JURNAL ONLINE INFORMATIKA

JOIN (Jurnal Online Informatika)
Volume 7 No. 2 | December 2022: 193-202
DOI: 10.15575/join.v7i2.858

p-ISSN: 2528-1682
e-ISSN: 2527-9165

PSO based Hyperparameter tuning of CNN Multivariate Time-Series Analysis

Agung Bella Putra Utama[1], Aji Prasetya Wibawa[2], Muladi[3], Andrew Nafalski[4]
[1,2,3]Department of Electrical Engineering, Universitas Negeri Malang, Indonesia
[4]UniSA Education Futures, School of Engineering, University of South Australia, Australia

Article Info

Article history:

Received February 15, 2022
Revised March 17, 2022
Accepted May 11, 2022
Published December 26, 2022

Keywords:

CNN
Hyperparameter tuning
Multivariate time-series
PSO

ABSTRACT

Convolutional Neural Network (CNN) is an effective Deep Learning (DL) algorithm that solves various image identification problems. The use of CNN for time-series data analysis is emerging. CNN learns filters, representations of repeated patterns in the series, and uses them to forecast future values. The network performance may depend on hyperparameter settings. This study optimizes the CNN architecture based on hyperparameter tuning using Particle Swarm Optimization (PSO), PSO-CNN. The proposed method was evaluated using multivariate time-series data of electronic journal visitor datasets. The CNN equation in image and time-series problems is the input given to the model for processing numbers. The proposed method generated the lowest RMSE (1.386) with 178 neurons in the fully connected and 2 hidden layers. The experimental results show that the PSO-CNN generates an architecture with better performance than ordinary CNN.

Corresponding Author:

Aji Prasetya Wibawa,
Department of Electrical Engineering, Faculty of Engineering,
Universitas Negeri Malang,
Jl. Semarang No. 5, Malang, Indonesia, 65145
Email: aji.prasetya.ft@um.ac.id

1. INTRODUCTION

Hyperparameter tuning arises because the default hyperparameter settings cannot guarantee the optimal performance of the model [1]. Hyperparameter tuning aims to find the hyperparameter values to produce the best-performing model. Each Machine Learning (ML) model has different hyperparameter settings [2]. The hyperparameter tuning settings are also different for each dataset [3]. The tuning is critical for tree-based ML models and deep neural networks with many hyperparameters [4]. Therefore, choosing the correct optimization technique is the key to generating optimal hyperparameter tuning.

Traditional optimization techniques may not suit hyperparameter tuning problems because of local values rather than a global optimal [5]. Gradient descent is a common type of traditional optimization algorithm that can tune hyperparameter tuning by calculating its gradient [6]. In addition, Grid Search (GS) [7], [8] and Random Search (RS) [9], [10] are traditional methods for hyperparameter tuning. Many other optimization approaches, such as metaheuristic algorithms, are more suited to hyperparameter tuning issues than classic optimization methods. [11]. A metaheuristic algorithm is a collection of strategies for solving increasingly significant and sophisticated search space optimization issues, such as hyperparameter tuning. [12].

Among metaheuristic methods, Genetic Algorithm (GA) [13] and Particle Swarm Optimization (PSO) [14] are the most popular algorithms for hyperparameter tuning problems. GA detects the combination of hyperparameters that perform well in each generation and passes them on to the next generation until the combination with the best performance is identified [15]. Each particle communicates with other particles in the PSO algorithm to detect and update the global optimal in each iteration until the last optimal value is detected [16]. The metaheuristic algorithm can efficiently explore the search space to detect optimal or near-optimal solutions. PSO is one of the evolutionary algorithms, and it is impossible to get stuck in the local optimum compared to other optimization methods. PSO is generally simpler than GA because complicated operations like permutations are not required. PSO also provides more consistent performance compared to

GA. Therefore, the primary SI metaheuristic method, PSO, is suitable for hyperparameter tuning problems with large configuration space because of its high efficiency [17]. Thus, it can be used in Deep Learning (DL) which has a large configuration space with several hyperparameters, such as activation type and optimizer.

CNN is one of the DL algorithms that successfully demonstrated its effectiveness in various image problems [18]. The CNN process is broadly divided into feature extraction and fully connected layers. The feature extraction layer extracts features in an image in numbers representing the image [19]. In this layer, there are two processes, namely convolution and polling. The feature map generated from the feature extraction layer is still in the form of a multidimensional array. Flattening or reshaping the feature map into a vector for input from the fully connected layer is necessary. The fully connected layer has the same way of working as the Artificial Neural Network (ANN) and has the same parts, namely the input layer, hidden layer, activation function, and output layer [20]. This layer aims so that the data can be classified. CNN network performance depends directly on its hyperparameters. PSO has been tried to optimize the CNN hyperparameters [21]–[24]. The results show that PSO can increase the accuracy of CNN by finding a better hyperparameter configuration. Most PSO tuning hyperparameters solve image classification problems, and computer vision from all existing studies may be used in time-series data analysis. The goal of using CNN on time series data is to learn filters that represent repeated patterns in the series and use them to forecast future values. CNN can also automatically learn and extract features from raw data without prior knowledge or feature engineering. The method may work well on noisy time series by discarding noise at each subsequent layer, constructing a hierarchy of valuable features, and extracting only the meaningful features [25].

Thus, the author optimizes the CNN architecture by utilizing PSO's global and local exploration capabilities in this study. This research focuses on using PSO-CNN for hyperparameter tuning in multivariate time series analysis. Hyperparameter tuning on the fully connected layer process to get the best minimum error with the six hyperparameters generated. The six hyperparameters are the number of hidden layer units (neurons), type of activation function, loss function, optimizer, number of epochs, and batch sizes. The contribution of this research is the use of PSO as a tuning hyperparameter on CNN. The tuning maintains the use of all existing features to achieve the initial goal of multivariate time-series analysis. This research is different from other studies, which use various neurons [26], activation function [27], and optimizer [28] to get the best forecasting performances. So, the novelty of this research is that hyperparameter tuning with PSO can be used to automatically generate efficient architectures without doing too much experimentation on various possible configurations of CNN parameters, which can reduce the long experimental and computation time.

This article consists of 4 structural sections. Section 1 introduces issues that require tuning CNN hyperparameters and an overview of similar research. The methods and theories used in this research are presented in Section 2. This method is implemented, and the results are analyzed in Section 3. The last section describes the conclusions from PSO tuning hyperparameter CNN results.

2. METHODS

This section describes the methods used to evaluate the setting of PSO tuning hyperparameter CNN. The process begins with the collection of datasets. The preprocessing process is done by normalizing the data. Evaluation is done using RMSE.

2.1. Dataset Collection

The data used is time-series data on the number of visitors to the knowledge engineering and data science (KEDS) journal website from January 1, 2018, to December 31, 2020. KEDS website can be accessed at http://journal2.um.ac.id/index.php/keds. The dataset is a multivariate type that contains four attributes: sessions, page views, visitors, and new visitors. Sessions is the number of visitors from one IP who access a journal portal for the first time in a certain period [29]. Pageviews are the total number of web pages visited. Visitors are the total number of website visitors, while new visitors are the ones who visit the website for the first time. The dataset visualization can be seen in Figure 1.

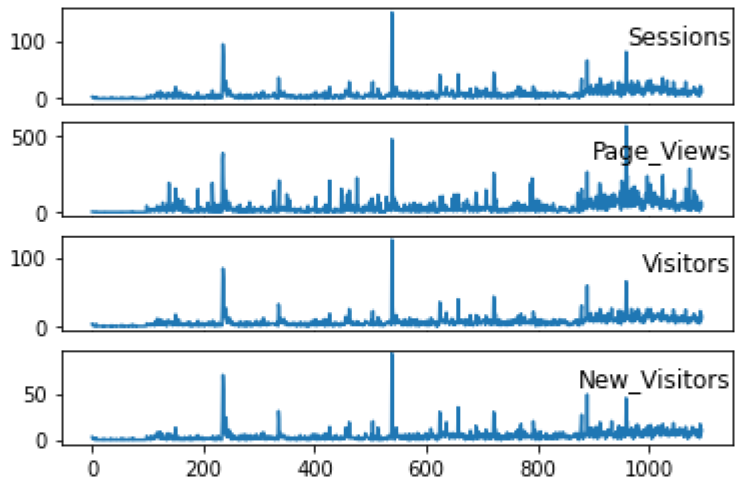

Figure 1. Dataset

In this study, the data used is annual data, resulting in three different dataset models, as shown in Table 1.

Table 1. Dataset Model

Model	Training (Year)	Testing (Year)
1	2018	2019
2	2019	2020
3	2018-2019	2020

2.2. Preprocessing

Preprocessing is done to improve the quality of input data to get results with a high level of performance. Preprocessing used is data normalization. Normalization of the data is carried out so that the network output follows the activation function used. Data normalization aims to minimize errors by changing the actual value to a value with a range of 0 to 1. The data normalization technique used is min-max normalization [30]. The min-max normalization equation can be seen in Eq. (1).

$$X' = \frac{(x - min_x)}{(max_x - min_x)} \tag{1}$$

X' is the result of normalization, x is the data to be normalized, min_x and max_x , represent the minimum and maximum values of all data.

2.3. Convolutional Neural Network (CNN)

CNN is included in DL, which is a sub-field of ML by applying the basic concepts of the ANN algorithm with more layers [31]. CNN is a feedforward network because the information flow occurs only in one direction, namely from input to output. CNN is quite popular and has good performance in various fields such as image processing and computer vision [32]. CNN uses a convolution layer that can handle the spatial information available in the image. In contrast, fully connected layers have a memory to store information available in time-series data [33]. The only difference between computer vision and time-series problems is the input given to the model, the image matrix input for computer vision, and the 1D array input for time series forecasting [34]. Thus, this principle can be implemented in time-series analysis. The architecture of the 1D CNN model is in Figure 2.

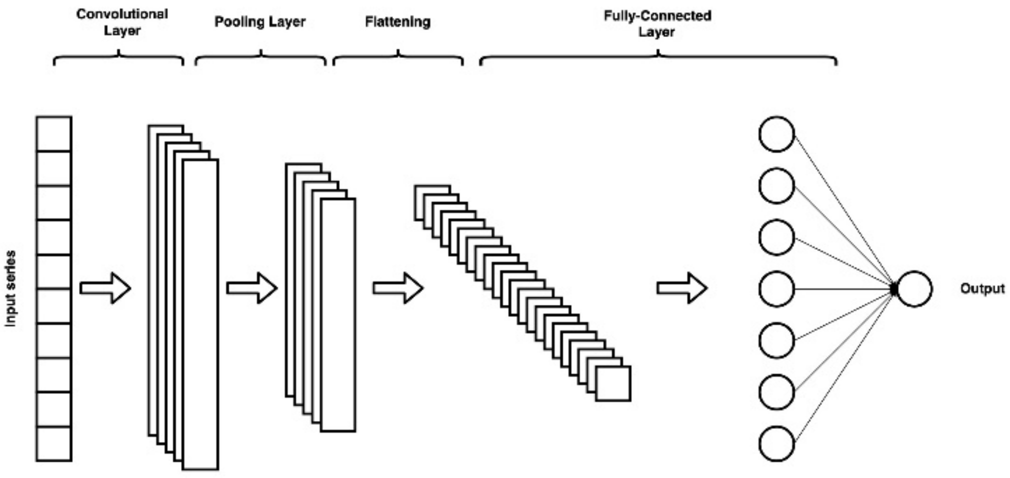

Figure 2. 1D CNN Layer Structure

To get good results when using the CNN method is not easy because it must require consideration of setting various parameters in the process. In CNN, there are several lists of different categories of hyperparameters according to the layer. The list of hyperparameters on CNN is in Table 2 [24][35].

Table 2. The list of Hyperparameters

Category	Hyperparameter
Convolutional layer	The number of convolutional layers
	The number of filters
	The filter size
	The activation function
	The pooling layer size
Pooling layer	Size of the pooling window
	Type of pooling
	The number of pooling layer
	Stride or step size
Fully connected layer	The number of units or neuron
	The Activation function output
	Loss function
	Type of optimizer
	The number of epochs
	The batch size

2.4. Particle Swarm Optimization (PSO)

PSO is a general optimization method that belongs to the straightforward metaheuristic. It searches for the optimal solution in search space within the swarm [36]. PSO has several advantages: better convergence, high efficient computing, and low use of computing resources [37]. PSO runs applying a population of particles with their respective velocities and positions. The algorithm will iterate over all the updated particles until it reaches the optimal value. Each particle gets a solution from the process individually or globally to update all velocities and positions. The update process can be done by calculating of an i^{th} particle in $(k+1)^{th}$ for speed in Eq. (2) and for position in Eq. (3).

$$\vec{v_i}(k+1) = \omega v_1(k) + c_1 r_1(\overrightarrow{pbest_1}(k) - \vec{x_i}\,(k)) + c_2 r_2(\overrightarrow{gbest}(k) - \vec{x_i}\,(k)) \tag{2}$$

$$\vec{x_i}\,(k+1) = \vec{x_i}(k) + \vec{v_i}(k+1) \tag{3}$$

The social and cognitive behavior of each particle influences the particle's mobility. The swarm's best experience determines social behavior $\overrightarrow{gbest}$ and the particle's own optimal experience in terms of cognitive behavior $\overrightarrow{pbest}$, traveled a long way. Two random variables r_1 and r_2 have values ranging from 0 to 1. The parameters inertia weight (ω), social coefficient (c_1), and cognitive coefficient (c_2) are used to manage particle behavior and balance exploration and exploitation.

2.5. The proposed method: PSO tuning hyperparameter CNN

This section presents an optimization approach for applying the PSO algorithm to optimize the CNN architecture hyperparameters, and this approach is called PSO tuning hyperparameter CNN. The first objective is to select the hyperparameters that affect good CNN performance and then implement the PSO algorithm to find the optimal parameters. The proposed PSO tuning hyperparameter CNN method can be seen in Figure 3.

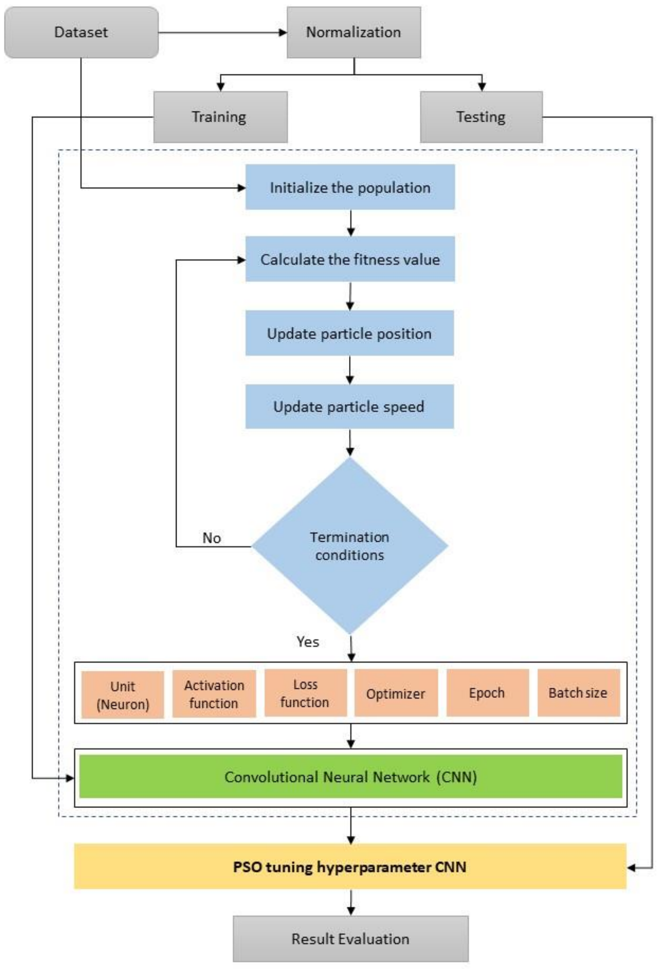

Figure 3. Proposed Method PSO tuning hyperparameter CNN

From the list of hyperparameters in Table 2, to solve problems in image classification, the majority of hyperparameters in all categories are used. Meanwhile, this research focuses on using PSO for tuning hyperparameter CNN in multivariate time-series analysis. Hyperparameter tuning will be carried out only on the fully connected layer. The hyperparameter space of the fully connected layer can be seen in Table 3. The dropout values used are equal, namely 0.2 [38].

Table 3. Hyperparameter Space

Hyperparameter	Search Space
Unit (Neuron)	16 - 256
Activation function output	Linear, ReLU, Tanh
Loss function	MSE, MAE
Optimizer	Adam, RMSprop
Epoch	5 - 500
Batch size	16, 32, 64

The selection of each parameter in Table 3 is based on a combination of research by [24], [39]–[42]. The first hyperparameter is the number of neurons; choosing the optimal number of neurons for each layer in the CNN network is not easy. If the number of neurons is tiny, the CNN will not remember all the information needed to make optimal predictions. Conversely, if the number of neurons is very high, the CNN will overfit the training instance and not show appropriate generalizations to estimate the test set accurately. The second is the activation function, which is a function used in the neural network to activate or deactivate the neurons in the network [43]. Many types of activation functions can be used, and three of them are linear, ReLU and tanh. The third is the loss function; in calculating the model error during the optimization process, a loss function must be chosen between MSE and MAE [44]. The fourth is the optimizer, an algorithm used to solve optimization problems that aim to minimize the objective function, which is intuitively the difference between the predicted data and the expected value [45]. There are various optimizers, but the most common and frequently used are Adam and RMSprop. Fifth is the number of epoch usage; if the number of training epochs is too small, the model will not capture the training instance pattern. However, the model will be overfitted if the epoch number is too large. Therefore, finding the appropriate number of epochs is essential in achieving a high-performance model [46]. Finally, namely the batch size, network settings with a batch value size that is too large or too low can reduce the stochasticity of the cause of gradient descent. It can show a negative effect on the results of the forecasting model during the training process [47].

2.6. Result Evaluation

The value of the forecasting results will be evaluated using the nominal error value from the root mean square error (RMSE). RMSE is a calculation by getting the average value of the sum of the squares of errors. RMSE is used to detect irregularities or outliers in the designed prediction system. The equation for calculating RMSE can be seen in Eq. (4) [48].

$$RMSE = \sqrt{\sum_{t=1}^{n} \frac{(e_t)^2}{n}} \tag{4}$$

$RMSE$ is the root mean square error value, $\sum$ is the total number of values, n is the number of data, e_t is the difference between the actual data and the forecast values. From the calculation of the RMSE value, it will be known which model has the best performance in forecasting. The smaller the resulting RMSE value, the better the forecasting results, the better the method used [49].

3. RESULTS AND DISCUSSION

Selection of the exemplary architecture and hyperparameters from the CNN model requires a very long process by trial-and-error. Therefore, an automated approach is needed to produce the best CNN architectural model with minimal human intervention. The proposed research uses random but integrated PSO properties to find the best CNN structure by optimizing hyperparameters on a given data set in a predefined search space. The optimization process requires determining the hyperparameter space at the beginning to determine the search. This section demonstrates the effectiveness of PSO tuning hyperparameter CNN on three types of dataset models. The proposed PSO tuning hyperparameter CNN model was implemented, trained, and tested using Google's Collaboratory.

Using CNN, set the hyperparameter using the default values for each layer category. The convolutional layer uses 64 filters with 1 kernel and the ReLU activation function in hidden and output layer. In the pooling layer, use 1 size and dropout 0.2. For the fully connected layer, the number of neurons varies from 16, 32, 64, 128, and 256, with a dropout of 0.2, batch size 16, and epochs 500. In this study, experiments were also carried out by adding hidden layers from 2-10 for ordinary CNN. The results of forecasting RMSE using CNN can be seen in Table 4. The best RMSE of 2.314 is obtained in Model 1 using 2 hidden layers and 64 neurons.

Table 4. RMSE of CNN

Hidden Layers	Neuron														
	Model 1					Model 2					Model 3				
	16	32	64	128	256	16	32	64	128	256	16	32	64	128	256
2	3.166	2.885	**2.314**	3.233	2.586	3.354	3.631	3.605	3.302	**3.186**	2.934	3.140	3.079	2.747	**2.331**
3	4.060	3.144	3.580	2.940	2.688	3.789	4.044	3.578	3.473	3.449	3.669	3.576	3.251	3.317	3.225
4	5.056	4.387	3.005	3.633	3.882	5.297	5.187	4.747	4.261	4.582	5.188	3.986	4.165	5.341	3.954
5	5.346	4.456	4.424	3.412	6.908	5.112	5.010	4.482	4.122	4.897	5.573	4.843	4.738	3.864	3.990
6	6.317	5.355	6.019	4.894	7.548	5.539	5.615	5.293	5.634	5.319	6.689	5.553	5.692	5.610	5.767
7	6.936	7.044	5.054	6.110	9.697	6.585	6.384	5.931	5.542	5.640	7.304	6.524	5.700	6.761	8.813
8	6.993	6.901	5.882	7.634	9.038	7.372	7.022	6.129	7.530	8.217	7.376	7.483	7.053	8.596	9.311
9	7.977	6.905	7.187	9.244	9.549	7.833	8.146	7.938	8.981	9.279	7.239	6.225	6.888	8.161	9.475
10	8.304	7.330	7.783	8.557	9.779	9.010	8.075	8.556	8.905	9.272	7.988	7.469	8.396	8.872	9.435

The results of hyperparameter tuning on different datasets also have different values. So, in this research, there will be three kinds of results from the hyperparameter search process of PSO tuning hyperparameter CNN according to the dataset model described in Table 1. The PSO tuning hyperparameter CNN hyperparameter search results for each model can be seen in Table 5.

Table 5. PSO Hyperparameter Search Result CNN

Hyperparameters	**Model 1**	**Model 2**	**Model 3**
Unit (Neuron)	28	131	178
Activation function ouput	ReLU	Tanh	Tanh
Loss function	MAE	MSE	MSE
Optimizer	RMSprop	RMSprop	Adam
Epoch	52	69	60
Batch size	16	16	32

From Table 5, it can be seen that the value of the hyperparameter search results in each model is different because the tuning process also pays attention to the contents of the dataset. The use of the number of neurons in each model in the ordinary CNN experiment has five scenarios, while in the PSO tuning hyperparameter CNN, the number of neurons in each model follows the results of the hyperparameter tuning. Using the number of neurons respectively for Models 1 to 3 were 28, 131, and 178. After all the hyperparameter tuning results in Table 5 are implemented, the new RMSE for each model can be seen in Table 6.

Table 6. RMSE of PSO tuning hyperparameter CNN

Hidden Layers	**Model 1**	**Model 2**	**Model 3**
2	**1.840**	**1.912**	**1.386**
3	2.498	2.406	1.880
4	3.346	2.204	2.236
5	4.320	2.090	1.998
6	5.500	2.650	2.734
7	6.060	2.628	3.310
8	6.098	2.240	2.954
9	6.602	2.682	3.602
10	6.568	2.612	3.342

Based on Table 6, it can be noticed that the results of Model 1 after tuning obtained a better RMSE value of 1.840 when using 2 hidden layers with 28 neuron units. The number of hidden layers in Model 1 also makes the resulting RMSE value more significant. When using 10 hidden layers on an ordinary CNN, it produces a value of RMSE that ranges from 7.330 – 9.779, while after tuning with PSO tuning hyperparameter CNN, it produces an RMSE which is a better value is 6.568. Likewise, with the other two models. RMSE in Model 2 shows that PSO tuning hyperparameter CNN results have minor RMSE results compared to ordinary CNN. The best RMSE in Model 2 is acquired using 2 hidden layers with 131 neurons, 1.912. Model 3 also has the best RMSE value resulting from the PSO tuning hyperparameter CNN process of 1.386 compared to the ordinary CNN when using 2 hidden layers with 78 neurons. The RMSE results are also the best among the PSO tuning hyperparameter CNN models.

The overall RMSE PSO tuning hyperparameter CNN results in all models in Table 6 have a better value when compared to ordinary CNN in Table 4, happens because PSO can automatically search for the best hyperparameter space to experiments that perform trial and error of CNN architecture. In addition, the use of 2 hidden layers in this study resulted in the best RMSE value among the addition of other hidden layers, both on ordinary CNN and PSO tuning hyperparameter CNN. A paired t-test was used to examine the influence of changes in the number of hidden layers on the RMSE value obtained by the PSO tuning hyperparameter CNN method. Table 7 displays the significance test results.

Table 7. Result of Paired T-test

Model	**2-tailed P value**
1	0.0094
2	0.0028
3	0.0011

From the significance test results in Table 7, it is known that the PSO tuning hyperparameter CNN results in all models produce a significance 2-tailed P value of < 0.05. This shows a significant effect on the difference in the use of the number of hidden layers on the resulting RMSE value. From the overall results, the PSO tuning hyperparameter CNN can improve the RMSE results and have a significant result for increasing the number of hidden layers.

4. CONCLUSION

This study optimizes CNN hyperparameters using the PSO method. The PSO algorithm generates the right hyperparameter to reduce the error value (RMSE) by comparing the ordinary CNN model with a higher RMSE value than the proposed PSO tuning hyperparameter CNN method. In addition, based on the results of the significance test, it is also known that there is a significant effect on increasing the number of hidden layers in the RMSE PSO tuning hyperparameter CNN results, which shows that tuning the PSO algorithm can produce an optimal architecture with better multivariate forecasting values. In future studies, a comparison of optimization using the Gray Wolf Optimizer (GWO) and transfer learning techniques will be carried out.

ACKNOWLEDGEMENTS

We thank Universitas Negeri Malang (UM) supported this work through a research grant, PPM 2022. We want to thank the KEDS journal for sharing the visitor dataset.

5. REFERENCES

[1] P. Schratz, J. Muenchow, E. Iturritxa, J. Richter, and A. Brenning, "Hyperparameter tuning and performance assessment of statistical and machine-learning algorithms using spatial data," *Ecol. Modell.*, vol. 406, pp. 109–120, Aug. 2019, doi: 10.1016/j.ecolmodel.2019.06.002.

[2] N. DeCastro-García, Á. L. Muñoz Castañeda, D. Escudero García, and M. V. Carriegos, "Effect of the Sampling of a Dataset in the Hyperparameter Optimization Phase over the Efficiency of a Machine Learning Algorithm," *Complexity*, vol. 2019, pp. 1–16, Feb. 2019, doi: 10.1155/2019/6278908.

[3] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, Nov. 2020, doi: 10.1016/j.neucom.2020.07.061.

[4] F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated Machine Learning*. Cham: Springer International Publishing, 2019.

[5] G. Luo, "A review of automatic selection methods for machine learning algorithms and hyper-parameter values," *Netw. Model. Anal. Heal. Informatics Bioinforma.*, vol. 5, no. 1, p. 18, Dec. 2016, doi: 10.1007/s13721-016-0125-6.

[6] D. Maclaurin, D. Duvenaud, and R. P. Adams, "Gradient-based Hyperparameter Optimization through Reversible Learning," *Proc. 32nd Int. Conf. Mach. Learn.*, vol. 37, 2015, [Online]. Available: http://arxiv.org/abs/1502.03492.

[7] B. H. Shekar and G. Dagnew, "Grid Search-Based Hyperparameter Tuning and Classification of Microarray Cancer Data," in *2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP)*, Feb. 2019, pp. 1–8, doi: 10.1109/ICACCP.2019.8882943.

[8] L. Zahedi, F. G. Mohammadi, S. Rezapour, M. W. Ohland, and M. H. Amini, "Search Algorithms for Automated Hyper-Parameter Tuning," pp. 1–10, 2021, [Online]. Available: http://arxiv.org/abs/2104.14677.

[9] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, 2012.

[10] L. Li and A. Talwalkar, "Random search and reproducibility for neural architecture search," *35th Conf. Uncertain. Artif. Intell. UAI 2019*, 2019.

[11] B. Bischl *et al.*, "Hyperparameter Optimization: Foundations, Algorithms, Best Practices and Open Challenges," 2021, [Online]. Available: http://arxiv.org/abs/2107.05847.

[12] Q. Yao *et al.*, "Taking the Human out of Learning Applications : A Survey on Automated Machine Learning," pp. 1–20.

[13] 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, pp. 1551–1559, doi: 10.1109/CEC45853.2021.9504761.

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

[15] X. Xiao, M. Yan, S. Basodi, C. Ji, and Y. Pan, "Efficient Hyperparameter Optimization in Deep Learning Using a Variable Length Genetic Algorithm," *arXiv*, vol. 1, 2020.

[16] P. Singh, S. Chaudhury, and B. K. Panigrahi, "Hybrid MPSO-CNN: Multi-level Particle Swarm optimized hyperparameters of Convolutional Neural Network," *Swarm Evol. Comput.*, vol. 63, p. 100863, Jun. 2021, doi: 10.1016/j.swevo.2021.100863.

[17] S. Fong, S. Deb, and X. Yang, "How Meta-heuristic Algorithms Contribute to Deep Learning in the Hype of Big Data Analytics," in *Progress in Intelligent Computing Techniques: Theory, Practice, and Applications*, 2018, pp. 3–25.

[18] V. Passricha and R. K. Aggarwal, "PSO-based optimized CNN for Hindi ASR," *Int. J. Speech Technol.*, vol. 22, no. 4, pp. 1123–1133, Dec. 2019, doi: 10.1007/s10772-019-09652-3.

[19] A. Latif *et al.*, “Content-Based Image Retrieval and Feature Extraction: A Comprehensive Review,” *Math. Probl. Eng.*, vol. 2019, pp. 1–21, Aug. 2019, doi: 10.1155/2019/9658350.

[20] F. Ghasemi, A. Mehridehnavi, A. Pérez-Garrido, and H. Pérez-Sánchez, “Neural network and deep-learning algorithms used in QSAR studies: merits and drawbacks,” *Drug Discov. Today*, vol. 23, no. 10, pp. 1784–1790, Oct. 2018, doi: 10.1016/j.drudis.2018.06.016.

[21] L. M. R. Rere, M. I. Fanany, and A. M. Arymurthy, “Metaheuristic Algorithms for Convolution Neural Network,” *Comput. Intell. Neurosci.*, vol. 2016, pp. 1–13, 2016, doi: 10.1155/2016/1537325.

[22] T. Yamasaki, T. Honma, and K. Aizawa, “Efficient Optimization of Convolutional Neural Networks Using Particle Swarm Optimization,” in *2017 IEEE Third International Conference on Multimedia Big Data (BigMM)*, Apr. 2017, pp. 70–73, doi: 10.1109/BigMM.2017.69.

[23] T. Sinha, A. Haidar, and B. Verma, “Particle Swarm Optimization Based Approach for Finding Optimal Values of Convolutional Neural Network Parameters,” in *2018 IEEE Congress on Evolutionary Computation (CEC)*, Jul. 2018, pp. 1–6, doi: 10.1109/CEC.2018.8477728.

[24] E. Tuba, N. Bačanin, I. Strumberger, and M. Tuba, “Convolutional Neural Networks Hyperparameters Tuning,” in *Artificial Intelligence: Theory and Applications*, 2021, pp. 65–84.

[25] I. Koprinska, D. Wu, and Z. Wang, “Convolutional Neural Networks for Energy Time Series Forecasting,” in *2018 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2018, pp. 1–8, doi: 10.1109/IJCNN.2018.8489399.

[26] A. R. F. Dewandra, A. P. Wibawa, U. Pujianto, A. B. P. Utama, and A. Nafalski, “Journal Unique Visitors Forecasting Based on Multivariate Attributes Using CNN,” *Int. J. Artif. Intell. Res.*, vol. 6, no. 1, 2022, doi: https://doi.org/10.29099/ijair.v6i1.274.

[27] M. A. Morid, O. R. L. Sheng, K. Kawamoto, and S. Abdelrahman, “Learning hidden patterns from patient multivariate time series data using convolutional neural networks: A case study of healthcare cost prediction,” *J. Biomed. Inform.*, vol. 111, p. 103565, Nov. 2020, doi: 10.1016/j.jbi.2020.103565.

[28] S. Liu, H. Ji, and M. C. Wang, “Nonpooling Convolutional Neural Network Forecasting for Seasonal Time Series With Trends,” *IEEE Trans. Neural Networks Learn. Syst.*, vol. 31, no. 8, pp. 2879–2888, Aug. 2020, doi: 10.1109/TNNLS.2019.2934110.

[29] A. P. Wibawa, Z. N. Izdihar, A. B. P. Utama, L. Hernandez, and Haviluddin, “Min-Max Backpropagation Neural Network to Forecast e-Journal Visitors,” in *2021 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, Apr. 2021, pp. 052–058, doi: 10.1109/ICAIIC51459.2021.9415197.

[30] T. Shintate and L. Pichl, “Trend Prediction Classification for High Frequency Bitcoin Time Series with Deep Learning,” *J. Risk Financ. Manag.*, vol. 12, no. 1, p. 17, Jan. 2019, doi: 10.3390/jrfm12010017.

[31] A. S. Lundervold and A. Lundervold, “An overview of deep learning in medical imaging focusing on MRI,” *Z. Med. Phys.*, vol. 29, no. 2, pp. 102–127, May 2019, doi: 10.1016/j.zemedi.2018.11.002.

[32] B. S. Kim and T. G. Kim, “Cooperation of Simulation and Data Model for Performance Analysis of Complex Systems,” *Int. J. Simul. Model.*, vol. 18, no. 4, pp. 608–619, Dec. 2019, doi: 10.2507/IJSIMM18(4)491.

[33] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, “Convolutional neural networks: an overview and application in radiology,” *Insights Imaging*, vol. 9, no. 4, pp. 611–629, Aug. 2018, doi: 10.1007/s13244-018-0639-9.

[34] E. Lewinson, “Python for Finance Cookbook,” in *Over 50 recipes for applying modern Python libraries to financial data analysis*, 1st ed., Packt Publishing, 2020, p. 434.

[35] J. Fregoso, C. I. Gonzalez, and G. E. Martinez, “Optimization of Convolutional Neural Networks Architectures Using PSO for Sign Language Recognition,” *Axioms*, vol. 10, no. 3, 2021, doi: https://doi.org/10.3390/axioms10030139.

[36] H.-Y. Tseng, P.-H. Chu, H.-C. Lu, and M.-J. Tsai, “Easy Particle Swarm Optimization for Nonlinear Constrained Optimization Problems,” *IEEE Access*, vol. 9, pp. 124757–124767, 2021, doi: 10.1109/ACCESS.2021.3110708.

[37] W. Lu, W. Jiang, N. Zhang, and F. Xue, “Application of PSO-based LSTM Neural Network for Outpatient Volume Prediction,” *J. Healthc. Eng.*, vol. 2021, pp. 1–9, Nov. 2021, doi: 10.1155/2021/7246561.

[38] J. Oh, J. Wang, and J. Wiens, “Learning to Exploit Invariances in Clinical Time-Series Data using Sequence Transformer Networks,” pp. 1–15, 2018, [Online]. Available: http://arxiv.org/abs/1808.06725.

[39] A. Parashar and A. Sonker, “Application of hyperparameter optimized deep learning neural network for classification of air quality data,” *Int. J. Sci. Technol. Res.*, vol. 8, no. 11, pp. 1435–1443, 2019.

[40] M. Tovar, M. Robles, and F. Rashid, “PV Power Prediction, Using CNN-LSTM Hybrid Neural Network Model. Case of Study: Temixco-Morelos, México,” *Energies*, vol. 13, no. 24, p. 6512, Dec. 2020, doi: 10.3390/en13246512.

[41] C. Pelletier, G. Webb, and F. Petitjean, “Temporal Convolutional Neural Network for the Classification of Satellite Image Time Series,” *Remote Sens.*, vol. 11, no. 5, p. 523, Mar. 2019, doi: 10.3390/rs11050523.

[42] R. Zatarain Cabada, H. Rodriguez Rangel, M. L. Barron Estrada, and H. M. Cardenas Lopez, “Hyperparameter optimization in CNN for learning-centered emotion recognition for intelligent tutoring systems,” *Soft Comput.*, vol. 24, no. 10, pp. 7593–7602, May 2020, doi: 10.1007/s00500-019-04387-4.

[43] S. Sharma, S. Sharma, and A. Athaiya, “Activation Functions in Neural Networks,” *Int. J. Eng. Appl. Sci. Technol.*, vol. 04, no. 12, pp. 310–316, 2020, doi: 10.33564/ijeast.2020.v04i12.054.

[44] V. N. Sewdien, R. Preece, J. L. R. Torres, E. Rakhshani, and M. van der Meijden, “Assessment of critical parameters for artificial neural networks based short-term wind generation forecasting,” *Renew. Energy*, vol. 161, pp. 878–892, Dec. 2020, doi: 10.1016/j.renene.2020.07.117.

[45] E. Okewu, P. Adewole, and O. Sennaike, “Experimental Comparison of Stochastic Optimizers in Deep Learning,” in *Lecture Notes in Computer Science*, 2019, pp. 704–715.

[46] H. Abbasimehr, M. Shabani, and M. Yousefi, “An optimized model using LSTM network for demand forecasting,” *Comput. Ind. Eng.*, vol. 143, no. July 2019, p. 106435, May 2020, doi: 10.1016/j.cie.2020.106435.

[47] Q. Zheng, X. Tian, N. Jiang, and M. Yang, “Layer-wise learning based stochastic gradient descent method for the optimization of deep convolutional neural network,” *J. Intell. Fuzzy Syst.*, vol. 37, no. 4, pp. 5641–5654, Oct. 2019, doi: 10.3233/JIFS-190861.

[48] T. T. Kieu Tran, T. Lee, J. Y. Shin, J. S. Kim, and M. Kamruzzaman, “Deep learning-based maximum temperature forecasting assisted with meta-learning for hyperparameter optimization,” *Atmosphere (Basel).*, vol. 11, no. 5, pp. 1–21, 2020, doi: 10.3390/ATMOS11050487.

[49] Z. Alameer, M. A. Elaziz, A. A. Ewees, H. Ye, and Z. Jianhua, “Forecasting gold price fluctuations using improved multilayer perceptron neural network and whale optimization algorithm,” *Resour. Policy*, vol. 61, no. September 2018, pp. 250–260, 2019, doi: 10.1016/j.resourpol.2019.02.014.