International Journal of Electrical and Computer Engineering (IJECE)
Vol. 13, No. 2, April 2023, pp. 2167~2176
ISSN: 2088-8708, DOI: 10.11591/ijece.v13i2.pp2167-2176

Automated-tuned hyper-parameter deep neural network by using arithmetic optimization algorithm for Lorenz chaotic system

Nurnajmin Qasrina Ann[1], Dwi Pebrianti[2], Mohammad Fadhil Abas[1], Luhur Bayuaji[3]
[1]Faculty of Electrical and Electronics Engineering Technology, Universiti Malaysia Pahang, Pahang, Malaysia
[2]Mechanical Engineering (Aerospace), International Islamic University Malaysia, Selangor, Malaysia
[3]Faculty of Computing, Universiti Malaysia Pahang, Pahang, Malaysia

Article Info

Article history:

Received May 11, 2022
Revised Sep 20, 2022
Accepted Oct 25, 2022

Keywords:

Arithmetic optimization algorithm
Automated-tuned system
Deep neural network
Lorenz chaotic system
Optimization

ABSTRACT

Deep neural networks (DNNs) are very dependent on their parameterization and require experts to determine which method to implement and modify the hyper-parameters value. This study proposes an automated-tuned hyper-parameter for DNN using a metaheuristic optimization algorithm, arithmetic optimization algorithm (AOA). AOA makes use of the distribution properties of mathematics' primary arithmetic operators, including multiplication, division, addition, and subtraction. AOA is mathematically modeled and implemented to optimize processes across a broad range of search spaces. The performance of AOA is evaluated against 29 benchmark functions, and several real-world engineering design problems are to demonstrate AOA's applicability. The hyper-parameter tuning framework consists of a set of Lorenz chaotic system datasets, hybrid DNN architecture, and AOA that works automatically. As a result, AOA produced the highest accuracy in the test dataset with a combination of optimized hyper-parameters for DNN architecture. The boxplot analysis also produced the ten AOA particles that are the most accurately chosen. Hence, AOA with ten particles had the smallest size of boxplot for all hyper-parameters, which concluded the best solution. In particular, the result for the proposed system is outperformed compared to the architecture tested with particle swarm optimization.

Corresponding Author:

Mohammad Fadhil Abas
Faculty of Electrical and Electronics Engineering Technology, Universiti Malaysia Pahang
26600 Pekan, Pahang Malaysia
Email: mfadhil@ump.edu.my

1. INTRODUCTION

Deep neural networks (DNNs) are very reliant on their parameterization. DNNs, like any machine learning model, have a number of hyper-parameters such as dropout rate and hidden layer number. To obtain excellent model training, these hyper-parameters must be appropriately configured. It takes a team of experts to figure out which method to use and how to change the hyper-parameters values. The requirement for developing automated methods for determining hyper-parameters is critical as DNN designs get exceedingly complex, rendering the trial-and-error method impractical.

Deep learning algorithms generate results based on available data rather than human-defined rules. When sufficient data and a well-implemented framework are available, these algorithms perform well in image processing, pattern recognition, and autonomous driving. However, certain elements dependently modify for a deep learning algorithm in the training process called hyper-parameters. A hyper-parameter is a constant that is not updated by training. However, it affects the algorithm [1]. Unlike the value of parameters such as weights learned during the training process [2]. Compared to the traditional machine learning models, DNNs are susceptible to the choice of hyper-parameters [3].

For years, the literature has demonstrated that hyper-parameter selection algorithms perform on par with human experts and even outperform them in certain situations. They remain, however, challenging to implement because of their computational difficulty in practice. This problem is solvable with a robust optimization algorithm and capable hardware. Nowadays, taming hyperparameters in deep learning systems has been challenging. Mai *et al.* [4] claimed that hyper-parameter compatibility must be taken into consideration when designing potential scalable deep learning systems in order to manipulate hyper-parameters. During training, the model’s hyperparameters must be adjusted to fit within the loss space, which employs a declining learning rate to capture sharp minima [5], improving accuracy. Next is the adaptive tuning of hyper-parameters that can schedule change-over-time. Other than that, a deep learning system should be adapted to maximize the utilization of expensive graphical processing units (GPUs) resources.

The selection of hyperparameters can be viewed as an optimization issue in which the objective is to maximize the accuracy of fitness functions while minimizing the inaccuracy of cost functions [6]. Manually fine-tuning these hyper-parameters is a challenging task [7]. Hence, many studies implemented automated hyper-parameter tuning in the literature to solve the problem. The metaheuristics optimization algorithms approach is the most popular technique nowadays, compared to random search [8], [9], grid search [10], [11], and Bayesian optimization [3], [12]–[15] as this method always gives an accurate and efficient performance.

The hyper-parameters involved in this experiment; are hidden layer number and size, optimizers, loss function, activation function, dropout, and validation split with different values each. Alibrahim and Ludwig [16] compared the performance of the grid search algorithm, Bayesian algorithm, and genetic algorithm (GA) to select the ideal hyper-parameter values for a neural network model. The algorithms are compared using several evaluation metrics, including accuracy and computational time. The GA outperformed the grid search followed by the Bayesian approach, as stated in the paper.

Wang *et al.* [17] developed a unique particle swarm optimization (PSO) version called cPSO-CNN for optimizing the setting of architecture based convolutional neural network (CNN) hyper-parameters. The update equations for cPSO-CNN have been changed to account for CNN’s hyper-parameter features. It is nearly impossible to manually tune a CNN to a near-optimal hyper-parameter configuration at a reasonable time consuming, which impedes the adaptation of CNNs to a variety of real-world issues. As seen by the cPSO-CNN curve, throughout the generations, it maintains a great capacity for investigation via a succession of free-falling decreases that culminate in an impressive 14.8 percent at the conclusion, significantly less than that of other PSOs.

Additionally, the most widely used metaheuristic algorithm, PSO, intelligently investigates the possible solutions, allowing DNNs with a minimal topology to perform competitively on the MNIST dataset [18]. The study demonstrated that limited DNNs optimized using PSO could accurately classify CIFAR-10. Additionally, PSO raises the performance of pre-existing DNN designs. PSO is beneficial for automating the selection of hyperparameters based on statistical results. Additionally, PSO demonstrated highly desirable scalability properties.

Mohakud and Dash [19] used a grey wolf optimization (GWO) algorithm to optimize the hyperparameters in a CNN architecture, which was then used to deduce the skin cancer class. The effectiveness of GWO performance was compared to that of PSO and GA-based hyperparameter optimized CNNs when applied to the International Skin Imaging Collaboration’s multi-class data set of skin lesions. The simulation results indicate that the proposed algorithm outperforms PSO and GA by 4% and 1%, respectively. The testing loss realized 39.2% and 15% less than PSO and GA, respectively.

In addition, Hossain *et al.* [20] proposed a framework comprised of deep learning models, an optimization algorithm, and a technique for decomposing data. The framework is being introduced to increase the forecasting accuracy for ultra-short-term wind energy generation. When the forecasting models were compared to the baseline models, it was determined that the suggested model is resistant to overfitting. The model’s hyper-parameters are tuned using the GWO technique to determine the optimal values. By merging the optimization approach with deep learning, the prediction accuracy significantly improved.

Furthermore, the hyperparameters of deep hybrid learning are tuned using the Harris hawks optimization algorithm to optimize the model’s performance [21]. The hybrid deep learning model comprises convolutional layers, gated recurrent units, and a fully connected neural network. It is used to forecast the very short-term wind energy generation from Australia’s Boco Rock Wind Farms. The study compares the proposed model to other deep learning models such as gate recurrent unit networks (GRU), long short-term memory networks (LSTM), recurrent neural networks (RNN), and neural network (NN) that are not automatically tuned using an optimization algorithm to demonstrate the proposed model’s effectiveness. Thus, the proposed model outperformed these other advanced models in terms of error minimization, thereby improving wind energy forecasting accuracy. Additionally, Mahdaddi *et al.* [22] presented an evolutionary method (EA) framework, namely the differential evolution (DE) algorithm, for determining the ideal configuration for the proposed model's hyper-parameters. The suggested model, CNN-attention-based bidirectional long short-term memory network (CNN-AbiLSTM), was compared using a manually built deep learning model and an EA-based hyper-parameter optimization technique. This comparison demonstrates that using EAs to determine an example quality configuration for the proposed model's hyper-parameters outperformed benchmark models manually parameterized according to the two evaluation procedures.

On the other hand, few researchers implemented Bayesian optimization [13] to fine-tune the deep convolutional neural network (DCNN) for image classification. The trained DCNN achieved an accuracy of 96.05%. This method, in particular, does not guarantee that the optimal solution will be found. However, it can find compared to grid search, good results can be achieved with far less computational time [12].

In this study, the arithmetic optimization algorithm (AOA) is implemented to fine-tune the hyper-parameters for the DNN model. AOA is mathematically modeled and implemented to optimize processes across a broad range of search spaces. The dataset used was Lorenz chaotic system [23]. This study aims to improve the effectiveness and accuracy of the DNN model for chaotic systems. As for the comparison purpose, AOA will be compared with PSO algorithm in terms of convergence curve and hyper-parameter combination.

The remainder of this essay is organized as follows: section 2 discusses pertinent literature for the study. Following that, section 3 discusses the AOA and the development of a hyper-parameter tuning framework as the research methodology. Section 4 discusses the experimental design, dataset, results, and analysis. Finally, section 5 brings the paper to a close-by outlining future recommendations.

2. METHOD

This subtopic discussed the methodology implemented for this study. The first section explained how the AOA works, including equations and a flowchart. Then, the development of the proposed hyper-parameter tuning framework is introduced.

2.1. The arithmetic optimization algorithm

The AOA is a new metaheuristic method that exploits the distribution behavior of the major arithmetic operators in mathematics, such as multiplication (M), division (D), addition (A), and subtraction (S) [24]. AOA is an arithmetically designed and applied technique for optimizing processes across a wide range of problem domains.

The solutions, X, are created randomly during the initialization phase, and the best solution acquired in each iteration is considered the best-obtained solution. Prior to initiating the AOA, utilizing the math optimizer accelerated (MOA) function, X should select the search space, whether exploration or exploitation. The coefficient is calculated by (1),

$$MOA(C_{Iter}) = Min + C_{Iter} \times (\frac{Max-Min}{M_{Iter}}) \tag{1}$$

where $MOA(C_{iter})$ Denotes the function value at the tth iteration. C_{iter} denotes current iteration, which is between 1 and the maximum iteration, M_{iter}. Min and Max denote the minimum and maximum values of the accelerated function, respectively.

The mathematical evaluations using either D or M operators obtained strong values or decisions based on the arithmetic operators, leading to the mechanism of exploration search. AOA's exploration operators randomly examine the search area across multiple regions and try to discover an ideal solution using two primary search strategies: D search and M search. When the value of a random number, r_1 is larger than the MOA value, then this search phase is started.

After that, r_2 value (also random number) conditioned by $r_2 < 0.5$, will employ the first operator D while the other operator M is neglected until this operator finishes its current task. It also works otherwise. The position updating equations for the exploration parts in this study are as (2),

$$x_{ij}(C_{iter}+1) = \begin{cases} best(x_j) \div (MOP+\epsilon) \times \left((UB_j - LB_j) \times \mu + LB_j\right), r_2 < 0.5 \\ best(x_j) \times MOP \times \left((UB_j - LB_j) \times \mu + LB_j\right), otherwise \end{cases} \tag{2}$$

where $x_i(C_{iter}+1)$ is the i^{th} solution in the next iteration, $x_{ij}(C_{iter})$ is the j^{th} position of the ith solution at the current iteration, and $best(x_j)$ represent the j^{th} position in the best-obtained solution so far. ϵ is the small integer number while UB_j and LB_j denote the upper and lower bound value at the jth position, respectively. Lastly, μ is a control parameter to adjust the search process, fixed equal to 0.5.

$$MOP(C_{Iter}) = 1 - \frac{C_{Iter}^{1/\alpha}}{M_{Iter}^{1/\alpha}} \tag{3}$$

where math optimizer probability (MOP) is a coefficient, $MOP(C_{Iter})$ denotes the function value at the tth iteration, and C_{Iter} denotes the current iteration and M_{Iter} denotes the maximum iteration number. α is a sensitive parameter and defines the exploitation accuracy over the iterations, and it is fixed to 5.

Mathematical calculations are employing either S or A produced dense results for the AOA exploitation approach by reference to the exploitation mechanism. The AOA exploitation operators thoroughly investigate the search area across numerous dense locations and approach finding a better solution using two distinct search strategies as depicted in (4). Furthermore, these operators aid the exploitation stage through improved communication.

$$x_{ij}(C_{Iter}) = \begin{cases} best(x_j) - MOP \times \left((UB_j - LB_j) \times \mu + LB_j\right), r_3 < 0.5 \\ best(x_j) + MOP \times \left((UB_j - LB_j) \times \mu + LB_j\right), otherwise \end{cases} \tag{4}$$

The settings are intended to generate a random value at each iteration in order to maintain exploitation not only during the initial iteration but also throughout the final iteration. In this phase, the first operator, Subtraction (S), is constrained by r_3 0.5, and the other operator A is ignored until this operator completes its current work and vice versa. This portion of the search is advantageous in cases of minimal local optima, particularly in the final iteration. Figure 1 depicts the clear and thorough AOA process.

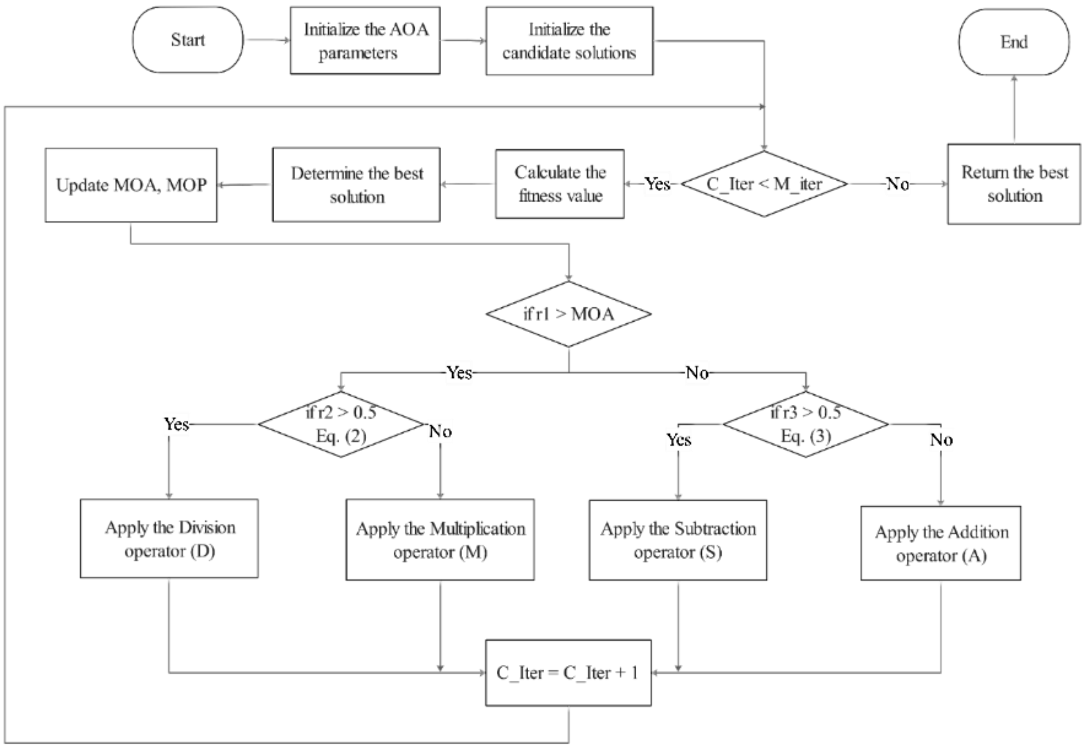
Start
Initialize the AOA parameters
Initialize the candidate solutions
End
Update MOA, MOP
Determine the best solution
Calculate the fitness value
Yes
C_Iter < M_iter
No
Return the best solution
if r1 > MOA
Yes
No
if r2 > 0.5 Eq. (2)
Yes
No
if r3 > 0.5 Eq. (3)
Yes
No
Apply the Division operator (D)
Apply the Multiplication operator (M)
Apply the Subtraction operator (S)
Apply the Addition operator (A)
C_Iter = C_Iter + 1

Figure 1. Flowchart of AOA

3.2. Hyper-parameter tuning framework

This study proposed an automated-tuning hyper-parameters technique based on a DNNs model for Lorenz chaotic system by AOA. The flowchart in Figure 2 is the proposed method for parameter tuning for DNNS by using AOA. Three DNNs hyper-parameters are tuned to obtain the best combinations concerning the accuracy value from the learning process of DNNs.

The goal of hyper-parameter tuning is to find the optimal combination while optimizing the model's accuracy. For this study, there is no specific fitness function for AOA. However, the solutions from the optimization algorithm will be trained automatic and directly in the DNNs model, refer to Figure 3, so that, the developed model is accurate and robust. The process, on the other hand, is the most time-consuming part.

$$\max DNN(H, W, D) \tag{5}$$

where DNN is the deep learning model, H is defined as hyper-parameters that need to be tuned, W indicates the weights of hyper-parameters, and D is referred to as datasets.

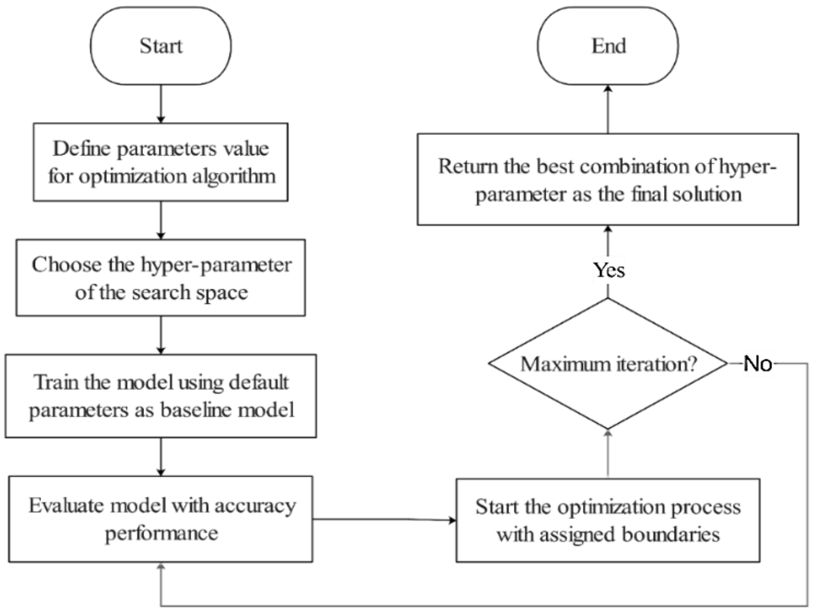
Start
Define parameters value for optimization algorithm
Choose the hyper-parameter of the search space
Train the model using default parameters as baseline model
Evaluate model with accuracy performance
Start the optimization process with assigned boundaries
Maximum iteration?
No
Yes
Return the best combination of hyper-parameter as the final solution
End

Figure 2. Proposed method for parameter tuning

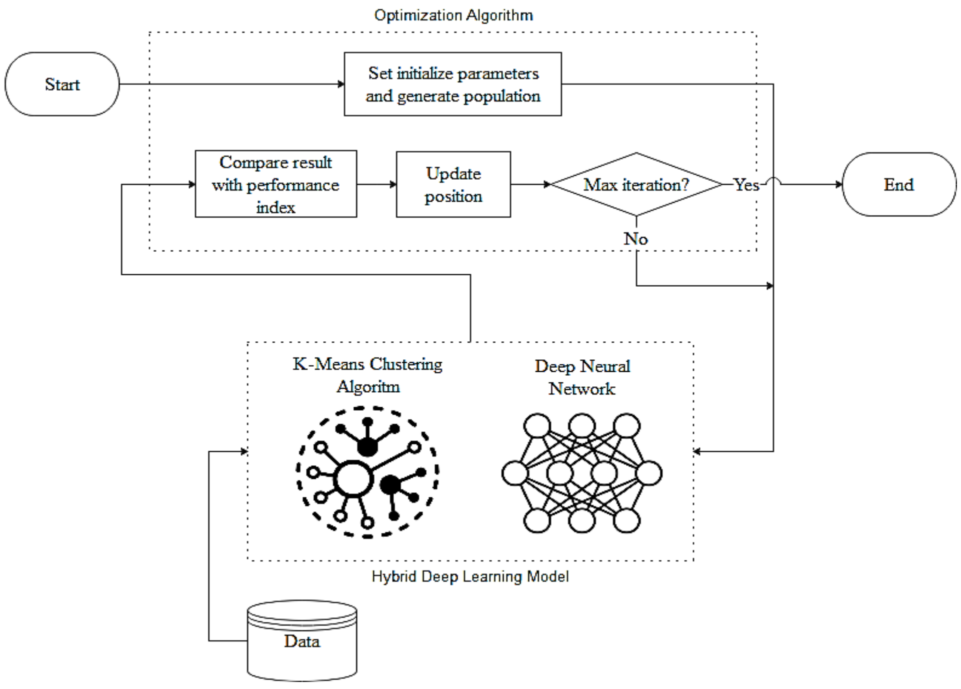
Optimization Algorithm
Start
Set initialize parameters and generate population
Compare result with performance index
Update position
Max iteration?
Yes
No
End
K-Means Clustering Algoritm
Deep Neural Network
Hybrid Deep Learning Model
Data

Figure 3. Proposed framework for tuning hyper-parameter based on AOA

3. RESULTS AND DISCUSSION

The experimental setup, dataset, result, and discussion of the study findings have been explained in this subtopic. The experimental study consists of two main experiments. Firstly, it verifies how the value of the particles affects the AOA capabilities to find the best solution and compares with the PSO with the different values of particles. The second part is about the descriptive statistics analysis to make sure the chosen model is proficient in terms of accuracy on the test dataset and consistency in the values of hyper-parameters.

3.1. Experimental setup

The hyper-parameter selection by AOA and training of DNN implemented in MATLAB R2021a. The experiments run on the following CPU hardware: Intel® Core™ i7-8700 Processor (3.20 GHz), RAM: 16GB, GPU: NVIDIA GeForce GTX 1060 3GB. Table 1 shows the hyper-parameters of DNNs involved in automated tuning using AOA and their ranges (boundary or selection) during the optimization and training process. For optimizer, there are three different selections which are stochastic gradient descent with momentum (SGDM), root-mean-square propagation (RMSProp), and adaptive moment estimation (ADAM). The last column of that table shows the expected value for each hyper-parameter when the algorithm is trained independently. Furthermore, Table 2 tabulates the parameter value of AOA and PSO, which are the number of iterations, number of particles, α, μ, c_1, c_2, w_{max} and w_{min}. Using a trained DNN model, the fitness function derived from this study is calculated and validated by the correctness of the testing data set.

Table 1. Hyper-parameter of DNN for automated tuning

Hyper-parameter	Range	Expected Value
Number of neurons for first hidden layer	[1, 100]	48
Optimizers	[SGDM, RMSProp, ADAM]	ADAM
Number of epochs	[10, 200]	200

Table 2. Parameters of AOA and PSO

Variable	Speed (rpm)	Power (kW)
AOA	α	5
	μ	0.5
PSO	c_1, c_2	2
	maximum weight, w_{max}	0.9
	minimum weight, w_{min}	0.2
Both	no. of particles	4, 10, 15
	no. of iteration	50

3.2. Dataset

In this study, the multi-class classification is focused on and utilizes the real dataset of the Lorenz chaotic system. Barrio *et al*. [23] proposed this dataset as a benchmark database of extremely precise numerically calculated and validated initial conditions for periodic orbits in the Lorenz model. The initial conditions of the periodic orbits and intervals of size 10,100 are calculated with 1,000 digits of precision using high-precision arithmetic and numerous approaches that demonstrate each orbit's existence. The dataset utilized in this study is quite huge, so it is recommended to split the dataset [25] into 70%, 15%, and 15% training, testing, and validation datasets.

3.3. Impact of the particle size on AOA

Refer to Figure 4, the graphs compare two meta-heuristic algorithms with different particles. Both PSO and AOA algorithms converge throughout the iterations until they reach the maximum iteration. Based on the convergence curve, for ten particles and 15 particles of AOA and PSO, respectively, the perfect fitness value is achieved. It showed that PSO needs more particles to give better accuracy. The exploration and exploitation for AOA are balanced, although the number of particles required is low. The convergence curve shows the best fitness function to iterations for each run. The convergence curves by AOA visualized that there are no distinct advantages in the first iterations. AOA distributes solutions over multiple local research areas rather than accumulating positions in a particular local area [24]. However, AOA can significantly reduce its dependency on local search areas and has competitive experience with global search areas.

Both AOA and PSO algorithms have experimented with a similar set of particles values. In Table 3, the display result is retrieved from the DNN architecture with the different number of particles together with a comparison to AOA and PSO. With 50 as the maximum iteration, the computational time and best hyper-parameter combination are recorded.

PSO algorithm is straightforward. The hyper-parameter tuning process is a tightrope walk to balance underfitting and overfitting. The optimization and learning process of the DNNs model produces a combination of three hyper-parameters investigated in this study. The hyper-parameters involved are the number of hidden neurons, the number of epochs, and the type of optimizers trained dependently on each other and gave the best combination of hyper-parameter of the DNN model. The chosen number of hidden neurons, number of epochs, and type of optimizers are 46, 200, and ADAM optimizers, respectively, which is ten particles of AOA. Thus, AOA with ten particles outperformed by obtaining the highest accuracy based on the test dataset and the closest expected value of the hyper-parameter combination presented in Table 1. In particular, AOA takes longer to compute the algorithm than PSO in terms of computational time.

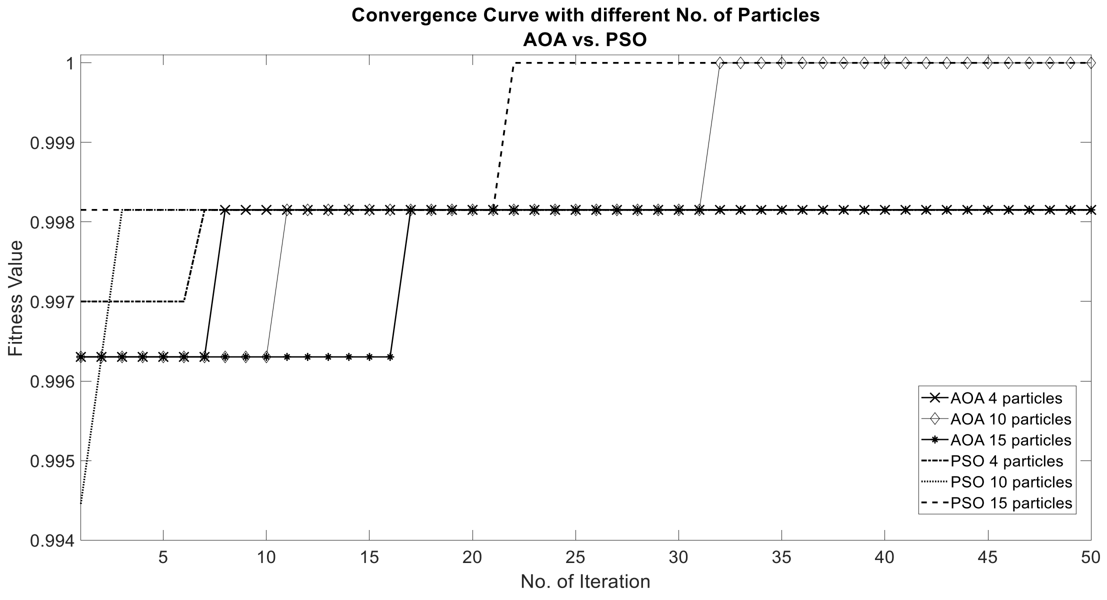

Figure 4. AOA and PSO convergence curve for different no. of particles

Table 3. Comparison of AOA and PSO with different numbers of particles

Algorithm	No. of particles	Time (s)	Best Hyper-Parameter Combination	Accuracy on Test Dataset
AOA	4	12827	[97, 162, ADAM]	0.9982
	10	33167	[46, 200, ADAM]	1.0000
	15	50988	[23, 180, ADAM]	0.9982
PSO	4	12047	[57, 178, RMSProp]	0.9982
	10	20496	[37, 177, RMSProp]	0.9982
	15	28475	[35, 107, ADAM]	1.0000

3.4. Descriptive statistics analysis on hyper-parameter selection by using arithmetic optimization algorithm

In the study, analysis of fitness value is not enough, as the experiment involves selecting hyper-parameters in DNN. Three hyper-parameters tested are the number of hidden neurons, epoch number, and type of optimizer. Therefore, boxplots helped visualize the features of numerical data through 10 runs dependently.

Figures 5 to 7 show the three hyper-parameters, the number of hidden neurons, epoch number, and type of optimizer, respectively. The boxplots represented the standard deviation of the data obtained from the simulations regarding the size of the boxplot. The most consistent mean data is ten particles of AOA for all three hyper-parameters based on the boxplots.

The 4 and 15 particles in boxplot analysis give the broad range value for the number of hidden neurons hyper-parameter, as shown in Figure 5. In addition, Figure 6 represents the boxplot for the epoch number needed to complete training for the developed model. The most considerable size of the boxplot is 16 particles, followed by four particles and ten particles having the smallest size of the boxplot. Figure 7 illustrates that ten particles are consistent at the 'ADAM' type of optimizer while 15 particles have outlier data. The optimizers obtained by four particles are 'ADAM' and 'RMSProp'. Thus, the result shown in this analysis is matched to the best hyper-parameter combination and the highest accuracy on the test dataset.

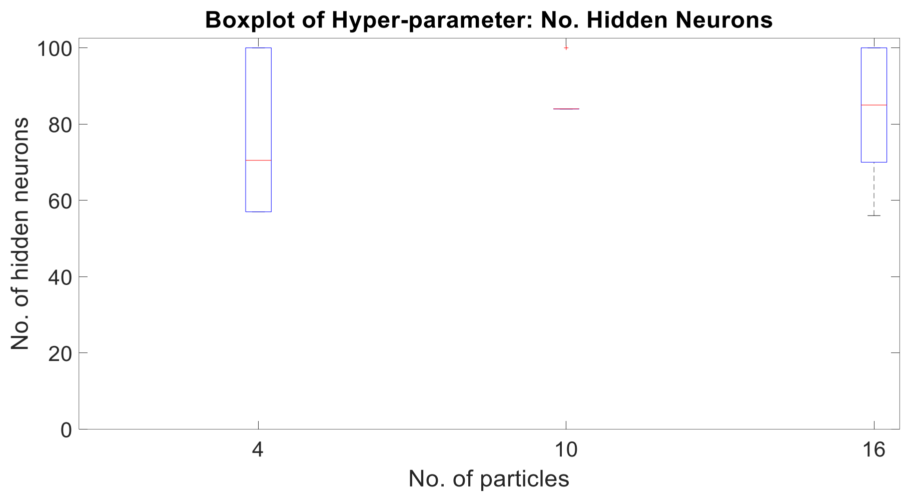

Figure 5. Boxplot for no. of hidden neurons

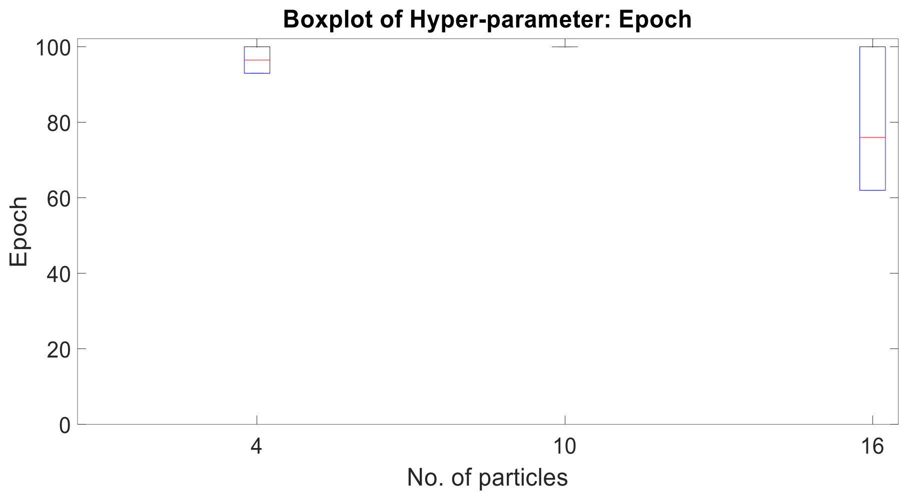

Figure 6. Boxplot for number of epochs

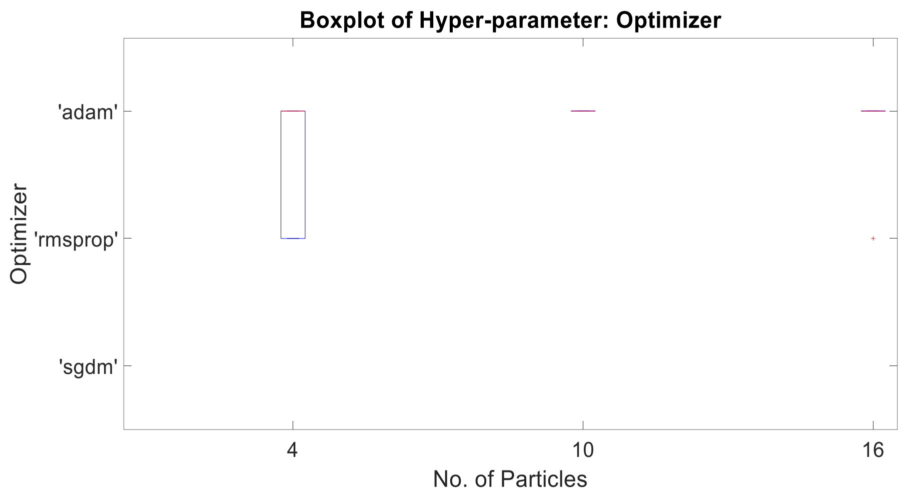

Figure 7. Boxplot for optimizers

4. CONCLUSION

To sum up, this study is presented the automated-tuned hyper-parameter of DNNs by using AOA for Lorenz chaotic dataset. A novel model architecture proposes solving the hyper-parameter problems in DNNs, an anchor for a deep learning model. The optimization and learning process of the AOA and DNNs model produced the best combination of three hyper-parameters: number of hidden neurons, epoch number, and type of optimizer, investigated in this study. In particular, the result showed that ten particles of AOA outperformed PSO with 15 particles in achieving the highest accuracy value, which is one on the test dataset. The boxplot analysis proved the consistency value of each hyper-parameters for the standard deviation value with ten independent runs. The boxplot statistical analysis ensures that the best number of particles is chosen for the trained model. Hence, AOA with ten particles had the smallest size of boxplot for all hyper-parameters, which concluded the best solution among all. For future works, after the chaotic system, the hyper-chaotic system will be experimented with using this model architecture. The experiment is essential for further real-world application, mainly in the image cryptosystems field.

ACKNOWLEDGEMENTS

The work presented in the paper has been supported by Universiti Malaysia Pahang Post Graduate Research Scheme PGRS2003205.

REFERENCES

[1] J.-H. Han, D.-J. Choi, S.-U. Park, and S.-K. Hong, "Hyperparameter optimization using a genetic algorithm considering verification time in a convolutional neural network," *Journal of Electrical Engineering & Technology*, vol. 15, no. 2, pp. 721–726, Mar. 2020, doi: 10.1007/s42835-020-00343-7.

[2] R. Pydipaty, J. George, K. Selvaraju, and A. Saha, "Improving the performance of deep learning for wireless localization," *Prepr. arXiv2006.08925*, Jun. 2020, [Online]. Available: http://arxiv.org/abs/2006.08925.

[3] H. Cho, Y. Kim, E. Lee, D. Choi, Y. Lee, and W. Rhee, "Basic enhancement strategies when using Bayesian optimization for hyperparameter tuning of deep neural networks," *IEEE Access*, vol. 8, pp. 52588–52608, 2020, doi: 10.1109/ACCESS.2020.2981072.

[4] L. Mai, A. Koliousis, G. Li, A. Brabete, and P. Pietzuch, "Taming hyper-parameters in deep learning systems," *ACM SIGOPS Operating Systems Review*, vol. 53, no. 1, pp. 52–58, Jul. 2019, doi: 10.1145/3352020.3352029.

[5] E. Hoffer, I. Hubara, and D. Soudry, "Train longer, generalize better: closing the generalization gap in large batch training of neural networks," in *NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems*, May 2017, pp. 1731–1741.

[6] A. E. Ibor, O. B. Okunoye, F. A. Oladeji, and K. A. Abdulsalam, "Novel hybrid model for intrusion prediction on cyber physical systems' communication networks based on bio-inspired deep neural network structure," *Journal of Information Security and Applications*, vol. 65, Mar. 2022, doi: 10.1016/j.jisa.2021.103107.

[7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, Illustrate. MIT Press, 2016.

[8] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.

[9] M. Jervis, M. Liu, and R. Smith, "Deep learning network optimization and hyperparameter tuning for seismic lithofacies classification," *The Leading Edge*, vol. 40, no. 7, pp. 514–523, Jul. 2021, doi: 10.1190/tle40070514.1.

[10] S. Dey, S. C. Kanala, and P. A. B. Keith M. Chugg, "Deep-n-cheap: An automated search framework for low complexity deep learning," in *the 12th Asian Conference on Machine Learning, PMLR*, 2020, vol. 129, pp. 273–288.

[11] I. Priyadarshini and C. Cotton, "A novel LSTM–CNN–grid search-based deep neural network for sentiment analysis," *The Journal of Supercomputing*, vol. 77, no. 12, pp. 13911–13932, Dec. 2021, doi: 10.1007/s11227-021-03838-w.

[12] J. Wu, X. Y. Chen, H. Zhang, L. D. Xiong, H. Lei, and S. H. Deng, "Hyperparameter optimization for machine learning models based on Bayesian optimization," *Journal of Electronic Science and Technology*, vol. 17, no. 1, pp. 26–40, 2019, doi: 10.11989/JEST.1674-862X.80904120.

[13] S. Bansal and A. Kumar, "Automatic deep neural network hyper-parameter optimization for maize disease detection," *IOP Conference Series: Materials Science and Engineering*, vol. 1022, no. 1, Jan. 2021, doi: 10.1088/1757-899X/1022/1/012089.

[14] H. Yi, K.-H. N. Bui, and H. Jung, "Implementing a deep learning framework for short term traffic flow prediction," in *Proceedings of the 9th International Conference on Web Intelligence, Mining and Semantics - WIMS2019*, 2019, pp. 1–8, doi: 10.1145/3326467.3326492.

[15] H. Yi and K.-H. N. Bui, "VDS data-based deep learning approach for traffic forecasting using LSTM network," in *Progress in Artificial Intelligence*, Springer, Cham, 2019, pp. 547–558.

[16] H. Alibrahim and S. A. Ludwig, "Hyperparameter optimization: Comparing genetic algorithm against grid search and Bayesian optimization," in *2021 IEEE Congress on Evolutionary Computation (CEC)*, Jun. 2021, vol. 978-1–7281, no. 21, pp. 1551–1559, doi: 10.1109/CEC45853.2021.9504761.

[17] Y. Wang, H. Zhang, and G. Zhang, "cPSO-CNN: An efficient PSO-based algorithm for fine-tuning hyper-parameters of convolutional neural networks," *Swarm and Evolutionary Computation*, vol. 49, pp. 114–123, Sep. 2019, doi: 10.1016/j.swevo.2019.06.002.

[18] P. R. Lorenzo, J. Nalepa, M. Kawulok, L. S. Ramos, and J. R. Pastor, "Particle swarm optimization for hyper-parameter selection in deep neural networks," in *Proceedings of the Genetic and Evolutionary Computation Conference*, Jul. 2017, pp. 481–488, doi: 10.1145/3071178.3071208.

[19] R. Mohakud and R. Dash, "Designing a grey wolf optimization based hyper-parameter optimized convolutional neural network classifier for skin cancer detection," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 8, pp. 6280–6291, Sep. 2022, doi: 10.1016/j.jksuci.2021.05.012.

[20] M. A. Hossain, E. M. Gray, M. R. Islam, R. K. Chakrabortty, and H. R. Pota, "Forecasting very short-term wind power generation using deep learning, optimization and data decomposition techniques," in *2021 24th International Conference on Electrical*

Machines and Systems (ICEMS), Oct. 2021, pp. 323–327, doi: 10.23919/ICEMS52562.2021.9634361.
[21] A. Ghaffari, “Image compression-encryption method based on two-dimensional sparse recovery and chaotic system,” *Scientific Reports*, vol. 11, no. 1, Dec. 2021, doi: 10.1038/s41598-020-79747-4.
[22] A. Mahdaddi, S. Meshoul, and M. Belguidoum, “EA-based hyperparameter optimization of hybrid deep learning models for effective drug-target interactions prediction,” *Expert Systems with Applications*, vol. 185, Art. no. 115525, Dec. 2021, doi: 10.1016/j.eswa.2021.115525.
[23] R. Barrio, A. Dena, and W. Tucker, “A database of rigorous and high-precision periodic orbits of the Lorenz model,” *Computer Physics Communications*, vol. 194, pp. 76–83, Sep. 2015, doi: 10.1016/j.cpc.2015.04.007.
[24] L. Abualigah, A. Diabat, S. Mirjalili, M. Abd Elaziz, and A. H. Gandomi, “The arithmetic optimization algorithm,” *Computer Methods in Applied Mechanics and Engineering*, vol. 376, Apr. 2021, doi: 10.1016/j.cma.2020.113609.
[25] K. K. Dobbin and R. M. Simon, “Optimally splitting cases for training and testing high dimensional classifiers,” *BMC Medical Genomics*, vol. 4, no. 1, Dec. 2011, doi: 10.1186/1755-8794-4-31.

BIOGRAPHIES OF AUTHORS

Nurnajmin Qasrina Ann received the Bachelor of Electrical and Electronics Engineering from University Tenaga Nasional, Malaysia, in 2016 and the Master of Science in electronics engineering from Universiti Malaysia Pahang, Malaysia in 2018. Currently, she is studying in Ph.D. in the Department of Electrical and Electronics Engineering, College of Engineering, Universiti Malaysia Pahang (UMP). She is interested in many research fields, such as image processing, optimization, deep learning, and chaos theory. She can be contacted at email: qasrinaann@mail.com.

Dwi Pebrianti is an assistant professor at the Department of Mechanical Engineering (Aerospace), International Islamic University Malaysia (IIUM). Previously, she was a senior lecturer in Universiti Malaysia Pahang, Malaysia in the field of Electrical and Electronics Engineering since 2013 until 2022. She received philosophy doctor in artificial system science from Chiba University, Japan in 2011, Master of Engineering in precision engineering from The University of Tokyo, Japan in 2006 and Bachelor of Engineering in electronics from Universitas Indonesia, Indonesia in 2001. Her main interests are including nonlinear control & robotics, unmanned aerial vehicle, underactuated mobile robot, vision-based robot navigation, motion & dynamics control, swarm robot control, optimization technique, machine learning and artificial intelligence. She was the principal investigator of 9 research grants including Exploratory Research Grant Scheme (ERGS) in 2013, Fundamental Research Grant Scheme (FRGS) in 2014 and Sustainable Research Collaboration Grant UMP-UiTM-IIUM in 2020. Additionally, she is the co-researcher for 17 research grants. She can be contacted at dwipebrianti@iium.edu.my.

Mohammad Fadhil Abas is a senior lecturer in Universiti Malaysia Pahang, Malaysia in the field of electrical and electronics engineering technology. He has been with Universiti Malaysia Pahang, Malaysia since 2004. He received philosophy doctor in artificial system science from Chiba University, Japan in 2013, Master of Science in power system from Universiti Putra Malaysia, Malaysia in 2006 and bachelor in electric, electronics and system from Universiti Kebangsaan Malaysia, Malaysia in 2001. His main interests include fault detection and identification, unmanned aerial vehicle, underactuated mobile robot, navigation and positioning system, optimization technique, machine learning and artificial intelligence, and water quality systems. He was the principal investigator of 8 research grants including Fundamental Research Grant Scheme (FRGS) in 2019. Additionally, he is the co-researcher for 17 research grants. He can be contacted at mfadhil@ump.edu.my.

Luhur Bayuaji received the B.Eng. degree in electrical engineering majoring computer engineering in 2001 from Universitas Indonesia, Indonesia and M. Eng. and Ph.D. both from Chiba University, Japan in 2006 and 2010. His research interest includes computer networks, internet of things (IoT), cybersecurity, image processing, remote sensing, and geographical information system (GIS). He is currently an assistant professor in Faculty of Computing, Universiti Malaysia Pahang, since 2013. Additionally, he is a Master Trainer for Cisco Networking Academy, who is responsible for teaching and give training Cisco Networking Academy (NetAcad) instructor candidate and student for the following courses: Cisco Certified Network Associate (CCNA), CyberOps Associate, DevNet Associate, Network Security, Cisco Certified Network Professional–Enterprise, Python Programming, and Network Essential and Cybersecurity Essential. He can be emailed at luhurbayuaji@gmail.com.