

Optimizing deep learning models from multi-objective perspective via Bayesian optimization

Abdul Rahman Mohamad Rom, Nursuriati Jamil, Shafaf Ibrahim

Department of Computing Sciences, College of Computing, Informatics and Mathematics, Universiti Teknologi MARA,
Shah Alam, Malaysia

Article Info

Article history:

Received Mar 20, 2024

Revised Oct 25, 2024

Accepted Nov 14, 2024

Keywords:

Bayesian optimization
Convolutional neural network
Deep learning
Hyperparameter tuning
LeNet
Multilayer perceptron

ABSTRACT

Optimizing hyperparameters is crucial for enhancing the performance of deep learning (DL) models. The process of configuring optimal hyperparameters, known as hyperparameter tuning, can be performed using various methods. Traditional approaches like grid search and random search have significant limitations. In contrast, Bayesian optimization (BO) utilizes a surrogate model and an acquisition function to intelligently navigate the hyperparameter space, aiming to provide deeper insights into performance disparities between naïve and advanced methods. This study evaluates BO's efficacy compared to baseline methods such as random search, manual search, and grid search across multiple DL architectures, including multi-layer perceptron (MLP), convolutional neural network (CNN), and LeNet, applied to the Modified National Institute of Standards and Technology (MNIST) and CIFAR-10 datasets. The findings indicate that BO, employing the tree-structured parzen estimator (TPE) search method and expected improvement (EI) acquisition function, surpasses alternative methods in intricate DL architectures such as LeNet and CNN. However, grid search shows superior performance in smaller DL architectures like MLP. This study also adopts a multi-objective (MO) perspective, balancing conflicting performance objectives such as accuracy, F1 score, and model size (parameter count). This MO assessment offers a comprehensive understanding of how these performance metrics interact and influence each other, leading to more informed hyperparameter tuning decisions.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Shafaf Ibrahim

College of Computing, Informatics and Mathematics, Universiti Teknologi MARA

Shah Alam, Selangor, Malaysia

Email: shafaf2429@uitm.edu.my

1. INTRODUCTION

The performance of deep learning (DL) relies heavily on the hyperparameters [1], reinforcing the necessity for hyperparameter optimization. The process of finding optimal hyperparameters configuration for the DL models is referred to as hyperparameter tuning [1]. In the context of hyperparameter tuning, there are various widely known techniques such as manual search [2], grid search and random search [3]. In manual search, the process of finding optimal hyperparameter configuration is intervened by a human directly where individuals rely on intuition, and experiences [4]. Due to that, it is laborious, time consuming and prone to errors [5]. In contrast to manual search, grid search automates the exploration process, systematically traversing the hyperparameter space in sequential order. Yet, this approach's brute-force methodology incurs significant computational overhead, particularly as the dimensionality of the search space escalates exponentially [6].

Random search is an alternative to grid search, which adopts a random approach, sampling hyperparameters configuration in random order within the designated search space [7]. Compared to grid search, the implementation of random search evidently proven to be more effective especially in high-dimensional space [8]. Even with the randomness nature, the results obtained by random search is purely by 'luck' as it samples the hyperparameter search space without any guidance. It is evident that the implementation of grid search and random search exhibits a lack of sophistication, resulting in a naïve approach to hyperparameter tuning [8]. This method often leads to substantial computational resource consumption due to the exponential increase in search space [6]. Due to that, a modern approach which utilizes the data and finds optimal hyperparameter configuration intelligently is needed. Within the context of this study, Bayesian optimization (BO) is an alternative choice for the naïve methods. The utilization of BO leverages statistical model, employing surrogate model and acquisition function to guide BO for finding optimal hyperparameter configuration. BO known for its capabilities to optimize expensive function by iteratively constructing a probabilistic surrogate model of the underlying target function [9].

According to Nasayreh *et al.* [10], the implementation of BO, grid search and random search were tested on different machine learning (ML) models. Support vector machine (SVM), logistic regression (LR), random forest (RF), and naïve Bayes (NB) showed grid search and random search provided superior results in most tested models except LR. Based on the study conducted in [11], the implementation of BO was done by using Gaussian process (GP) as the surrogate model. Unfortunately, the process of hyperparameter tuning with large dimension space and small fitness evaluation budget showed that an alternative to GP is necessary [12]. In this study, the implementation of hyperparameter tuning was done using BO, an alternative to the brute-force methodology inherent in grid search and the stochastic nature of random search. BO offers a sophisticated approach to identifying optimal solutions. BO leverages prior data to intelligently navigate the search space by employing a probabilistic model, notably the tree-structured parzen estimator (TPE), which encapsulates the underlying objective function.

Within the context of hyperparameter tuning, previous studies have predominantly focused on the single objective optimization (SSO) as in [10]. SSO provides advantages such as reduced runtime and improved convergence; however, it limits performance evaluation to a single objective, precluding the consideration of conflicting objectives. The implementation of a single objective typically fails to meet the scenario of the real-world, where it involves many conflicting objectives. The occurrence of clashes between multiple conflicting objectives happens most of the time in the real-world scenario.

This study proposes multi-objective hyperparameter tuning by using BO on the different architectures of the DL models. The primary objective of this paper is to show that the implementation of BO for the hyperparameter tuning from multi-objective perspective provides a better performance compared to baseline methods on the different DL architectures, particularly when the search space grows exponentially when the new hyperparameters are added. In addition to that, this study offers a comprehensive exploration of the performance using different hyperparameter tuning methods within the context of multi-objective optimization.

2. METHODS

2.1. Multi-objective hyperparameter tuning

The effectiveness of the learning algorithm is heavily dependent on the configuration of hyperparameters, λ . The performance of a DL model can directly be impacted by the right hyperparameter configurations [1], [13]. Mathematically, the efficacy of a learning algorithm with designated hyperparameter is denoted as \mathcal{A}_λ , and $f = \mathcal{A}_\lambda(X^{(\text{train})})$ for a training set $X^{(\text{train})}$. As instance, in a convolutional neural network (CNN) model where the batch size is bs and learning rate denoted as l , the equation for λ will be written as $\lambda = (bs, l)$. In the context of hyperparameter configuration, the search space can be measured as stated in (1).

$$N = \prod_{i=1}^n m_i \quad (1)$$

As referring to (1), hyperparameters are represented in n m whereas the possible values of each hyperparameters are represented in m_i . Within the context of this study, the n (hyperparameters) is learning rate, epochs, batch size, kernel size, and neuron layers, depending on different architecture of multi-layer perceptron (MLP), LeNet, or CNN. Theoretically, based on the (1), the dimension of the search space increases exponentially when the new hyperparameters is added into the equation [14]. As the dimension of the search space grows bigger, the traditional approach of hyperparameter tuning is proven to be tedious, laborious, prone to errors, and consumes a lot of computing power [2].

Other than that, within the context of hyperparameter tuning, the conflicting objectives often arises when optimizing for multiple performance metrics. Theoretically, multi-objective optimization problem can be defined as in (2) [15].

$$\begin{aligned} \min \text{ or } \max: & f_1(x), f_2(x), \dots, f_n(x) \\ \text{subject to: } & x \in U \end{aligned} \quad (2)$$

Where x representing variables, n representing number of objective functions, and U is the feasible set, and min-max are objective functions. Now, within the context of DL, based on the previous studies, the conflicting arises between accuracy vs model size [16], specificity vs accuracy vs sensitivity [17], latency and accuracy [18]. In the context of this study, the conflicting objectives are not limit to bi-objective, but extended to tri-objectives which are accuracy, F1-score and weight of the model.

2.2. Bayesian optimization

In BO, there are two crucial components involved, which are the probabilistic surrogate model and the acquisition function [19]. BO leverages prior data to intelligently navigate the search space by constructing a probabilistic model (also known as a surrogate model) [20], notably the TPE, which encapsulates the underlying objective function. This model not only provides estimations of the objective function but also quantifies the uncertainty surrounding these estimations. Consequently, BO continually refines its understanding of the objective landscape, iteratively adapting its search strategy to converge towards optimal solutions with enhanced precision and efficiency.

In the context of hyperparameter tuning with large dimension space and small fitness evaluation budget, an alternative to typical GP is necessary [12]. In this experiment, the surrogate model will be used is TPE, which was introduced in [21]. The implementation of GP directly models $P(y/x)$ as stated in (3) [12].

$$P(y|x) = \frac{P(x|y)xP(y)}{P(x)} \quad (3)$$

Contradicts to GP, TPE concentrating on the approximation of the conditional probability $P(x|y)$, rather than directly modelling $P(y|x)$. This conditional probability, $P(x|y)$, is estimated using two distinct functions: $l(x)$ for cases where the performance is below a certain threshold, and $g(x)$ for cases where the performance surpasses the specified threshold are as stated in (4) [21]. In the context of this study, the threshold is adaptively adjusted and configured by using Optuna depending on the problems given.

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

These two density functions, $l(x)$ and $g(x)$, are subsequently employed in the expected improvement (EI) function. The EI, as represented by this equation, guides the decision of where to sample the next set of hyperparameters. In essence, this rephrasing aims to convey the key concepts of TPE's approach in approximating the conditional probability of hyperparameters given a score, with a specific focus on the functions $l(x)$ and $g(x)$ and their utilization in the EI function for determining optimal hyperparameter sampling points.

As mentioned previously, the acquisition function used in this experiment is EI. Alternatively, the other widely used acquisition function in BO is called probability of improvement (PI). The implementation of PI only considers the probability of improving our current best estimate, but it does not factor in the magnitude of the improvement. Contrary to that, the implementation of EI in BO is widely used as it considered both the probability and increasement of a point. In addition to that, EI is able to solve the problem of falling into the local optimum solution. In the scope of this study, the acquisition function that will be using in this study is EI. The mathematical notation for EI is as stated in (5) [2].

$$\max EI(x) = \begin{cases} (\mu(x) - f(x^+) - \xi)\Phi(Z) + \sigma(x)\phi(Z), & \sigma(x) > 0, \\ \sigma(x) = 0 \end{cases} \quad (5)$$

$\Phi(\cdot)$ and $\phi(\cdot)$ denote the cumulative distribution function and the probability density function of the standard normal distribution, respectively. Maximizing EI is corresponds to maximizing the ratio $\frac{l(x)}{g(x)}$ in TPE, as shown in (6), where y^* is some quantile of the observed y .

2.3. Workflow of the research

The workflow of the research will be discussed in more details. Starting from data acquisition and preparation, configuration of hyperparameter search space, the implementation of hyperparameter tuning on different datasets, different DL architectures, and performance evaluation. Figure 1 demonstrated the workflow of the research.

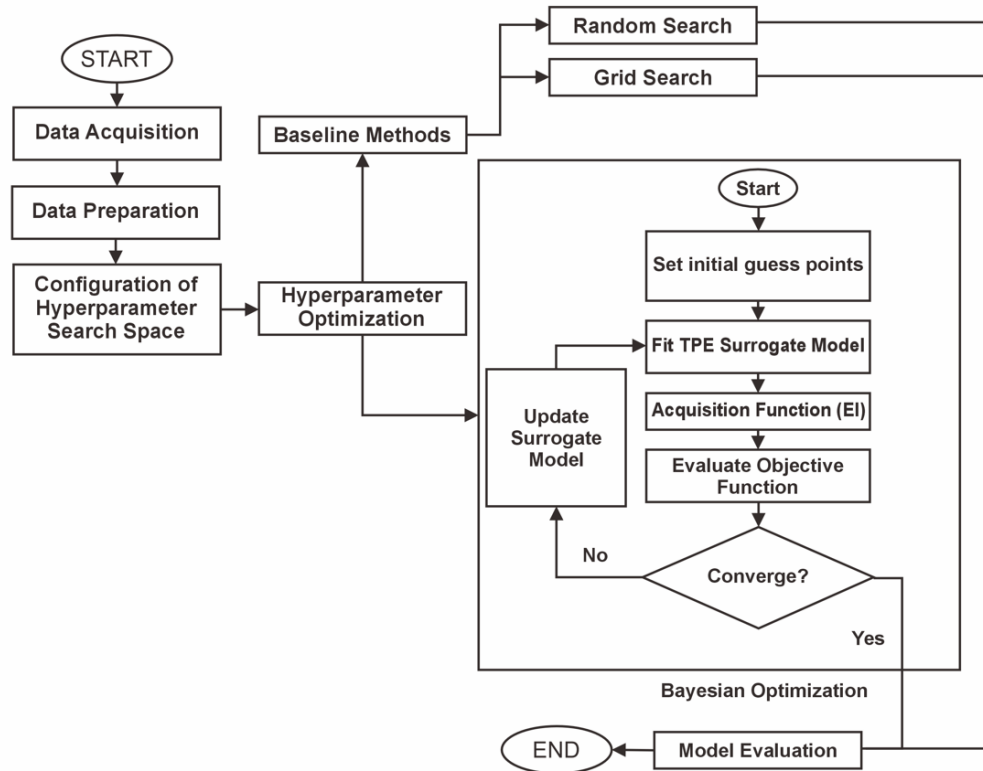


Figure 1. The workflow of the research

2.3.1. Data acquisition and preparation

This study utilized two benchmark datasets that are well-known in the DL field. One of the datasets that will be employed in this study is Modified National Institute of Standards and Technology (MNIST) dataset, which comprises of 70,000 28×28 grayscale images of handwritten digits. 60,000 of the images allocated for training, and another 10,000 for testing purposes [22]. Another benchmark dataset is CIFAR-10, which comprises of 60,000 colour images evenly distributed across ten different classes of vehicles, and animals [23]. Similar to MNIST, CIFAR-10 is also a tiny image which sized at 32×32 pixels adding extra layer of complexity to the classification task. Both datasets have played a crucial role in the development of DL techniques.

2.3.2. Configuration of hyperparameter search space

The configuration of the hyperparameter search space varies based on the specific DL architecture and the hyperparameters under consideration. In this study, the hyperparameter search space for a MLP includes combinations of the following hyperparameters: optimizer, learning rate, number of epochs, and batch size. For the LeNet architecture, the hyperparameter search space extends beyond those of the MLP to include additional parameters: kernel size for the first convolutional layer (kernel_size_1), kernel size for the second convolutional layer (kernel_size_2), filter size for the first convolutional layer (filter_size_1), and filter size for the second convolutional layer (filter_size_2). In the case of a CNN, the hyperparameter search space encompasses all the hyperparameters of both MLP and LeNet architectures, with further extensions to include the kernel size for the third convolutional layer (kernel_size_3) and the filter size for the third convolutional layer (filter_size_3).

2.3.3. Hyperparameter tuning using Bayesian optimization

As mentioned previously, there are two crucial components involved in BO, which are surrogate model and acquisition function [24]. The purpose of surrogate model acting as a probabilistic model, which encapsulates the underlying objective function, and the purpose of acquisition function is to dictate the next point for evaluation by balancing exploration (sampling uncertain regions) and exploitation (sampling regions with high EI) [20]. The implementation of surrogate model in the context of this study will be using TPE approach, whereas the implementation of acquisition function will be employed by using EI.

The workflow of BO begins with the initial sampling which randomly set the initial guess points within the hyperparameter search space. Next, the surrogate model, the surrogate model (TPE) is applied to the initial data points. TPE divides the data into two subsets: one for exploration, and one for exploitation. The exploration subset consists of data points with low objective function values, while the exploitation subset contains data points with high objective function values. TPE then fits separate models to these subsets, estimating the probability density functions (PDFs) of the hyperparameters given their corresponding objective function values. These models provide predictions and uncertainty estimates for unexplored regions of the search space.

This model captures the underlying behaviour of the objective function and provides predictions along with uncertainty estimates for unexplored region of the search space. With the surrogate model established, the next challenge is to determine on how to identify the next points that yield the maximum of the objective function. In the context of BO, the acquisition function can be employed to guide the search for the optimal hyperparameters.

The point suggested by the acquisition function is evaluated by computing the true objective function. This evaluation provides new data points that are used to update the surrogate model. After evaluating the objective function at the suggested point, the surrogate model is updated to incorporate the new data. This update refines the model's understanding of the objective landscape, improving its predictive capability. The process will be repeated until the termination criteria is met. At every iteration, the surrogate model is refined, and the acquisition function guides the search towards regions of the search space likely to contain optimal solutions.

2.3.4. Performance assessment

In this study, the assessment of performance will be assessed from multi-objective perspective trading off between accuracy, F1-score and the weight of the model. Classification metrics, particularly accuracy, will be evaluated through the application of confusion matrix. Table 1 presents a confusion matrix from which we will derive two metrics: accuracy (6) and F1-score (7).

Table 1. The confusion matrix for accuracy and F1-score calculation

	Positive	Negative
Positive	True positive (TP)	False negative (FN)
Negative	False positive (FP)	True negative (TN)

$$Accuracy = \frac{TN + TP}{TN + FN + TP + FP} \quad (6)$$

In addition to accuracy, the F1-score will also be evaluated as a classification parameter in this study. Defined as the harmonic mean of precision and recall [25], the F1-score offers a balanced assessment of model efficacy. F1-score can be calculated by utilizing precision and recall as mentioned in (7). Precision and recall are integral components of the F1-score. Precision quantifies the accuracy of positive predictions, representing the ratio of correctly identified positive instances to all instances predicted as positive. Recall, conversely, measures the model's ability to detect positive instances, defined as the proportion of correctly identified positive cases among all actual positive instances in the dataset. The formula for measuring the F1 score is presented in (7).

$$\begin{aligned} Recall &= \frac{TP}{TP + FN} \\ Precision &= \frac{TP}{TP + FP} \\ F1 \text{ score} &= 2 * (\text{precision} * \text{recall}) / (\text{precision} + \text{recall}) \end{aligned} \quad (7)$$

In addition to classification metrics, the model's efficiency will be assessed by examining its computational complexity. This evaluation will be conducted by calculating the model's weight, which is determined by the total number of parameters it contains. Similar approach of measuring efficiency of the

model was seen in [26]. The performance of the model will be evaluated from the multi-objective point of view, using scalarization approach. In order to do that, a weighted sum will be implemented to measure the performance from multi-objective perspective. In (8) shows the formula for weighted sum to calculate three conflicting objectives in a scalarization equation.

$$\text{weighted_sum} = (F1_score * w1) + (Accuracy * w2) + (\text{inversed_normalized_params} * w3) \quad (8)$$

As seen in the (8), w refers to the weight of performance metrics. In the context of this study, the weightage ($w1$, $w2$, $w3$) are equivalent as there is no prioritize between one and another conflicting objective. Hence, all weightages are set to 1.

3. RESULTS AND DISCUSSION

This section discusses the evaluation of a proposed method which is hyperparameter tuning via BO on different DL architectures from multi-objective perspective. Based on the experiment conducted, is proven that the performance of hyperparameter tuning by using BO to be more cost effective from the perspective of computing power, convergence time, and the performance of the DL model. Subsections 3.1 and 3.2 discusses the performance of the DL architecture on MNIST and CIFAR-10 datasets.

3.1. MNIST dataset

In this subsection, the result of the hyperparameter tuning by using BO on different DL architectures namely MLP, LeNet, and CNN, are recorded. Tables 2 to 4 shows the top-5 optimal hyperparameter configurations on MLP, LeNet, and CNN on MNIST dataset. As according to Table 2, the identified configuration for MLP architecture on MNIST dataset consists of Adamax optimizer, a learning rate of 0.0001, batch size of 64, and 50 epochs. This configuration yielded impressive performance metrics: accuracy of 0.9823 and F1 score of 0.9822. The resulting model comprises 101,770 parameters, which corresponds to 1.0 normalized parameters. The cumulative values for normalized params, accuracy and F1 score added up to 2.9645. In the different architecture of LeNet, as referring to Table 3, the process of hyperparameter tuning produced a notable outcome of an accuracy of 0.9932, F1 score of 0.9931, 115,902 number of parameters, which equivalent to 1.0 after normalized summing up to 2.9863. The hyperparameter combination for above result is (kernel size 1=4, kernel size 2=4, filters 1=16, filters 2=48, optimizer=RMSprop, learning rate=0.1, batch size=128, epochs=50, activation = relu).

Table 2. Hyperparameter tuning results for MLP on MNIST dataset using BO

OPT	LR	EP	BS	Accuracy	F1 score	Params	Normalized params	Weighted sum
Adamax	0.0001	50	64	0.9823	0.9822	101770.0	1	2.9645
Adamax	0.0001	50	64	0.9818	0.9817	101770.0	1	2.9635
RMSprop	0.0001	40	64	0.9818	0.9816	101770.0	1	2.9634
Adamax	0.01	50	64	0.9816	0.9814	101770.0	1	2.963
Adadelata	0.001	50	64	0.9815	0.9813	101770.0	1	2.9628

Table 3. Hyperparameter tuning results for LeNet on MNIST dataset using BO

OPT	LR	F1	K1	ACT	F2	K2	EP	BS	Accuracy	F1 score	Params	Normalized params	Weighted sum
RMSprop	0.1	16	4	relu	48	4	50	128	0.9932	0.9931	115902.0	1.0	2.9863
RMSprop	0.1	16	4	relu	48	4	50	128	0.9932	0.9931	115902.0	1.0	2.9863
Adagrad	0.1	16	4	relu	48	4	50	128	0.9928	0.9927	115902.0	1.0	2.9855
RMSprop	0.1	16	4	relu	48	4	50	128	0.9927	0.9926	115902.0	1.0	2.9853
Adagrad	0.1	16	4	relu	48	4	50	128	0.9926	0.9925	115902.0	1.0	2.9851

Table 4. Hyperparameter tuning results for CNN on MNIST dataset using BO

OPT	LR	F1	K1	ACT	F2	K2	F3	K3	EP	BS	Accuracy	F1 score	Params	Normalized params	Weighted sum
RMSprop	0.0001	16	4	elu	48	3	64	3	50	256	0.9932	0.9931	40714.0	0.9997	2.9861
Adadelata	0.01	16	4	elu	48	3	64	3	50	256	0.993	0.9929	40714.0	0.9997	2.9857
RMSprop	0.01	16	4	elu	48	3	64	3	50	256	0.9929	0.9928	40714.0	0.9997	2.9855
Adadelata	0.01	16	4	elu	48	3	64	3	50	256	0.9926	0.9925	40714.0	0.9997	2.9848
Adadelata	0.01	16	4	elu	48	3	64	3	50	256	0.9925	0.9924	40714.0	0.9997	2.9847

Similarly in the CNN architecture, the most effective combination of hyperparameter found by BO is articulated as (kernel size 1=4, kernel size 2=3, kernel size 3=3, filters 1=16, filters 2=48, filters 3=64, optimizer=RMSprop, learning rate=0.0001, batch size=256, epochs=50, activation = elu), yielding an accuracy of 0.9932, an F1 score of 0.9931, and 40,714 number of parameters equivalent to normalized parameters of 0.9997. The combined value of these three metrics is 2.9861. The results underscore the effectiveness of BO in navigating the complex hyperparameter landscape to identify optimal configurations. Across all architectures, the optimized configurations achieved high accuracy and F1 scores, demonstrating their effectiveness in accurately classifying digits in the MNIST dataset.

3.2. CIFAR-10 dataset

In this subsection, the result of the hyperparameter tuning by using BO on different DL architectures namely MLP, LeNet, and CNN are recorded. Tables 5 to 7 shows the top-5 optimal hyperparameter configurations on MLP, LeNet, and CNN on CIFAR-10 dataset. Tables 5 to 7 present the top five outcomes of hyperparameter tuning on the CIFAR-10 dataset across different architectures, including MLP, LeNet, and CNN. As according to Table 5, the optimal hyperparameter configuration for MLP dataset obtained by using BO is identified as (optimizer=Adamax, learning rate=0.001, batch size=256, epochs=50), yielding an accuracy of 0.4851, F1 score of 0.4789, number of params of 394,634 which equivalent to 1.0 normalized params. The cumulative values for normalized params, accuracy and F1 score added up to 1.964.

Table 5. Hyperparameter tuning results for MLP on CIFAR-10 dataset using BO

OPT	LR	EP	BS	Accuracy	F1 score	Params	Normalized params	Weighted sum
Adamax	0.001	50	256	0.4851	0.4789	394634.0	1	1.964
SGD	0.001	50	256	0.4851	0.4788	394634.0	1	1.9639
Adamax	0.001	50	512	0.4826	0.4786	394634.0	1	1.9612
RMSprop	0.001	50	256	0.483	0.4778	394634.0	1	1.9608
RMSprop	0.1	50	512	0.4819	0.4787	394634.0	1	1.9606

Table 6. Hyperparameter tuning results for LeNet on CIFAR-10 dataset using BO

OPT	LR	F1	K1	ACT	F2	K2	EP	BS	Accuracy	F1 score	Params	Normalized params	Weighted sum
Nadam	0.1	16	4	relu	48	4	10	64	0.6787	0.6739	168254.0	0.8902	2.2428
Adadelata	0.1	16	4	relu	48	4	10	64	0.6663	0.6663	168254.0	0.8902	2.2228
Adadelata	0.1	16	4	relu	48	4	10	64	0.6671	0.6626	168254.0	0.8902	2.2199
Nadam	0.1	16	4	relu	48	4	10	64	0.6642	0.6652	168254.0	0.8902	2.2196
Adadelata	0.1	16	4	relu	48	4	10	64	0.6625	0.6626	168254.0	0.8902	2.2153

Table 7. Hyperparameter tuning results for CNN on CIFAR-10 dataset using BO

OPT	LR	F1	K1	ACT	F2	K2	F3	K3	EP	BS	Accuracy	F1 score	Params	Normalized params	Weighted sum
Adagrad	0.0001	16	4	elu	48	3	64	3	30	256	0.6984	0.6968	45706.0	0.9898	2.385
Adadelata	0.0001	32	3	elu	48	3	64	3	30	256	0.704	0.7027	52730.0	0.9758	2.3825
Adadelata	0.0001	32	3	elu	48	3	64	3	30	256	0.7024	0.7009	52730.0	0.9758	2.3791
SGD	0.0001	16	4	elu	48	3	64	3	30	256	0.6936	0.6922	45706.0	0.9898	2.3756
Adadelata	0.1	16	3	relu	48	4	64	3	50	512	0.6978	0.6964	50746.0	0.9798	2.374

In the different architecture of LeNet, as referring to Table 6, the process of hyperparameter tuning produced a notable outcome of an accuracy of 0.6787, F1 score of 0.6739, 168,254 number of parameters, which equivalent to 0.8902 after normalized summing up to 2.2428. The hyperparameter combination for above result is (kernel size 1=4, kernel size 2=4, filters 1=16, filters 2=48, optimizer=Nadam, learning rate=0.1, batch size=64, epochs=10, activation = relu). On the other hand, in LeNet architecture, the best hyperparameter produced an accuracy of 0.6822, F1-score of 0.6827, 18,1093,54 number of parameters which equivalent to 0.8864 after normalized the parameters summing up to 2.2513. The hyperparameter combination contributing to this outcome is specified as (kernel size 1=4, kernel size 2=4, kernel size 3=4, kernel size 4=2, kernel size 5=2, filters 1=96, filters 2=64, filters 3=64, filters 4=128, filters 5=64, optimizer=Adadelata, learning rate=0.1, batch size=256, epochs=30, activation = relu).

Similarly in the CNN architecture, the most effective combination of hyperparameter found by BO is articulated as (kernel size 1=4, kernel size 2=3, kernel size 3=3, filters 1=16, filters 2=48, filters 3=64, optimizer=RMSprop, learning rate=0.0001, batch size=256, epochs=30, activation = elu), yielding an accuracy of 0.6984, an F1 score of 0.6968, and 45,706 number of parameters equivalent to normalized parameters of 0.9898. The combined value of these three metrics is 2.385. Comparing these results to the MNIST dataset, the CIFAR-10 dataset presents greater challenges, as evidenced by the lower average

accuracy of approximately 61%. Nonetheless, the hyperparameter tuning process successfully identified configurations that significantly improved model performance across all architectures, demonstrating the effectiveness of BO in optimizing DL models for image classification tasks.

3.3. Bayesian optimization and baseline methods

This subsection will be discussing on the performance of BO with the other baseline methods in different DL architectures on MNIST and CIFAR-10 datasets. Tables 8 to 10 shows the performance of the hyperparameter tuning by using BO and other baseline methods namely manual search, grid search, random search and BO. As referred to Table 8, the outcomes of hyperparameter tuning on the MLP architecture evidently shows the outstanding performance compared to other alternative methods. Grid search demonstrates its superiority by exhaustively sampling every hyperparameter configuration within the defined search space. In contrast, manual search relies on trial-and-error, while random search employs stochastic sampling techniques. BO utilizes probabilistic models. However, in theory, none of these methods can surpass grid search, as it ensures that no hyperparameter configuration is left unexplored. This advantage is particularly notable due to the relatively constrained hyperparameter search space inherent in MLP architecture. However, the results reveal a slight variation in performance, with grid search achieving a cumulative weighted sum of 2.966, followed closely by random search with 2.965 and BO with 2.964 on the MNIST dataset. On the CIFAR-10 dataset, hyperparameter tuning through grid search resulted in a weighted sum of 2.048, followed by manual search with 2.014, random search with 2.005, and BO with 1.964.

Following to that, the performance of LeNet architecture as recorded in Table 9 shows that the performance of BO outperforms other baselined methods on both MNIST and CIFAR-10 dataset. Within the context of MNIST dataset, BO yielded the weighted sum of 2.9863 follows by random search with 2.9845, grid search of 2.9169 and manual search with 2.8917. On CIFAR-10 dataset, BO yielded the weighted sum of 2.2428 follows by random search with 2.1424, manual search of 2.0483, and lastly grid search of 2.0466.

Table 8. MLP architecture performance: outcomes of hyperparameter tuning

Dataset	Hyperparameter tuning technique	MLP			Inversed normalized params	Weighted sum
		Accuracy	F1 score	Number of params		
MNIST	Manual search	0.9813	0.9812	101,770	1.0	2.962
	Grid search	0.9833	0.9832	101,770	1.0	2.966
	Random search	0.9826	0.9829	101,770	1.0	2.965
	Bayesian optimization	0.9823	0.9822	101,770	1.0	2.964
CIFAR-10	Manual search	0.5084	0.5058	394,634	1.0	2.014
	Grid search	0.5260	0.5223	394,634	1.0	2.048
	Random search	0.5051	0.5005	394,634	1.0	2.005
	Bayesian optimization	0.4851	0.4789	394,634	1.0	1.964

Table 9. LeNet architecture performance: outcomes of hyperparameter tuning

Dataset	Hyperparameter tuning technique	LeNet			Inversed normalized params	Weighted sum
		Accuracy	F1 score	Number of params		
MNIST	Manual search	0.9824	0.9823	150,374	0.927	2.8917
	Grid search	0.9661	0.9658	122,702	0.985	2.9169
	Random search	0.9923	0.9922	115,902	1.0	2.9845
	Bayesian optimization	0.9932	0.9931	115,902	1.0	2.9863
CIFAR-10	Manual search	0.6523	0.653	238,398	0.743	2.0483
	Grid search	0.6847	0.6839	269,486	0.677	2.0466
	Random search	0.6935	0.6984	235,118	0.750	2.1424
	Bayesian optimization	0.6787	0.6739	168,254	0.890	2.2428

Table 10. CNN architecture performance: outcomes of hyperparameter tuning

Dataset	Hyperparameter tuning technique	CNN			Inversed normalized params	Weighted sum
		Accuracy	F1 score	Number of params		
MNIST	Manual search	0.977	0.9769	82,970	0.916	2.8699
	Grid search	0.9907	0.9906	92,698	0.896	2.8773
	Random search	0.9923	0.9923	40,602	1.0	2.9846
	Bayesian optimization	0.9932	0.9931	40,714	0.999	2.9861
CIFAR-10	Manual search	0.6631	0.6604	133,450	0.815	2.1385
	Grid search	0.6579	0.6578	133,466	0.815	2.1307
	Random search	0.7082	0.7087	94,202	0.893	2.3109
	Bayesian optimization	0.6984	0.6968	45,706	0.9898	2.3842

Similarly on CNN architecture, the performance of BO once again stands out comparing to other baselined methods on both datasets. On MNIST dataset, BO achieves a weighted sum of 2.9861, follows by random search with 2.9846, grid search with 2.8773, and manual search with 2.8699. On the CIFAR-10 dataset, BO obtained 2.3842 of weighted sum comparing to other baselined methods which achieved 2.3109 (random search), 2.1385 (manual search) and 2.1307 (grid search).

4. CONCLUSION

BO stands out from manual search, grid search, and random search by leveraging past data to guide the next iteration, aiming for an optimal solution. Unlike grid search, where the hyperparameter search space exponentially increases with the addition of new hyperparameters, BO offers a more efficient approach, reducing computing costs. Additionally, random search's inconsistency due to its random nature contrasts with BO's ability to provide more consistent results. However, it's important to acknowledge that BO is not without its weaknesses. One limitation is its reliance on probabilistic models, which may not always accurately capture the underlying complexities of the hyperparameter space. Furthermore, BO's performance heavily depends on the configuration of its hyperparameters, and the quality of the surrogate model used. Future studies could focus on addressing these limitations and further refining the BO approach. Research avenues might include exploring more advanced surrogate models, enhancing the acquisition function to better balance exploration and exploitation, and investigating strategies to handle noisy or uncertain evaluations. Other than that, future studies could focus more on other modern optimization techniques, such as heuristic or swarm intelligence approach, which could be beneficial within the field of hyperparameter tuning from the perspective of multi-objective. In addition to that, comparative studies could delve deeper into understanding the trade-offs between BO and other hyperparameter tuning techniques across a wider range of DL architectures and datasets.

ACKNOWLEDGEMENTS

The research was supported by Ministry of Higher Education Malaysia (MoHE), and Universiti Teknologi MARA through the Fundamental Research Grant Scheme (FRGS) (600-RMC/FRGS 5/3 (024/2021)).





REFERENCES

- [1] R. Krithiga and E. Ilavarasan, "Hyperparameter tuning of adaboost algorithm for social spammer identification," *International Journal of Pervasive Computing and Communications*, vol. 17, no. 5, pp. 462–482, 2020, doi: 10.1108/IJPC-09-2020-0130.
- [2] L. Wen, X. Ye, and L. Gao, "A new automatic machine learning based hyperparameter optimization for workpiece quality prediction," *Measurement and Control*, vol. 53, no. 7–8, pp. 1–11, 2020, doi: 10.1177/0020294020932347.
- [3] A. M. Vincent and P. Jidesh, "An improved hyperparameter optimization framework for automl systems using evolutionary algorithms," *Scientific Reports*, vol. 13, no. 1, 2023, doi: 10.1038/s41598-023-32027-3.
- [4] H.-C. Kim and M.-J. Kang, "Comparison of hyper-parameter optimization methods for deep neural networks," *Journal of IKEE*, vol. 24, no. 4, pp. 969–974, 2020, doi: 10.7471/ikee.2020.24.4.969.
- [5] A. A. R. K. Bsoul, M. A. Al-Shannaq, and H. M. Aloqool, "Maximizing cnn accuracy: a bayesian optimization approach with gaussian processes," *9th 2023 International Conference on Control, Decision and Information Technologies, CoDIT 2023*, pp. 2597–2602, 2023, doi: 10.1109/CoDIT58514.2023.10284448.
- [6] X. Zhang, S. Kuenzel, N. Colombo, and C. Watkins, "Hybrid short-term load forecasting method based on empirical wavelet transform and bidirectional long short-term memory neural networks," *Journal of Modern Power Systems and Clean Energy*, vol. 10, no. 5, pp. 1216–1228, 2022, doi: 10.35833/MPCE.2021.000276.
- [7] J. A. Pandian *et al.*, "A five convolutional layer deep convolutional neural network for plant leaf disease detection," *Electronics*, vol. 11, no. 8, 2022, doi: 10.3390/electronics11081266.
- [8] A. R. M. Rom, N. Jamil, and S. Ibrahim, "Multi objective hyperparameter tuning via random search on deep learning models," *Telkomnika (Telecommunication Computing Electronics and Control)*, vol. 22, no. 4, pp. 956–968, 2024, doi: 10.12928/TELKOMNIKA.v22i4.25847.
- [9] A. Mathern *et al.*, "Multi-objective constrained bayesian optimization for structural design," *Structural and Multidisciplinary Optimization*, vol. 63, no. 2, pp. 689–701, 2021, doi: 10.1007/s00158-020-02720-2.
- [10] A. Nasayreh *et al.*, "Arabic sentiment analysis for chatgpt using machine learning classification algorithms: a hyperparameter optimization technique," *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 23, no. 3, 2024, doi: 10.1145/3638285.
- [11] O. Stephen and M. Sain, "Using deep learning with bayesian-gaussian inspired convolutional neural architectural search for cancer recognition and classification from histopathological image frames," *Journal of Healthcare Engineering*, vol. 2023, 2023, doi: 10.1155/2023/4597445.
- [12] S. Hanifi, A. Cammarono, and H. Zare-Behtash, "Advanced hyperparameter optimization of deep learning models for wind power prediction," *Renewable Energy*, vol. 221, 2024, doi: 10.1016/j.renene.2023.119700.
- [13] A. Morales-Hernández, I. V. Nieuwenhuyse, and S. R. Gonzalez, "A survey on multi-objective hyperparameter optimization algorithms for machine learning," *Artificial Intelligence Review*, vol. 56, no. 8, pp. 8043–8093, 2023, doi: 10.1007/s10462-022-10359-2.





- [14] M. A. Amirabadi, M. H. Kahaei, and S. A. Nezamalhosseni, "Novel suboptimal approaches for hyperparameter tuning of deep neural network [under the shelf of optical communication]," *Physical Communication*, vol. 41, 2020, doi: 10.1016/j.phycom.2020.101057.
- [15] B. Gülmez, "A new multi-objective hyperparameter optimization algorithm for covid-19 detection from x-ray images," *Soft Computing*, vol. 28, pp. 11601–11617, 2024, doi: 10.1007/s00500-024-09872-z.
- [16] L. Fromberg, T. Nielsen, F. D. Frumosu, and L. K. H. Clemmensen, "Beyond accuracy: fairness, scalability, and uncertainty considerations in facial emotion recognition," in *Proceedings of the 5th Northern Lights Deep Learning Conference (NLDL)*, 2024.
- [17] S. S. Mostafa, F. Mendonca, A. G. Ravelo-Garcia, G. Julia-Serda, and F. Morgado-Dias, "Multi-objective hyperparameter optimization of convolutional neural network for obstructive sleep apnea detection," *IEEE Access*, vol. 8, pp. 129586–129599, 2020, doi: 10.1109/ACCESS.2020.3009149.
- [18] S. P. Chen, J. Wu, and X. Y. Liu, "EMORL: effective multi-objective reinforcement learning method for hyperparameter optimization," *Engineering Applications of Artificial Intelligence*, vol. 104, 2021, doi: 10.1016/j.engappai.2021.104315.
- [19] H. Albrahim and S. A. Ludwig, "Hyperparameter optimization: comparing genetic algorithm against grid search and bayesian optimization," *2021 IEEE Congress on Evolutionary Computation, CEC 2021*, pp. 1551–1559, 2021, doi: 10.1109/CEC45853.2021.9504761.
- [20] X. Wang, Y. Jin, S. Schmitt, and M. Olhofer, "Recent advances in bayesian optimization," in *ACM Computing Surveys*, vol. 55, no. 13s, Jul. 2023, pp. 1–36, doi: 10.1145/3582078.
- [21] J. Bergstra, R. Bardenet, Y. Bengio and B. Kégl, "Algorithms for hyper-parameter optimization," in *NIPS'11: Proceedings of the 24th International Conference on Neural Information Processing Systems*, 2011.
- [22] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012, doi: 10.1109/MSP.2012.2211477.
- [23] Q. Gou and Y. Ren, "Research on multi-scale cnn and transformer-based multi-level multi-classification method for images," *IEEE Access*, vol. 12, pp. 103049–103059, 2024, doi: 10.1109/ACCESS.2024.3433374.
- [24] Z. Gao and D. S. Boning, "A review of bayesian methods in electronic design automation," *arXiv-Statistics*, pp. 1–24, 2023.
- [25] H. M. Rai, K. Chatterjee, and S. Dashkevich, "Automatic and accurate abnormality detection from brain mr images using a novel hybrid unetresnext-50 deep CNN model," *Biomedical Signal Processing and Control*, vol. 66, 2021, doi: 10.1016/j.bspc.2021.102477.
- [26] S. M. Jeong, S. G. Lee, C. L. Seok, E. C. Lee, and J. Y. Lee, "Lightweight deep learning model for real-time colorectal polyp segmentation," *Electronics*, vol. 12, no. 9, 2023, doi: 10.3390/electronics12091962.

BIOGRAPHIES OF AUTHORS







Abdul Rahman Mohamad Rom     obtained his bachelor's degree in computer science with first class honours in the year 2020 from the esteemed Universiti Teknologi MARA (UiTM). His academic journey continues as he is currently a fast-track Ph.D. student at UiTM, now entering his third year of rigorous doctoral studies. In addition to his pursuit of advanced knowledge, he serves as a graduate research assistant, actively contributing to the realm of academic research. His commitment to the field is evident through his diligent work and dedication to expanding the boundaries of computer science. His research interests encompass cutting-edge topics in computer science and technology, reflecting his unwavering dedication to advancing knowledge in this dynamic domain. He can be contacted at email: rahmanrom@gmail.com.



Nursuriati Jamil     is a Professor in College of Computing, Informatics and Mathematics, Universiti Teknologi MARA. She holds a bachelor's degree and masters in computer science. Having completed her Ph.D. in information sciences, her researches mainly focused in the area of artificial intelligence, pattern recognition and image recognition, healthcare applications, internet of things, and multimedia information retrieval. She can be contacted at email: liza_jamil@uitm.edu.my.



Shafaf Ibrahim     is an Associate Professor in College of Computing, Informatics and Mathematics, Universiti Teknologi MARA Shah Alam, Malaysia. She holds a diploma, bachelor's degree, masters, and Ph.D. in computer science. Her research interests are artificial intelligence, evolutionary algorithms, deep learning, machine learning, and image processing. She can be contacted at email: shafaf2429@uitm.edu.my.