machine learning*. Vol. 2. MIT press Cambridge, MA.

[8] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. 2018. SNAS: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926* (2018).