

Multi objective hyperparameter tuning via random search on deep learning models

Abdul Rahman Mohamad Rom, Nursuriati Jamil, Shafaf Ibrahim

College of Computing, Informatics and Mathematics, Universiti Teknologi MARA, Selangor, Malaysia

Article Info

Article history:

Received Nov 7, 2023

Revised Mar 6, 2024

Accepted Mar 29, 2024

Keywords:

Convolutional neural network
Deep learning models
Hyperparameter tuning
Multiobjective optimization
Random search

ABSTRACT

This research examines the efficacy of random search (RS) in hyperparameter tuning, comparing its performance to baseline methods namely manual search and grid search. Our analysis spans various deep learning (DL) architectures-multilayer perceptron (MLP), convolutional neural network (CNN), and AlexNet implemented on prominent benchmark datasets of Modified National Institute of Standards and Technology (MNIST) and Canadian Institute for Advanced Research-10 (CIFAR-10). In the context of this study, the evaluation will be adopting a multi-objective framework, navigating the delicate trade-offs between conflicting performance metrics, including accuracy, F1-score, and model parameter size. The primary objective of employing a multi-objective evaluation framework is to enhance the understanding regarding the interactions of these performance metrics interact and influence each other. In real-world scenarios, DL models often need to strike a balance between these conflicting objectives. This research adds to the increasing wealth of knowledge in hyperparameter tuning for DL models and serves as a reference point for practitioners seeking to optimize their DL architectures. The results of our analysis are positioned to provide invaluable insights into the intricate balancing act required during the process of hyperparameter fine-tuning. These insights will contribute to the ongoing advancement of best practices in optimizing DL models and facilitating the ongoing optimization of the DL models.

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

Selangor, Malaysia

Email: shafaf2429@uitm.edu.my

1. INTRODUCTION

The advancement in artificial intelligence (AI) and deep learning (DL) have exerted a profound impact on both scientific research and industry, and the application of DL to practical problems has been extensively researched across a multitude of domains, including but not limited to medical diagnosis [1], scene classification [2], autonomous vehicles [3], and among others. The rapid progress in DL has been facilitated by several factors including access to vast amount of data, advancement in computational hardware, and the development of the novel algorithm [4]. In DL implementations, the selection of appropriate hyperparameters is very crucial as they hold a significant responsibility over the performance of the DL model [5], [6]. The process of identifying the optimal set of hyperparameter is known as hyperparameter tuning [7]. Recognizing the best choice of a hyperparameter is often a cumbersome process to a level that some people consider it as a “black art” [8].

Currently, the predominant usage of hyperparameter tuning falls in the category of single objective optimization (SOO). Numerous studies were done in the context of hyperparameter tuning [9]–[11]. These

studies were done by using hyperparameter tuning by using SOO approach which undeniably benefits from the lower runtime, and a better convergence, but it restricts the performance evaluation to only a single objective limiting its practical applicability to meet the need of optimization in the real-world scenario. In such real-world contexts, the clashing between two conflicting objectives often arises, highlighting the need for more versatile optimization strategies.

Thus, in this paper, the author proposes a multi-objective hyperparameter tuning in DL models, utilizing a random search to identify the optimal hyperparameter configuration. As such, the target of this research is to demonstrate the enhanced performance of DL architectures when utilizing random search, particularly as the hyperparameter search space expands, in comparison to other conventional methods. In addition to that, this investigation provides a thorough examination of performance metrics across various hyperparameter tuning techniques within the realm of multi-objective optimization.

2. METHODS

This section includes the mechanism used namely multi-objective hyperparameter tuning and hyperparameter tuning techniques which focuses on baseline methods and random search together with the workflow of the research.

2.1. Multi objective hyperparameter tuning

Hyperparameters configuration λ is one of the key factors to the effectiveness of a learning algorithm \mathcal{A} , either to minimize, or maximize a specific function, f . Finding the right hyperparameter configuration can directly impact the performance of a DL model [6]. The act of identifying the optimal hyperparameters is referred to as hyperparameter tuning [7], which is one of the crucial processes in the development of the DL model. Mathematically, the effectiveness of a learning algorithm with assigned hyperparameters can be written as \mathcal{A}_λ , and $f = \mathcal{A}_\lambda(X^{(train)})$ for a training set $X^{(train)}$. For example, with a convolutional neural network (CNN) model, where learning rate is l and epoch size as e , the $\lambda = (l, e)$. Now, the search space for hyperparameter configuration in the context of machine learning (ML) can be defined mathematically using the product symbol of \prod as stated in the (1), where n represents the hyperparameters, each parameter has m_i possible values.

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

With an increasing number of hyperparameters, the size of dimension space containing all of the hyperparameter configurations increases exponentially [12]. Due to the increasing in size of dimensionality, the implementation of hyperparameter tuning using handcrafted approaches is tedious, laborious, prone to errors, and consumes a lot of computing power [13]. In the context of real-world scenario, the process of objectives of optimization tends to be conflicting from one to another. Within this event, the implementation of multi objective optimization (MOO) are more relevant to be implemented to satisfy the requirement from both conflicting objectives.

As instances within the discipline of DL, the objectives can vary depending on the classification or the efficiency performances, where it could be in the form of accuracy, F1-score, loss function, size of the model, latency, and to name a few. Based on previous studies, various examples of MOO are being implemented into the discipline of DL, such as accuracy vs computational complexity [14], accuracy vs specificity vs sensitivity [15], and accuracy vs latency [16]. Contrary to that, this study will be implementing MOO, and the performance will be evaluated between three conflicting objectives of accuracy, F1- score and weight of the model by using random search. Theoretically, multi-objective optimization can be expressed as either a minimization or maximization problem, as shown in (2) [17]:

$$\begin{aligned} \text{Minimize: } \overline{F(\vec{x})} &= \{f_1(\vec{x}), f_2(\vec{x}), \dots, f_o(\vec{x})\} \\ \text{Maximize: } \overline{F(\vec{x})} &= \{f_1(\vec{x}), f_2(\vec{x}), \dots, f_o(\vec{x})\} \\ \text{Subject to: } g_i(\vec{x}) &\geq 0, i = 1, 2, \dots, m \\ h_i(\vec{x}) &= 0, i = 1, 2, \dots, p \\ lb_i \leq x_i &\leq ub_i, i = 1, 2, \dots, n \end{aligned} \quad (2)$$

Consider a vector, denoted as (\vec{x}) , which represents a collection of variables $(\vec{x} = x_1, x_2, x_3, x_4, x_{n-1}, x_n)$ relevant to the given problem, where n represents the quantity of variables, m denotes the count of inequality constraints, p signifies the number of equality constraints, lb_i denotes the lower boundary of the i -th variable, and ub_i denotes the upper boundary of the i -th variable. Based on the above formulation, there is a vector that store multiple variables which also referred to as parameters or decision

variables. This vector encapsulates all the relevant variables of the problem and is input into the objective function, which yield a numerical result.

2.2. Hyperparameter tuning techniques

In the current practice, the implementation of hyperparameter tuning is done in the traditional way, which involves human directly to manually tuning the hyperparameter. Manual tuning entails modifying hyperparameters through intuition or trial and error [18]. However, as depicted, this method is both time-consuming and may not yield optimal results as it relies on the expertise of the practitioner [19]. There is another widely use method for hyperparameter which act as an alternative to manual tuning, which is called grid search [20].

Grid search has become the predominant baseline optimization strategy for hyperparameter tuning [21]. The theoretical concept of grid search is to try all possible solution which leads to finding the most accurate solution [22] yet it will be afflicted by the curse of dimensionality [23]. Consequently, the effectiveness of grid search in locating the optimal solution within the search space depends on the dimension size where the increasing number of hyperparameters will contribute to increase the dimension size exponentially. Figure 1 illustrates the disparity of grid search and random search.

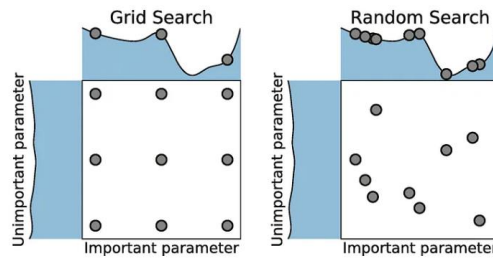


Figure 1. Grid search vs random search [24]

To address the limitation of the grid search, a promising approach is the random search technique. Contrary to grid search, random search sample hyperparameters value randomly within the searching space [24]. The different between grid search and random search are illustrated in Figure 1. The key advantage of random search lies on its simplicity and ease of implementation [24]. Unlike other optimization techniques, random search able to achieve improved outcomes through exploring a broader, albeit less appealing configuration space [25], can be automated and it does not require any gradient information or any other prior knowledge about the optimization landscape [26]. This makes it ideal for problems where the optimization landscape is complex and poorly understood. Within the scope of this study, hyperparameter tuning will be conducted using random search.

2.3. Workflow of the research

In the context of Figure 2, there are a few distinct phases, each of which plays a crucial role in this research. These phases include the process of data preparation and configuration, the process of hyperparameter optimization and the systematic process of analysis and conclusion. Figure 2 shows the flowchart of this research from the general perspective.

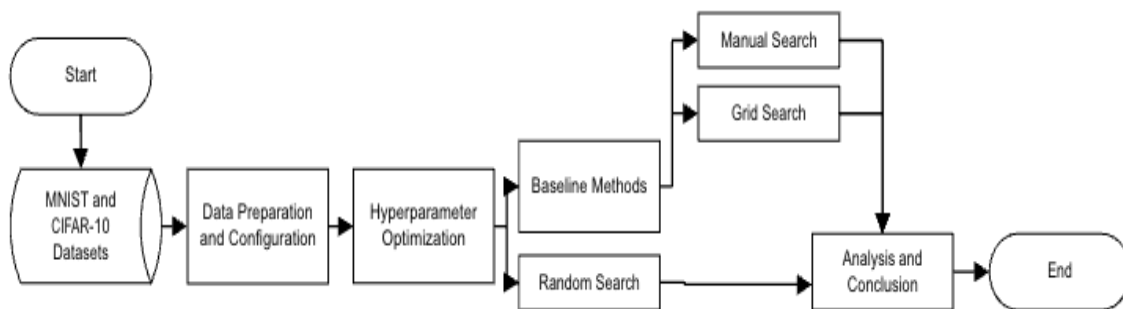


Figure 2. Schematic representation of the research methodology

2.3.1. Data collection

The data that will be used in this experiment is the benchmark datasets that are typically implemented in DL experiments. One of the most popular lightweight benchmark datasets for image classification is Modified National Institute of Standards and Technology (MNIST) which consists a dataset of 60,000 handwritten digit images for training and an additional 10,000 images for testing [27]. Each of the images in the MNIST dataset is the grayscale images with a resolution of 28×28 pixels. Notably, MNIST dataset represents ten classes which consists of digit 0 to 9. The other benchmark dataset that will be used in this experiment is Canadian Institute for Advanced Research-10 (CIFAR-10), which is also famously known for image classification. CIFAR-10 dataset comprises 60,000 red, green, and blue (RGB) images, each with a resolution of 32×32 pixels. These images are divided and represented into ten classes consisting of animals, and vehicles [28].

2.3.2. Data preparation and configuration

As referred to Figure 3, the process of data preparation and configuration consists of four crucial phases, starting from splitting the data. The process of splitting the data is to partition the dataset into training and testing subsets, with the former being utilized for model training and the latter to evaluate the model's performance. Subsequently, the data undergo normalization to ensure consistency in scale and distribution across features.

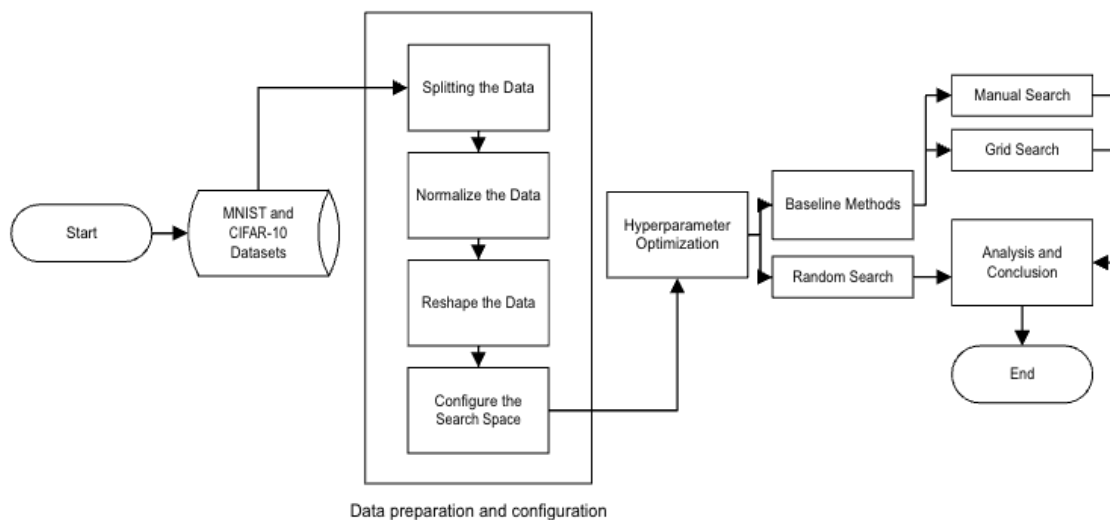


Figure 3. Flowchart for the process of data preparation and configuration

Following to the process of data normalization, the continued step involves the processing of reshaping the dataset to a new resolution of 64×64 pixels. This adjustment is necessary as it aligns with the subsequent design choice of reducing the kernel size in the network architecture, which precedes the application of max-pooling layer. By going through the process of reshaping the data, the model can effectively capture and extract meaningful features while ensuring compatibility with subsequent layers in the network. Finally, the search space is set up according to the hyperparameters outlined in Table 1. The dimension of each searching space is differ based on number of hyperparameters, and values of hyperparameters.

2.3.3. Hyperparameters

Hyperparameters are a set of parameters used for learning process during training and testing [29]. In this experiment, the choices of hyperparameters depend on the DL architecture due to the existence of extra convolutional or any other hidden layers. Based on the Table 1, the configuration of hyperparameters will be as:

Table 1 shows the hyperparameters used in multilayer perceptron (MLP), CNN, and AlexNet architectures for this experiment. The recorded hyperparameters in the Table 1 can be categorized into two groups, which are (1) hyperparameters for network architecture, which directly impact the parameters number and the model size, and (2) hyperparameters for learning structures, which associated with the learning structure, and does not impact on the number of parameters and final model size.

Table 1. The hyperparameters in different DL architecture (a) MLP, (b) CNN, and (c) AlexNet

Hyperparameters	Description	Range/values
LR	Learning rate	[1, 0.1, 0.01, 0.001, 0.0001]
NE	Number of epochs	[10, 20, 30, 40, 50]
BS	Batch size	[32, 64, 128, 256, 512]
OP	Optimizer	['SGD', 'RMSprop', 'Adagrad', 'Adadelata', 'Adam', 'Adamax', 'Nadam']
LR	Learning rate	[1, 0.1, 0.01, 0.001, 0.0001]
NE	Number of epochs	[10, 20, 30, 40, 50]
BS	Batch size	[32, 64, 128, 256, 512]
OP	Optimizer	['SGD', 'RMSprop', 'Adagrad', 'Adadelata', 'Adam', 'Adamax', 'Nadam']
LR	Learning rate	[1, 0.1, 0.01, 0.001, 0.0001]
NE	Number of epochs	[10, 20, 30, 40, 50]
BS	Batch size	[32, 64, 128, 256, 512]
OP	Optimizer	['SGD', 'RMSprop', 'Adagrad', 'Adadelata', 'Adam', 'Adamax', 'Nadam']
NF1	Number of filter (1)	[16, 32, 64, 96]
KS1	Kernel size (1)	[3, 4, 5, 6]
NF2	Number of filter (2)	[48, 64, 96, 128]
KS2	Kernel size (2)	[3, 4, 5, 6]
NF3	Number of filter (3)	[64, 96, 128]
KS3	Kernel size (3)	[3, 4, 5]
AC	Activation	['relu', 'elu']
LR	Learning rate	[1, 0.1, 0.01, 0.001, 0.0001]
NE	Number of epochs	[10, 20, 30, 40, 50]
BS	Batch size	[32, 64, 128, 256, 512]
OP	Optimizer	['SGD', 'RMSprop', 'Adagrad', 'Adadelata', 'Adam', 'Adamax', 'Nadam']
NF1	Number of filter (1)	[16, 32, 64, 96]
KS1	Kernel size 1	[3, 4, 5, 6]
NF2	Number of filter (2)	[48, 64, 96, 128]
KS2	Kernel size (2)	[3, 4, 5, 6]
NF3	Number of filter (3)	[64, 96, 128]
KS3	Kernel size (3)	[3, 4, 5]
NF4	Number of filter (4)	[64, 96, 128]
KS4	Kernel size (4)	[3, 4]
NF5	Number of filter (5)	[64, 96, 128]
KS5	Kernel size (5)	[3, 4]
AC	Activation	['relu', 'elu']

2.3.4. Hyperparameter tuning via random search

Hyperparameter tuning will be conducted using random search, and the performance will be evaluated and compared to the other baseline methods. The time constraints are applied to random search and other baseline methods and will serve as a stopping criterion after 24 hours. The procedural details of random search within this study are depicted in Figure 4.

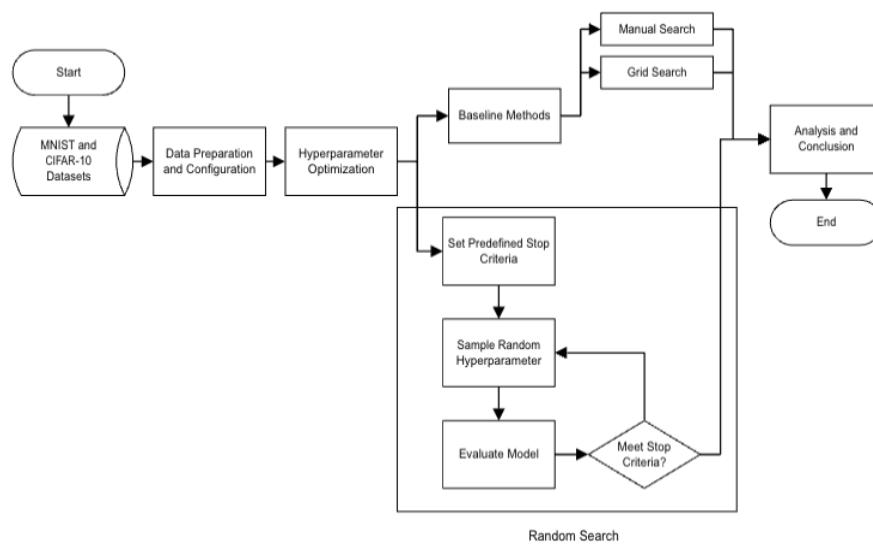


Figure 4. The random search process

As illustrated in Figure 4, the process in random search consists of three important elements, sample random hyperparameter, evaluate model, and check the stopping criteria to re-iterate the procedure. Once the stopping criteria is met, the result is recorded and will be analyzed to compare with the other baseline methods. In the end, the collective results for random search and other baseline methods contributing valuable insights to the research community.

2.3.5. Performance evaluation

The performance evaluation in this study will be assessed by using the trading off between accuracy, F1-score, and number of parameters. The evaluation of the classification metrics, specifically accuracy, can be formulated by utilizing the confusion matrix, which comprises four key components. The components involved are true positive (TP), true negative (TN), false positive (FP), and false negative (FN). The formula to measure accuracy is constructed from the elements stored within the confusion matrix. Table 2 presents the confusion matrix and demonstrates the calculation for accuracy and F1-score.

Table 2. The illustration of confusion matrix to calculate accuracy and F1-score

	Positive	Negative	
Positive	TP	FN	
Negative	FP	TN	
Recall	recision	Negative predictive value (NPV)	Accuracy
$\frac{TP}{TP+FN}$	$\frac{TP}{TP+FP}$	$\frac{TN}{TN+FN}$	$\frac{TN+TP}{TN+FN+TP+FP}$

In addition to accuracy, another classification metric that will be examined in this study is F1-score. F1-score can be defined as the measurement of the harmonic mean of precision and recall [30], offering a fair evaluation of the model's performance. F1-score can be measured by using the recall, and precision as stated in Table 2. Precision is the proportion of instances predicted as positive, while recall represents the ratio of correctly identified positive instances to the total number of actual positive instances. In (3) shows the formula to measure the F1-score.

$$F1\ score = \frac{2(precision \times recall)}{(precision + recall)} \quad (3)$$

Another performance metric that will be evaluated in this study is weight of the model, which will be determined in the form of parameters number. Theoretically, the higher the parameters number indicates the bigger model size as using similar mechanism in [31]. To assess performance from a multi-objective perspective, it is best to aggregate the three opposing criteria of accuracy, F1-score and weight of the model.

Due to the contradict direction of objectives, where the greater values are desired for the accuracy and F1-score, while a lesser number of parameters is preferred, the procedure of normalising and inverting the number of parameters is necessary. This is done to ensure that these indicators are in line with the same direction for each of the metrics, enabling a fair comparison and trade-off between two conflicting objectives. The formula for the weighted sum is (4):

$$\max_params = \max(P)$$

$$\min_params = \min(P)$$

$$inversed_normalized_params = (\max_params - Px) / (\max_params - \min_params)$$

$$weighted_sum = (F1_score * w1) + (Accuracy * w2) + (inversed_normalized_params * w3) \quad (4)$$

In the context of the (4), P refers to the parameters, while w refers to the performance metrics weight. As for this study, the priority for all of the performance metrics is equal, thus the value for $w1$, $w2$, and $w3$ are set to 1.

3. RESULT AND ANALYSIS

The performance evaluation of hyperparameter tuning is explored through random searches across different DL architectures, including MLP, CNN, and AlexNet on MNIST and CIFAR-10 datasets. The random search performance will be compared against two baseline methods namely (1) manual search, and (2) grid search. Based on the experiment, it is proven the process of manual search required expertise, experience, and

prone to biasness. With that being said, the alternative baseline methods which is grid search can be computationally expensive, particularly under the worst circumstances, in which the optimal hyperparameter configuration is positioned at the end of the sequential order.

Within the scope of this experiment, as depicted in Table 3 the dimension size of searching space containing hyperparameter configurations drastically increased as the number of hyperparameters increased. If given the time constraint of 24 hours, the process of grid search only realistic to be executed completely on MLP architecture only. Table 3 records the exploration of the hyperparameter values and search space dimension.

Table 3. The hyperparameter values and the search space dimension

DL architecture	Hyperparameters	Value	Num of combinations	Avg time each combination (s)		Est total time (hours)							
				MNIST	CIFAR-10	MNIST	CIFAR-10						
MLP	Epoch	[10, 20, 30, 40, 50]	980	27.03	30.30	7.35	8.24						
	Batch size	[32, 64, 128, 256, 512]											
	Learning rate	[1, 0.1, 0.01, 0.001, 0.0001]											
	Optimizer	['SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Adamax', 'Nadam']											
CNN	Epoch	[10, 20, 30, 40, 50]	2,268,000	69.73	90.65	43929.9	57109.5						
	Batch size	[32, 64, 128, 256, 512]											
	Learning rate	[1, 0.1, 0.01, 0.001, 0.0001]											
	Optimizer	['SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Adamax', 'Nadam']											
	Number of filter (1)	[16, 32, 64, 96]											
	Kernel size (1)	[3, 4, 5]											
	Number of filter (2)	[48, 64, 96, 128]											
	Kernel size (2)	[3, 4, 5]											
	Number of filter (3)	[64, 96, 128]											
	Kernel size (3)	[3, 4, 5]											
	Activation	['relu', 'elu']											
	AlexNet	Epoch						[10, 20, 30, 40, 50]	145,152,000	207.81	230.21	8,378,899.2	9.282,067.2
		Batch size						[32, 64, 128, 256, 512]					
		Learning rate						[1, 0.1, 0.01, 0.001, 0.0001]					
Optimizer		['SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Adamax', 'Nadam']											
Number of filter (1)		[16, 32, 64, 96]											
Kernel size (1)		[3, 4, 5, 6]											
Number of filter (2)		[48, 64, 96, 128]											
Kernel size (2)		[3, 4, 5, 6]											
Number of filter (3)		[64, 96, 128]											
Kernel size (3)		[3, 4, 5]											
Number of filter (4)		[64, 96, 128]											
Kernel size (4)		[3, 4]											
Number of filter (5)		[64, 96, 128]											
Kernel size (5)		[3, 4]											

Within the scope of this experiment, where the time constraint is 24 hours, the process of grid search can only be done affectively on MLP architecture, which is equivalent to 7.35 hours and 8.24 hours to

completely sample all hyperparameter configuration in the hyperparameter search space. Whereas, by implementing grid search on other DL architectures, it takes more than 24 hours to completely executed. Due to that, the probability of random search to perform better is higher if the optimal solution located beyond the sequential order explored by grid search in CNN, and AlexNet architecture. The finding of optimal hyperparameter configuration from multi-objective point of view using Pareto method for MLP, CNN, and AlexNet on MNIST dataset by using random search are plotted in Figures 5(a)-(c).

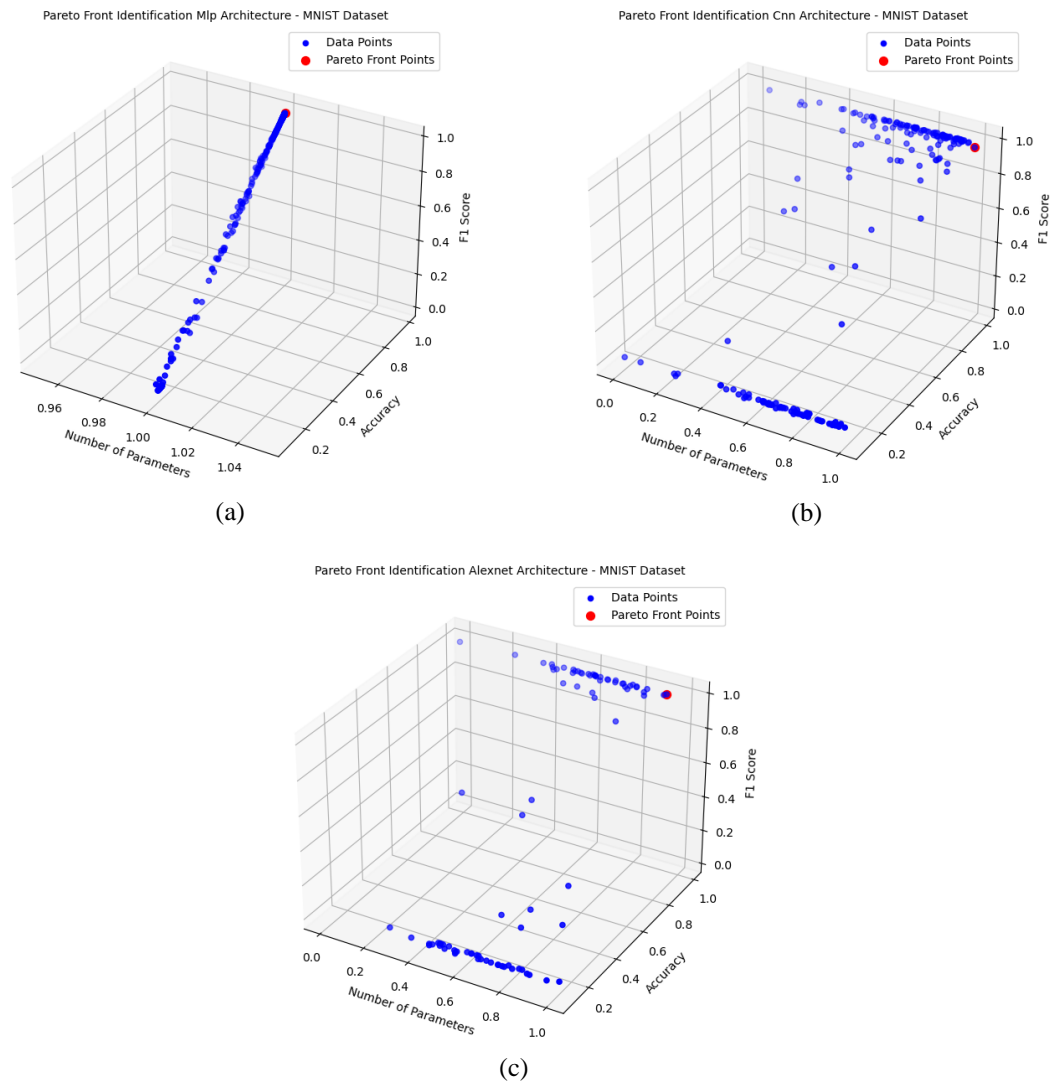


Figure 5. The Pareto identification for different architectures on MNIST dataset: (a) MLP, (b) CNN, and (c) AlexNet

Based on the graphical representation in Figure 5, the Pareto identification is marked red on each of the graphs according to Figure 5(a) MLP, Figure 5(b) CNN, and Figure 5(c) AlexNet. Within the context of the MNIST dataset, the Pareto identification for the MLP yields an accuracy of 0.9833, an F1-score of 0.9832, and normalized parameters equating to 1. In the realm of CNN, the Pareto identification manifests an accuracy of 0.9923, an F1-score of 0.9923, and normalized parameters holding a value of 1.0. Lastly, within the framework of the AlexNet architecture, the Pareto identification culminates in an accuracy of 0.9892, an F1-score of 0.9891, and normalized parameters measuring at 0.9287. Figures 6(a)-(c) shows the Pareto identification for MLP, CNN, and AlexNet on CIFAR-10 dataset.

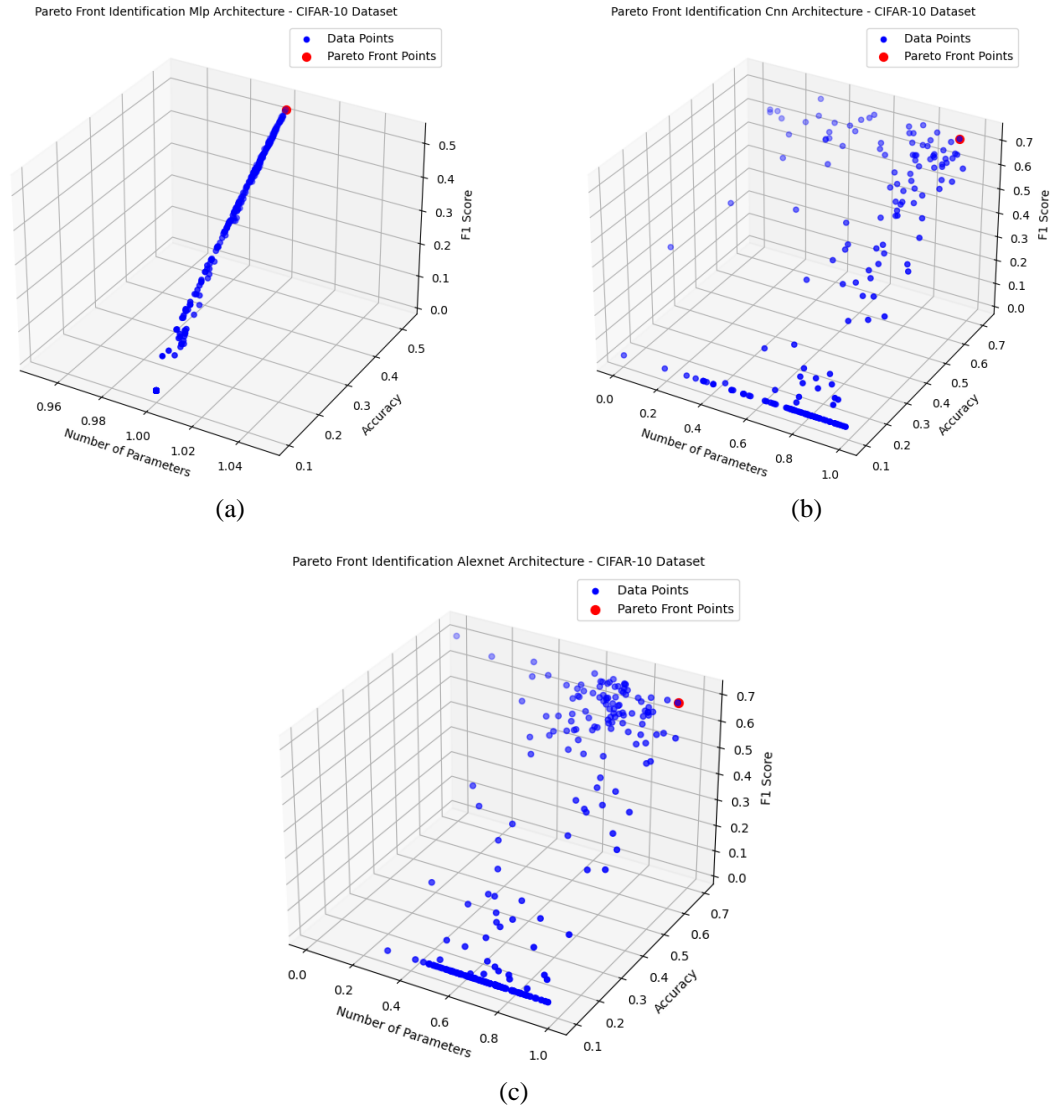


Figure 6. The Pareto identification for different architectures on CIFAR-10 dataset: (a) MLP, (b) CNN, and (c) AlexNet

According to Figure 6, the Pareto identification is marked red on each of the corresponding graphs for MLP denoted as Figure 6(a), CNN denoted as Figure 6(b), and AlexNet denoted as Figure 6(c). Within the context of the MNIST dataset, the Pareto identification for the MLP yields an accuracy of 0.5260, an F1-score of 0.5223, and normalized parameters equating to 1. Conversely in CNN, the Pareto identification manifests an accuracy of 0.7082, an F1-score of 0.7087, and normalized parameters holding a value of 0.9341. Lastly, within the framework of the AlexNet architecture, the Pareto identification culminates in an accuracy of 0.6834, an F1-score of 0.6811, and normalized parameters measuring at 0.9437. Tables 4 to 6 shows the recorded top-10 results for the process of hyperparameter tuning on MNIST dataset and various architectures (MLP, CNN, and AlexNet respectively). The columns involved in these tables are optimizer (OPT), learning rate (LR), batch size (BS), epochs (EP), activation (ACT), kernel 1 (K1), kernel 2 (K2), kernel 3 (K3), kernel 4 (K4), kernel 5 (K5), filter 1 (F1), filter 2 (F2), filter 3 (F3), filter 4 (F4), and filter 5 (F5).

Table 4. The results for hyperparameter tuning using random search on MLP architecture and MNIST dataset

OPT	LR	BS	EP	ACT	Accuracy	F1-score	Normalized params	Sum params	accuracy	F1-score
Adagrad	0.100	32	30	elu	0.9833	0.9832	1		2.9665	
Adadelata	1.000	64	40	relu	0.9823	0.9822	1		2.9645	
Adagrad	0.100	32	40	elu	0.9822	0.9821	1		2.9643	
Adagrad	0.100	32	40	elu	0.9817	0.9815	1		2.9632	
Adagrad	0.100	32	20	relu	0.9814	0.9812	1		2.9626	

Table 5. The results for hyperparameter tuning using random search on CNN architecture and MNIST dataset

K1	K2	K3	F1	F2	F3	OPT	LR	BS	EP	ACT	Accuracy	F1-score	Normalized params	Sum params accuracy F1-score
3	3	3	16	48	64	Adagrad	0.1	32	10	relu	0.9923	0.9923	1.000	2.9847
4	5	3	16	64	64	RMSprop	0.0001	64	30	elu	0.9898	0.9897	0.9681	2.9476
4	3	4	32	48	64	RMSprop	0.0001	256	30	elu	0.9870	0.9869	0.9665	2.9404
3	4	3	32	64	64	RMSprop	0.0010	512	50	elu	0.9918	0.9917	0.9470	2.9305
4	3	3	96	48	64	Nadam	0.0001	128	20	elu	0.9896	0.9895	0.9454	2.9245

Table 6. The results for hyperparameter tuning using random search on AlexNet architecture and MNIST dataset

K1	K2	K3	K4	K5	F1	F2	F3	F4	F5	ACT	OPT	LR	BS	EP	Accuracy	F1-score	Normalized params	Sum params accuracy F1-score
5	3	3	3	4	32	64	128	64	64	elu	Adagrad	0.0100	128	30	0.9892	0.9891	0.9287	2.9069
3	5	3	4	4	32	48	96	64	64	relu	Adadelta	1.0000	512	10	0.9851	0.9849	0.9182	2.8882
6	4	3	3	3	16	128	96	128	64	relu	Adagrad	0.1000	512	20	0.9900	0.9899	0.8431	2.8231
4	3	5	4	3	16	48	64	64	96	elu	Adam	0.0010	64	10	0.9779	0.9779	0.8365	2.7923
6	4	4	3	4	16	96	64	128	64	elu	Adadelta	0.0100	32	10	0.9673	0.9672	0.8432	2.7776

Based on Table 4 on MLP architecture, the best combination of hyperparameters is (OPT=Adagrad, LR=0.1, BS=32, EP=30, ACT=elu) which resulted to accuracy of 0.9833, F1-score of 0.9832 and inversed normalized params of 1.0. The sum of normalized params, accuracy and F1-score is 2.9665. Whereas, in CNN architecture, the best combination of hyperparameters is (K1=3, K2=3, K3=3, F1=16, F2=48, F3=64, OPT=Adagrad, LR=0.1, BS=32, EP= 10, and ACT=relu) which resulted to accuracy of 0.9923, F1-score of 0.9923 and normalized params of 1.0. The sum of total three metrics altogether is 2.9847. Meanwhile, the result for AlexNet architecture is accuracy of 0.9892, F1-score of 0.9891, and normalized params of 0.9287, where the sum of those metrics is 2.9069. The best combination that resulting to the result is (K1=5, K2=3, K3=3, K4=3, K5=4, F1=32, F2=64, F3=128, F4=64, F5=64, ACT=elu, OPT=Adagrad, LR=0.01, BS =128 and EP=30. Tables 7 to 9 shows the recorded top-10 results for the process of hyperparameter tuning on CIFAR-10 dataset and various architectures (MLP, CNN, and AlexNet respectively).

Table 7. The results for hyperparameter tuning using random search on MLP architecture and CIFAR-10 dataset

OPT	LR	BS	EP	ACT	Accuracy	F1-score	Normalized params	Sum params accuracy F1-score
Nadam	0.0001	32	50	elu	0.5260	0.5223	1	2.0483
Adagrad	0.0100	32	50	elu	0.5219	0.5190	1	2.0409
Adagrad	0.0100	32	40	elu	0.5140	0.5126	1	2.0266
Nadam	0.0001	128	50	relu	0.5162	0.5103	1	2.0265
Adagrad	0.0100	128	40	elu	0.5158	0.5103	1	2.0261

Table 8. The results for hyperparameter tuning using random search on CNN architecture and CIFAR-10 dataset

K1	K2	K3	F1	F2	F3	OPT	LR	BS	EP	ACT	Accuracy	F1-score	Normalized params	Sum params accuracy F1-score
3	3	5	32	48	64	Adamax	0.0010	64	40	relu	0.7082	0.7087	0.9341	2.3510
5	3	3	96	64	64	Adagrad	0.0100	32	20	elu	0.6982	0.6964	0.9062	2.3008
3	3	3	16	48	128	Adagrad	0.1000	32	40	relu	0.6735	0.6736	0.9535	2.3005
4	3	3	16	64	64	Adamax	0.0100	512	10	relu	0.6481	0.6462	1.0000	2.2943
3	4	4	16	64	64	Adagrad	0.0100	64	40	relu	0.6739	0.6754	0.9446	2.2939

Based on Table 7 on MLP architecture, the best combination of hyperparameters is (OPT=Nadam, LR=0.0001, BS=32, EP=50, ACT=elu) which resulted to accuracy of 0.5260, F1-score of 0.5223 and inversed normalized params of 1.0. The sum of normalized params, accuracy and F1-score is 2.9665. Whereas, in CNN architecture, the best combination of hyperparameters is (K1=3, K2=3, K3=5, F1=32, F2=48, F3=64, OPT=Adamax, LR=0.001, BS=64, EP=40, and ACT=relu) which resulted to accuracy of 0.7082, F1-score of 0.7087 and normalized params of 0.9341. The sum of total three metrics altogether is 2.3510. Meanwhile, the

result for AlexNet architecture is accuracy of 0.6834, F1-score of 0.6811, and normalized params of 0.9473, where the sum of those metrics is 2.3118. The best combination that resulting to the result is (K1=6, K2=4, K3=3, K4=3, K5=3, F1=32, F2=48, F3=96, F4=96, F5=64, ACT=relu, OPT=Adamax, LR=0.001, BS = 64, and EP=10.

Table 9. The results for hyperparameter tuning using random search on AlexNet architecture and CIFAR-10 dataset

K1	K2	K3	K4	K5	F1	F2	F3	F4	F5	ACT	OPT	LR	BS	EP	Accuracy	F1-score	Normalized params	Sum params accuracy F1-score
6	4	3	3	3	32	48	96	96	64	relu	Adamax	0.0010	64	10	0.6834	0.6811	0.9473	2.3118
6	3	3	3	4	64	96	64	96	64	elu	SGD	0.0100	32	20	0.6827	0.6816	0.9055	2.2698
4	4	3	3	3	16	64	64	64	64	elu	Adam	0.0001	32	50	0.6127	0.6118	1.0000	2.2245
5	5	4	3	3	64	96	64	96	64	relu	Nadam	0.0001	64	20	0.6796	0.6777	0.8421	2.1994
3	4	3	3	4	32	48	64	96	64	relu	Adagrad	0.1000	256	20	0.6174	0.6144	0.9480	2.1798

Based on the result as recorded in Tables 10 to 12, the results obtained from hyperparameter tuning across multiple DL architecture show the distinct patterns of performance for different optimization methods. As recorded, the result of random search appears to be standout performer consistently demonstrating superior results compared to manual search and grid search. However, on MLP architecture, grid search outperforms other methods of hyperparameter tuning due to the hyperparameter search space is small and manageable for grid search to completely execute the whole iteration within the time constraint given.

Table 10. Results of hyperparameter tuning by using MLP

MLP				
Hyperparameter tuning methods	Dataset	Accuracy	F1-score	Number of params
Manual search	MNIST	0.9813	0.9812	101,770
	CIFAR-10	0.5260	0.5223	394,634
Grid search	MNIST	0.9826	0.9829	101,770
	CIFAR-10	0.5084	0.5058	394,634
Random search	MNIST	0.9833	0.9832	101,770
	CIFAR-10	0.5051	0.5005	394,634

Table 11. Results of hyperparameter tuning by using CNN

CNN				
Hyperparameter tuning methods	Dataset	Accuracy	F1-score	Number of params
Manual search	MNIST	0.977	0.9769	82,970
	CIFAR-10	0.6631	0.6604	133,450
Grid search	MNIST	0.9907	0.9906	92,698
	CIFAR-10	0.6579	0.6578	133,466
Random search	MNIST	0.9923	0.9923	40,602
	CIFAR-10	0.7082	0.7087	94,202

Table 12. Results of hyperparameter tuning by using AlexNet

AlexNet				
Hyperparameter tuning methods	Dataset	Accuracy	F1-score	Number of params
Manual search	MNIST	0.975	0.974	17,572,650
	CIFAR-10	0.6373	0.6385	17,448,154
Grid search	MNIST	0.9876	0.9875	17,131,674
	CIFAR-10	0.6919	0.6904	17,638,378
Random search	MNIST	0.9892	0.9891	17,321,098
	CIFAR-10	0.6834	0.6811	17,296,602

4. CONCLUSION

Our study thoroughly explored the effectiveness of multi-objective (MO) hyperparameter tuning methods, particularly focusing on random search in comparison to manual and grid search techniques, using MNIST and CIFAR-10 datasets. Our findings demonstrate that random search generally outperforms other baseline methods, although grid search proves advantageous in scenarios with limited hyperparameter search spaces and time constraints. Notwithstanding the positive outcomes of random search, it suffers from certain

limitations, including inconsistency in results due to its random sampling approach, potential redundancy, and inefficiency. Addressing these limitations, we advocate for the exploration of advanced optimization techniques, such as genetic algorithms (GA), which capitalize on prior information to enhance consistency and efficiency in hyperparameter tuning. GA offers numerous advantages, including the ability to handle interdependencies among hyperparameters, explore broader search spaces, and converge rapidly to multiple optimal solutions. Furthermore, GA's parallelizability enhances computational efficiency. Future research endeavors should focus on evaluating the efficacy of GA for MO hyperparameter tuning, juxtaposed with other baseline methods like random search, grid search, and manual search. Such comparative analyses promise deeper insights into optimizing hyperparameter tuning for multi-objective problems, potentially elevating the performance and generalizability of machine learning models across diverse datasets and applications.

ACKNOWLEDGEMENTS

The Ministry of Higher Education Malaysia (MoHE) and Universiti Teknologi MARA provided support for this research through the Fundamental Research Grant Scheme (FRGS) (600-RMC/FRGS 5/3 (024/2021)).




REFERENCES

- [1] M. Tsuneki, "Deep learning models in medical image analysis," *Journal of Oral Biosciences*, vol. 64, no. 3, pp. 312–320, Sep. 2022, doi: 10.1016/j.job.2022.03.003.
- [2] D. Zeng *et al.*, "Deep Learning for Scene Classification: A Survey," 2021, doi: <https://doi.org/10.48550/arXiv.2101.10531>.
- [3] J. D. Choi and M. Y. Kim, "A sensor fusion system with thermal infrared camera and LiDAR for autonomous vehicles and deep learning based object detection," *ICT Express*, vol. 9, no. 2, pp. 222–227, Apr. 2023, doi: 10.1016/j.ict.2021.12.016.
- [4] Q. Wang, S. Bi, M. Sun, Y. Wang, D. Wang, and S. Yang, "Deep learning approach to peripheral leukocyte recognition," *PLoS ONE*, vol. 14, no. 6, p. e0218808, 2018, doi: 10.1371/journal.pone.0218808.
- [5] A. A. R. K. Bsoul, M. A. Al-Shannaq, and H. M. Aloqool, "Maximizing CNN Accuracy: A Bayesian Optimization Approach with Gaussian Processes," in *9th 2023 International Conference on Control, Decision and Information Technologies*, IEEE, Jul. 2023, pp. 2597–2602, doi: 10.1109/CoDIT58514.2023.10284448.
- [6] 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, Aug. 2023, doi: 10.1007/s10462-022-10359-2.
- [7] 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, Dec. 2020, doi: 10.1108/IJPC-09-2020-0130.
- [8] A. Aghaebrahimian and M. Cieliebak, "Hyperparameter tuning for deep learning in natural language processing," *CEUR Workshop Proceedings*, vol. 2458, 2019.
- [9] Sukanto, Hadiyanto, and Kurnianingsih, "KNN Optimization Using Grid Search Algorithm for Preeclampsia Imbalance Class," *E3S Web of Conferences*, vol. 448, p. 02057, Nov. 2023, doi: 10.1051/e3sconf/202344802057.
- [10] S. Sah, B. Surendiran, R. Dhanalakshmi, and M. Yamin, "Covid-19 cases prediction using SARIMAX Model by tuning hyperparameter through grid search cross-validation approach," *Expert Systems*, vol. 40, no. 5, Jun. 2022, doi: 10.1111/exsy.13086.
- [11] I. Jamaledin, R. El Ayachi, and M. Biniz, "An improved approach to Arabic news classification based on hyperparameter tuning of machine learning algorithms," *Journal of Engineering Research (Kuwait)*, vol. 11, no. 2, p. 100061, Jun. 2023, doi: 10.1016/j.jer.2023.100061.
- [12] M. A. Amirabadi, M. H. Kahaei, and S. A. Nezamalhosseini, "Novel suboptimal approaches for hyperparameter tuning of deep neural network [under the shelf of optical communication]," *Physical Communication*, vol. 41, p. 101057, Aug. 2020, doi: 10.1016/j.phycom.2020.101057.
- [13] L. Wen, X. Ye, and L. Gao, "A new automatic machine learning based hyperparameter optimization for workpiece quality prediction," *Measurement and Control (United Kingdom)*, vol. 53, no. 7–8, pp. 1088–1098, Aug. 2020, doi: 10.1177/0020294020932347.
- [14] S. C. Smithson, G. Yang, W. J. Gross, and B. H. Meyer, "Neural networks designing neural networks: Multi-objective hyperparameter optimization," in *IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers*, New York, NY, USA: ACM, Nov. 2016, pp. 1–8, doi: 10.1145/2966986.2967058.
- [15] 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.
- [16] 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, p. 104315, Sep. 2021, doi: 10.1016/j.engappai.2021.104315.
- [17] N. Gunantara, "A review of multi-objective optimization: Methods and its applications," *Cogent Engineering*, vol. 5, no. 1, pp. 1–16, Jan. 2018, doi: 10.1080/23311916.2018.1502242.
- [18] M. A. Setitra, M. Fan, and Z. E. A. Bensalem, "An efficient approach to detect distributed denial of service attacks for software defined internet of things combining autoencoder and extreme gradient boosting with feature selection and hyperparameter tuning optimization," *Transactions on Emerging Telecommunications Technologies*, vol. 34, no. 9, Sep. 2023, doi: 10.1002/ett.4827.
- [19] A. Kaplunovich, "Real Time Automatic Hyperparameter Tuning for Deep Learning in Serverless Cloud," 2020.
- [20] D. M. Belete and M. D. Huchaiah, "Grid search in hyperparameter optimization of machine learning models for prediction of HIV/AIDS test results," *International Journal of Computers and Applications*, vol. 44, no. 9, pp. 875–886, Sep. 2022, doi: 10.1080/1206212X.2021.1974663.
- [21] T. Yarally, L. Cruz, D. Feitosa, J. Sallou, and A. Van Deursen, "Uncovering Energy-Efficient Practices in Deep Learning Training: Preliminary Steps Towards Green AI," in *Proceedings-2023 IEEE/ACM 2nd International Conference on AI Engineering-Software Engineering for AI*, IEEE, May 2023, pp. 25–36, doi: 10.1109/CAIN58948.2023.00012.




- [22] M. Ogunsanya, J. Isichei, and S. Desai, "Grid search hyperparameter tuning in additive manufacturing processes," *Manufacturing Letters*, vol. 35, pp. 1031–1042, Aug. 2023, doi: 10.1016/j.mfglet.2023.08.056.
- [23] T. Yu and H. Zhu, "Hyper-Parameter Optimization: A Review of Algorithms and Applications," 2020, doi: <http://arxiv.org/abs/2003.05689>.
- [24] J. Bergstra and B. Yoshua, "Random search for hyper-parameter optimization yoshua bengio," *Journal of Machine Learning Research*, vol. 13, no. 10, pp. 281–305, 2012.
- [25] Y. A. Ali, E. M. Awwad, M. Al-Razgan, and A. Maarouf, "Hyperparameter Search for Machine Learning Algorithms for Optimizing the Computational Complexity," *Processes*, vol. 11, no. 2, p. 349, Jan. 2023, doi: 10.3390/pr11020349.
- [26] E. Elgeldawi, A. Sayed, A. R. Galal, and A. M. Zaki, "Hyperparameter tuning for machine learning algorithms used for arabic sentiment analysis," *Informatics*, vol. 8, no. 4, p. 79, Nov. 2021, doi: 10.3390/informatics8040079.
- [27] L. Deng, "The MNIST database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, Nov. 2012, doi: 10.1109/MSP.2012.2211477.
- [28] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images.(2009)," *Cs.Toronto.Edu*, pp. 1–58, 2009, [Online]. Available: <http://www.cs.utoronto.ca/~kriz/learning-features-2009-TR.pdf>
- [29] A. H. Victoria and G. Maragatham, "Automatic tuning of hyperparameters using Bayesian optimization," *Evolving Systems*, vol. 12, no. 1, pp. 217–223, Mar. 2021, doi: 10.1007/s12530-020-09345-2.
- [30] 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, p. 102477, Apr. 2021, doi: 10.1016/j.bspc.2021.102477.
- [31] 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 (Switzerland)*, vol. 12, no. 9, p. 1962, Apr. 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 is readily accessible and welcomes professional inquiries at his email address. 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 AI, 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 at the 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 include AI, evolutionary algorithms, DL, ML, and image processing. She can be contacted at email: shafaf2429@uitm.edu.my.