Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control
Journal homepage: http://kinetik.umm.ac.id
ISSN: 2503-2267
Vol. 6, No. 4, November 2021, Pp. 287-294

Implementation of Particle Swarm Optimization (PSO) to improve neural network performance in univariate time series prediction

Fitri Ayuning Tyas[*1], Mamur Setianama[2], Rizqi Fadilatul Fajriyah[3], Ahmad Ilham[4]
STMIK Muhammadiyah Paguyangan Brebes, Indonesia[1,2,3]
Universitas Muhammadiyah Semarang, Indonesia[4]

Article Info
Keywords:
Particle Swarm Optimation (PSO), Neural Network, Forecasting, Prediction, Univariate Time Series

Article history:
Received: August 27, 2021
Accepted: October 29, 2021
Published: November 30, 2021

Cite:
Tyas, F. A., Setianama, M., Fadilatul Fajriyah, R., & Ilham, A. (2021). Implementation of Particle Swarm Optimization (PSO) to Improve Neural Network Performance in Univariate Time Series Prediction. Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control, 6(4).
https://doi.org/10.22219/kinetik.v6i4.1330

*Corresponding author
Fitri Ayuning Tyas
E-mail address:
tyas_fa@stmikmpb.ac.id

Abstract
One of the oldest known predictive analytics techniques is time series prediction. The target in time series prediction is use historical data about a specific quantity to predicts value of the same quantity in the future. Multivariate time series (MTS) data has been widely used in time series prediction research because it is considered better than univariate time series (UTS) data. However, in reality MTS data sets contain various types of information which makes it difficult to extract information to predict the situation. Therefore, UTS data still has a chance to be developed because it is actually simpler than MTS data. UTS prediction treats forecasts as a single variable problem, whereas MTS may employ a large number of time-concurred series to make predictions. Neural Network (NN) model could be built to predict the target variable given the other (predictor) variables. In this study, we used Particle Swarm Optimization (PSO) algorithm to optimize performance of NN on a UTS dataset. Our proposed model is validated using x-validation and and use RMSE to measure its performance. The experimental results show that NN performance after optimization using PSO produces good results compared to classical NN performance. This is evidenced by the value of RMSE = 0.410 which is the smallest RMSE value produced. The smaller the RMSE value, the better the model performance. It can be concluded that the proposed method can improve NN performance on UTS data.

1. Introduction

Time series data is a finite sequence of real values extracted from successive observations over a regular time interval with unique chacarteristic [1]. Time series data can be univariate, where at each time instant, only one real value is taken, or multivariate, where many real values are obtained simultaneously [2]. In the case of prediction, both of these data sets can be further analyzed. Time series analysis is an examination of observational data that occurs in a fixed time sequence and time interval, and it consists of methods for analyzing time series data to discover patterns and characteristics [3]. Time series prediction is a complex problem in many scientific fields and industrial sectors, including climate, finance, medicine, and computer science [1][4][5]. The outcomes of time series predictions take the form of future prediction information based on previous data that can be used as a decision-making tool.

Prediction must be carried out using the appropriate methods [6]. The methods for time series prediction are based on the idea that historical data contain inherent patterns that provide useful information for the future description of the phenomenon under consideration [7]. Machine learning prediction methods such as linear regression (LR), support vector machine (SVM), and neural network (NN) have been frequently employed for time series predictions in recent research [8][9]. Because NN has been successfully applied in a variety of fields and has reliable predictive capabilities [10], the focus of this research will be on using it to predict time series data. Furthermore, NN is reported to be one of the most popular and well-established data mining algorithms [8]. The issue with NN is that when initializing weights, NN's prediction performance can suffer significantly when working on noisy and multi-valued datasets [10]. Hence, we need a method for optimizing NN performance.

As time series prediction research has progressed, many studies have been presented that use combined methods to predict time series in order to optimize the performance of their proposed model [9]. One of the popular algorithms used to optimize traditional time series prediction models is particle swarm optimization (PSO) [11][12]. Particle Swarm Optimization (PSO) was proposed by Kennedy and Eberhart in 1995, and is based on the social behaviors of birds and fish during the predatory process. The algorithm is distinguished by its simple process, random initialization, and fewer control parameters, demonstrating strong optimization ability [13]. PSO became the most popular methods for the advantages of easy implementation, which had high precision and fast convergence among numerous of metaheuristic algorithms [14]. According to another study, PSO is one of the most researched population-based stochastic optimization algorithms [15]. Mentioned in another study [16] PSO is one of the most widely used

population-based metaheuristic optimization algorithms, and it has been successfully applied in a wide range of scientific fields, including humanities, engineering, chemistry, medicine, and advanced physics. Since its introduction in 1995, this method has been extensively researched, resulting in the creation of hundreds of versions of PSO as well as numerous theoretical and empirical findings about their convergence and parameterization. Hence, some scholars used heuristic algorithms such as PSO to select NN parameters [14].

NN have been successfully applied in research [6], which considers the problem of forecasting the content of news feeds. A method is proposed that takes into account the space and time relationships of the processed data in greater detail. The method is demonstrated using an example of a NN forecasting system that uses it. Data retrieval from news feeds is followed by special preprocessing, coding, and forecasting of word sets and their interconnections, before highlighting news topics and describing the content of news feeds.

Research [17] proposed a method called NN GAPO+B, which is short for an integration of Genetic Algorithm (GA) based NN parameter optimization and bagging technique, to achieve better prediction performance of software defect prediction. The optimization method is used to select features, while the bagging technique is used to improve the accuracy of software defect prediction. The results revealed an improvement in the accuracy of the proposed method. Furthermore, when used as feature selection in software defect prediction models, there is no significant difference between GA and PSO optimization, indicating that GA and PSO both perform well in optimizing software defect prediction models. Another research [18] proposed a new model to accurately predict long-term monthly gold price fluctuations using a new meta-heuristic method called Whale Optimization Algorithm (WOA) to investigate Multilayer Perceptron Neural Network. The proposed model's results are compared to those of other models such as classical NN, PSO-NN), GA-NN, and Gray Wolf Optimization on NN (GWO-NN). Empirical results show that the WOA-NN model is superior to other models, but the PSO-NN model is also more accurate than Classical NN, GA-NN, and GWO-NN.

The research on predictions that have been mentioned previously mostly uses MTS data. On the other hand, research [19] conducted a performance comparison between predictions on MTS and UTS data. It is frequently assumed that MTS produces better predictions than UTS. However, the MTS data set contains a variety of information, and extracting information useful for predicting the state may be difficult. The results showed that the univariate model predicted better than the multivariate model, especially when used for short-term prediction.

Research [20] has applied a UTS model to predict the number of COVID-19 infected cases that can be expected in the coming days in India using ARIMA model. The results showed an increasing trend in the actual and forecasted numbers of COVID-19 cases with approximately 1500 cases per day, based on available data as on 04th April 2020. Another research [8] used UTS data for Forecasting Temperature and Rain using machine learning algorithms (NN and SVM) instead of traditional algorithms (AR, Auto-ARFIMA, BATS, and Theta). According to the results of the experiments, machine learning algorithms and classical algorithms appear to compete with one another.

According to the above description, the NN model can be built to make predictions on time series data, and PSO is still widely used as a reliable optimization algorithm. Furthermore, it is stated that the UTS prediction model has simpler characteristics than the MTS prediction model and performs better when used correctly. In this paper, we propose using PSO to improve the performance of NN when predicting UTS data. PSO is used to increase the NN attribute's weight. Our proposed model is validated using x-validation, and its performance is measured using root mean square error (RMSE).

2. Research Method

2.1 Datasets

The dataset used in this study was obtained from https://www.kaggle.com/dougcresswell/daily-total-female-births-in-california-1959. This dataset is a UTS that contains 365 records and describes the total number of records of women giving birth in California, USA during 1959. Using x-validation, this dataset is divided into two parts: training and testing data. X-validation steps: (1) Divide the data into *k* subsets of the same size; (2) Use one subset for testing and the rest for training. X-validation will repeat the test *k* times and the measurement result is the average value of *k* times of testing. Daily Total Female Births data can be seen in Table 1.

Tabel 1. Dataset Daily Total Female Births

No	Date	Births
1	1959-01-01	35
2	1959-01-02	32
3	1959-01-03	30
…	.....	..
365	1959-12-31	50

2.2 Preprocessing

There are three stages in preprocessing including set role, normalize, and windowing. The set role operator is used to alter the role of an attribute, specifically an attribute. Data normalization (Normalize) was carried out according to the activation function used, in this study two activation functions were used, namely binary sigmoid and bipolar sigmoid functions. Windowing functions to determine input data and output data in predicting UTS datasets.

2.3 Implementation Model

In this study, experiments will be conducted by feeding UTS data into the NN model, which will then be compared to the PSO-based NN model (NN-PSO) using RapidMiner Studio. The model will be validated using x-validation. *k*-fold cross-validation is another name for x-validation. X-validation is a sampling technique broadly used for evaluating models in machine learning. For time series prediction, the technique searches for the most adequate parameter values for global approach methods [7]. One of the performance measures that predictive models can use is RMSE [18]. RMSE is widely used, and it is an excellent general-purpose error metric for numerical predictions. The model with the lowest RMSE will have the best performance. Figure 1 shows block design of the proposed method.

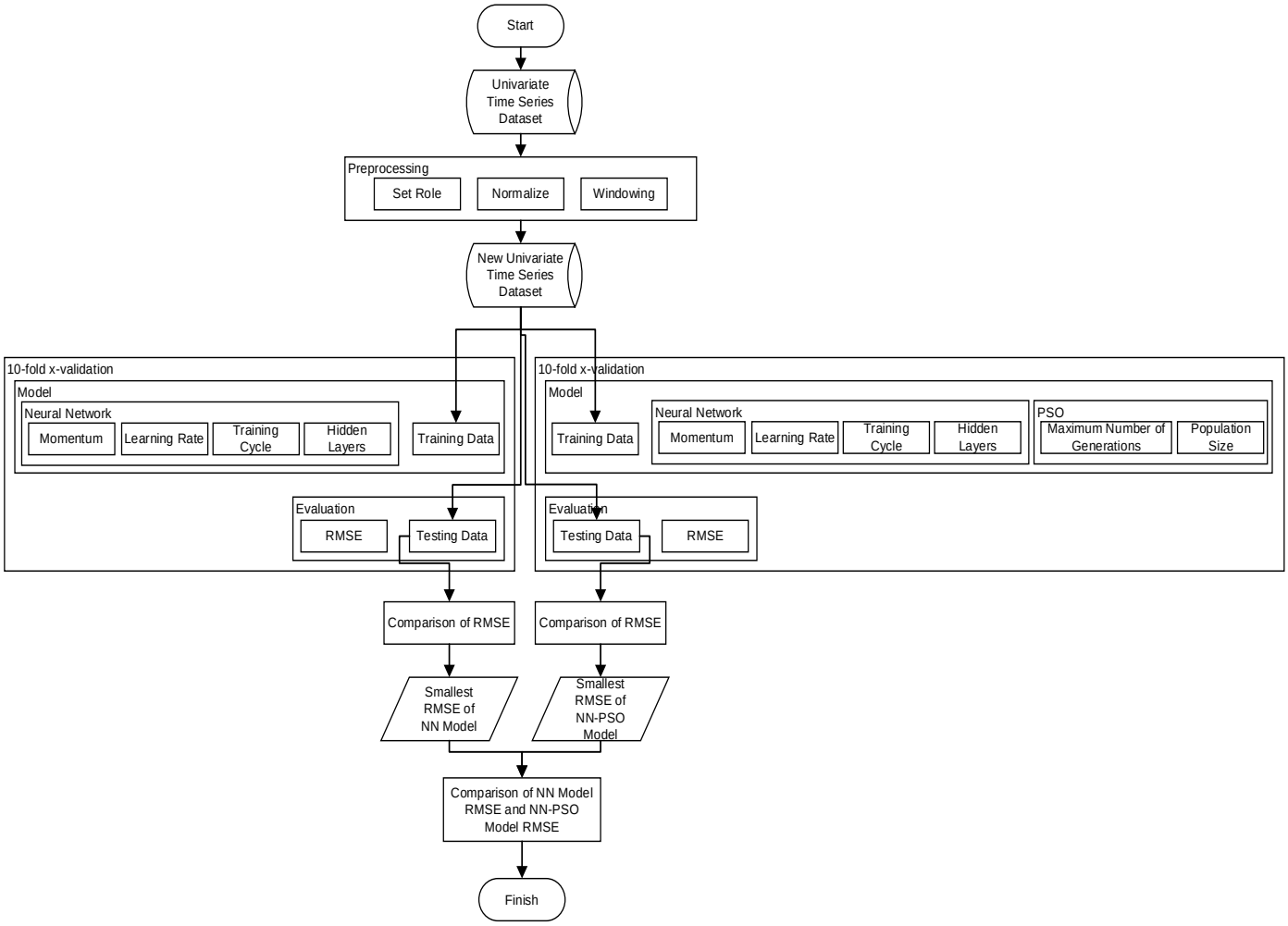

Figure 1. Blok Design of the Proposed Method

The experiment flow in this study begins with the input of the UTS dataset, followed by sequential pre-processing of the data using role sets, normalizing, and windowing. Rapidminer includes all three operators. Assigning the attribute's role is critical for assigning the proper role for the attribute in the dataset; in this case, the set role operator can be used to set the role on the attribute that will be used. The date attribute, which was originally a regular attribute, was converted to a special attribute, namely id. Following the assignment of the role, the dataset is normalized using RapidMiner's normalize operator. The range transformations method is used to normalize data for attribute Births, with attribute values changed to a range of 0 – 1 (min – max). Furthermore, the windowing operator is used in the windowing process. The UTS dataset will then be divided into several input data and one output data using this technique. The

input data is the desired data from the previous time period, and the output data is the data from the next period. The dataset is then trained and tested using the NN and NN-PSO methods. The modeling stage follows, which involves applying the dataset to the NN model and comparing it to the NN-PSO model. PSO is used to increase the NN attribute's weight.

Experiments are conducted by changing or adjusting the NN parameter values in the classic NN model, which includes the training cycle, learning rate, momentum, and hidden layer. The number of training cycles used to train the NN is determined by the training cycle parameter. The learning rate parameter determines how much the weights change at each step. It should not be zero. The momentum simply multiplies the previous weight update by a fraction of the current one. This avoids local maxima and smoothes out optimization directions. The parameter hidden layers specifies the name and size of all hidden layers. With these parameters, the user can define the structure of the NN. PSO parameter settings are also added to the NN-PSO model, such as maximum numbers of generation and population size.

The dataset is divided into two parts for training and testing using x-validation. Following the generation of models with more influential attributes, the error rate of each model was calculated using the root mean square error (RMSE). In addition, the RMSE results of the classic NN and NN-PSO models were recorded for comparison. The model's performance improves as the RMSE decreases.

3. Results and Discussion

The following are the stages of the experiment carried out in this study: (1) preparing the UTS dataset; (2) pre-processing data; (3) designing the NN architecture by inputting the NN parameter values, which include momentum, learning rate, training cycle, and hidden layer; (4) training and testing the NN model, and recording the RMSE results obtained; (5) Creating the NN-PSO architecture by inputting NN parameters such as momentum, learning rate, training cycle, and hidden layer, as well as PSO parameters such as maximum number of generations and population size; (6) Training and testing the proposed model by applying PSO to NN (NN-PSO), then recording the RMSE results obtained; (7) A comparison of the NN model's RSME results with the NN-PSO model's RSME results. Table 2 shows the model parameters that will be used.

Tabel 2. Parameter Model

Model	Parameters	Tested Value
NN	Momentum	0.1 – 0.9
	Learning Rate	0.1 – 0.6
	Training Cycle	10 – 100
	Hidden Layer Size	2 – 5
PSO	Maximum Number of Generations	1 – 10
	Population Size	1 – 10
Windowing	Window Size	1 – 10
X-Validations	Number of Validations	2 – 10

In the first experiment, the NN parameter was initialized in the form of momentum until the best resulting model was identified by the acquisition of the RMSE results with the smallest value. The experimental results of testing the momentum parameters are shown in Table 3. The smallest RMSE gain, 0.1 for the classic NN model and 0.7 for the NN-PSO model, indicates the value of the momentum parameter to be selected for the next experiment.

Table 3. Experimental Results of Testing Changes in Momentum Value

Learning Rate	Training Cycle	Hidden Layers Size	Maximum number of generations	Population Size	Window Size	Number of Validations	Momentum	RMSE NN Classic	RMSE NN-PSO
0.3	10	2	5	5	4	10	**0.1**	**0.154 +/- 0.019**	0.145 +/- 0.014
							0.2	0.155 +/- 0.019	0.145 +/- 0.014
							0.3	0.156 +/- 0.019	0.146 +/- 0.014
							0.4	0.158 +/- 0.019	0.146 +/- 0.014
							0.5	0.161 +/- 0.020	0.145 +/- 0.014
							0.6	0.167 +/- 0.023	0.145 +/- 0.013
							0.7	0.175 +/- 0.031	**0.145 +/- 0.012**
							0.8	0.191 +/- 0.046	0.152 +/- 0.015
							0.9	0.220 +/- 0.069	0.175 +/- 0.034

The same procedure will be used to determine the best value for the remaining parameters such as learning rate, training cycle, hidden layer size, maximum number of generations, population size, window size, and number of validations. Table 4 displays the results of the learning rate test, Table 5 displays the results of the training cycle test, Table 6 displays the results of the hidden layer size test, Table 7 displays the results of the maximum number of generations test, Table 8 displays the results of the population size test, Table 9 displays the results of the window size test, and Table 10 displays the results of the number of validations.

Table 4. Experimental Results of Testing Changes in Learning Rate Value

Momentum	Training Cycle	Hidden Layers Size	Maximum number of generations	Population Size	Window Size	Number of Validations	Learning Rate	RMSE NN Classic	RMSE NN-PSO
NN Classic 0.1	10	2	5	5	4	10	**0.1**	**0.151 +/- 0.016**	0.150 +/- 0.020
							0.2	0.152 +/- 0.016	0.150 +/- 0.019
							0.3	0.154 +/- 0.019	**0.145 +/- 0.012**
NN-PSO 0.7							0.4	0.158 +/- 0.022	0.151 +/- 0.021
							0.5	0.163 +/- 0.024	0.151 +/- 0.021
							0.6	0.169 +/- 0.035	0.168 +/- 0.026

Table 5 shows that the three smallest RMSE values for the Classical NN model are 60, 70, and 80 for Training Cycles. We chose 60 because it had the lowest RMSE in the Training Cycle value when compared to NN-PSO.

Table 5. Experimental Results of Testing Changes in Training Cycle Value

Momentum	Learning Rate	Hidden Layers Size	Maximum number of generations	Population Size	Window Size	Number of Validations	Training Cycle	RMSE NN Classic	RMSE NN-PSO
NN Classic		2	5	5	4	10	**10**	0.151 +/- 0.016	**0.145 +/- 0.012**
0.1	0.1						20	0.149 +/- 0.017	0.146 +/- 0.012
							30	0.149 +/- 0.017	0.147 +/- 0.012
							40	0.149 +/- 0.017	0.147 +/- 0.012
							50	0.158 +/- 0.019	0.159 +/- 0.054
NN-PSO							**60**	**0.145 +/- 0.016**	0.156 +/- 0.056
							70	**0.145 +/- 0.016**	0.157 +/- 0.055
0.7	0.3						**80**	**0.145 +/- 0.016**	0.157 +/- 0.055
							90	0.147 +/- 0.021	0.159 +/- 0.062
							100	0.150 +/- 0.014	0.154 +/- 0.067

Table 6. Experimental Results of Testing Changes in Hidden Layer Size Value

Momentum	Learning Rate	Training Cycle	Maximum number of generations	Population Size	Window Size	Number of Validations	Hidden Layers Size	RMSE NN Classic	RMSE NN-PSO
NN Classic			5	5	4	10	2	0.151 +/- 0.014	0.153 +/- 0.065
0.1	0.1	60					**3**	0.149 +/- 0.016	**0.153 +/- 0.063**
NN-PSO							**4**	**0.147 +/- 0.016**	0.157 +/- 0.067
0.7	0.3	10					5	0.152 +/- 0.021	0.154 +/- 0.061

The maximum number of generations and population size are PSO parameters, not NN parameters. As a result, Table 7 and Tabel 8 only show the results of RSME on the NN-PSO model.

Table 7. Experimental Results of Testing Changes in Maximum Number of Generations Value

Momentum	Learning Rate	Training Cycle	Hidden Layers Size	Population Size	Window Size	Number of Validations	Maximum number of generations	RMSE NN-PSO
0.7	0.3	10	3	5	4	10	1	0.160 +/- 0.083
							2	0.159 +/- 0.067
							3	0.153 +/- 0.063
							4	0.153 +/- 0.063
							5	0.153 +/- 0.063
							6	0.153 +/- 0.063
							7	0.153 +/- 0.063
							8	0.153 +/- 0.063
							9	0.153 +/- 0.063
							10	**0.152 +/- 0.068**

Table 8. Experimental Results of Testing Changes in Population Size Value

Momentum	Learning Rate	Training Cycle	Hidden Layers Size	Maximum number of generations	Window Size	Number of Validations	Population Size	RMSE NN-PSO
0.7	0.3	10	3	10	4	10	1	0.161 +/- 0.066
							2	0.156 +/- 0.076
							3	0.150 +/- 0.075
							4	0.149 +/- 0.070
							5	0.152 +/- 0.068
							6	0.153 +/- 0.063
							7	0.147 +/- 0.068
							8	0.151 +/- 0.059
							9	0.150 +/- 0.069
							10	**0.147 +/- 0.062**

Table 9. Experimental Results of Testing Changes in Window Size Value

Momentum	Learning Rate	Training Cycle	Hidden Layers Size	Maximum number of generations	Population Size	Number of Validations	Window Size	RMSE NN Classic	RMSE NN-PSO
NN Classic							1	0.144 +/- 0.019	0.148 +/- 0.067
0.1	0.1	60	4				2	0.147 +/- 0.014	0.155 +/- 0.067
							3	0.152 +/- 0.018	0.151 +/- 0.077
							4	0.146 +/- 0.019	0.147 +/- 0.062
							5	0.149 +/- 0.063	0.147 +/- 0.015
NN-PSO						10	6	0.156 +/- 0.026	0.148 +/- 0.067
							7	**0.144 +/- 0.018**	**0.141 +/- 0.071**
0.7	0.3	10	3	10	10		8	0.150 +/- 0.065	0.147 +/- 0.015
							9	0.153 +/- 0.061	0.146 +/- 0.025
							10	0.149 +/- 0.017	0.149 +/- 0.059

Table 10 displays the results of the x-validation validation. The dataset is divided into two sections: training data and testing data. X-validation will run the test *k* times, and the measurement result will be the average of the *k* times of testing. In this experiment, the value of *k* is 10. This means that x-validation will be performed 10 times.

Table 10. Experimental Results of Testing Changes in Number of Validations Value

Momentum	Learning Rate	Training Cycle	Hidden Layers Size	Maximum number of generations	Population Size	Window Size	Number of Validations	RMSE NN Classic	RMSE NN-PSO
NN Classic							**1**	0.148 +/- 0.022	**0.140 +/- 0.007**
							2	0.148 +/- 0.022	**0.140 +/- 0.007**
0.1	0.1	60	4				3	0.150 +/- 0.010	0.143 +/- 0.016
							4	**0.141 +/- 0.004**	0.143 +/- 0.011
							5	0.142 +/- 0.008	0.143 +/- 0.014
NN-PSO						7	6	0.147 +/- 0.010	0.145 +/- 0.022
							7	0.147 +/- 0.018	0.143 +/- 0.013
0.7	0.3	10	3	10	10		8	0.141 +/- 0.015	0.145 +/- 0.015
							9	0.144 +/- 0.016	0.144 +/- 0.022
							10	0.147 +/- 0.020	0.143 +/- 0.018

According to Table 10, when the number of validations is set to 1, 2, 3, 6, 7, and 10, the NN-PSO model outperforms the NN Classic model six times, while the NN model only outperforms the NN-PSO model three times when the number of validations is set to 4, 5, and 8. When the number of validations is set to 9, the NN model has the same RSME value as the NN-PSO model. However, the smallest RSME value is obtained when the number of validations is set to 1 or 2 for NN-PSO and 4 for classic NN. Table 11 summarizes the results of testing the chosen parameter values based on previous experiment results.

Table 11. Selected Parameters

Model	Parameters	Selected Value for NN	Selected Value for NN-PSO
NN	Momentum	0,1	0,7
	Learning Rate	0,1	0,3
	Training Cycle	60	10
	Hiden Layer Size	4	3
PSO	Maximum Number of Generations	-	10
	Population Size	-	10
Windowing	Window Size	7	7
X-Validations	Number of Validations	1 and 2	4

4. Conclusion

In this study, the proposed method was applied in the form of the PSO algorithm to improve NN performance on UTS predictions, and based on the results of the experiments, it is proven to improve NN performance by comparing the resulting RMSE values. The smallest RMSE value obtained by using the NN method is 0.141, and the smallest RMSE value obtained by using the NN-PSO method is 0.140. As a result, it is possible to conclude that using the PSO algorithm can improve the performance of NN on UTS predictions. Another important point to note is that the determination of model parameters is also important in this study; both the best NN and NN-PSO models can only be generated by trial-and-error, namely by randomly changing the value of each parameter to produce the best model with the lowest RMSE value. This necessitates a great deal of precision and time in order to perform parameter test experiments many times in order to find the desired parameter value.

References

[1] F. Arias *et al.*, "Auto-adaptive multilayer perceptron for univariate time series classification," *Expert Systems with Applications*, Vol. 181, No. April, 2021. https://doi.org/10.1016/j.eswa.2021.115147

[2] A. Lahreche and B. Boucheham, "A fast and accurate similarity measure for long time series classification based on local extrema and dynamic time warping," *Expert Systems with Applications*, Vol. 168, No. May 2020, P. 114374, 2021. https://doi.org/10.1016/j.eswa.2020.114374

[3] N. Suhermi, D. D. Prastyo, and B. Ali, "Roll motion motion prediction prediction using a hybrid deep learning and ARIMA model," *Procedia Computer Science*, Vol. 144, Pp. 251–258, 2018. https://doi.org/10.1016/j.procs.2018.10.526
[4] C. Koutlis, S. Papadopoulos, M. Schinas, and I. Kompatsiaris, "LAVARNET : Neural network modeling of causal variable relationships for multivariate time series forecasting," *Applied Soft Computing Journal*, Vol. 96, P. 106685, 2020. https://doi.org/10.1016/j.asoc.2020.106685
[5] J. Sun, Y. Yang, Y. Liu, C. Chen, W. Rao, and Y. Bai, "Univariate time series classification using information geometry," *Pattern Recognition*, Vol. 95, Pp. 24–35, Nov. 2019. https://doi.org/10.1016/j.patcog.2019.05.040
[6] V. Osipov, S. Kuleshov, D. Levonevskiy, and D. Miloserdov, "Neural network forecasting of news feeds," *Expert Systems With Applications*, 2020. https://doi.org/10.1016/j.eswa.2020.114521
[7] A. R. S. Parmezan, V. M. A. Souza, and G. E. A. P. A. Batista, "Evaluation of statistical and machine learning models for time series prediction : Identifying the state-of-the-art and the best conditions for the use of each model," *Information Sciences*, Vol. 484, Pp. 302–337, 2019. https://doi.org/10.1016/j.ins.2019.01.076
[8] G. Papacharalampous, H. Tyralis, and D. Koutsoyiannis, "Univariate Time Series Forecasting of Temperature and Precipitation with a Focus on Machine Learning Algorithms : a Multiple-Case Study from Greece," *Water Resources Management*, 2018. https://doi.org/10.1007/s11269-018-2155-6
[9] B. Li, J. Ding, Z. Yin, K. Li, X. Zhao, and L. Zhang, "Optimized neural network combined model based on the induced ordered weighted averaging operator for vegetable price forecasting," *Expert Systems with Applications*, Vol. 168, P. 114232, Apr. 2021. https://doi.org/10.1016/j.eswa.2020.114232aie
[10] H. Quan, D. Srinivasan, and A. Khosravi, "Particle swarm optimization for construction of neural network-based prediction intervals," *Neurocomputing*, Vol. 127, Pp. 172–180, 2014. https://doi.org/10.1016/j.neucom.2013.08.020
[11] B. Jamali, M. Rasekh, F. Jamadi, R. Gandomkar, and F. Makiabadi, "Using PSO-GA algorithm for training artificial neural network to forecast solar space heating system parameters," *Applied Thermal Engineering*, 2019. https://doi.org/10.1016/j.applthermaleng.2018.10.070
[12] N. Noviandi and A. Ilham, "Optimization fuzzy inference system based particle swarm optimization for onset prediction of the rainy season," *Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control*, Vol. 4, No. 1, Pp. 61–70, 2020. https://doi.org/https://doi.org/10.22219/kinetik.v5i1.985
[13] X. Liu, Y. Gu, S. He, Z. Xu, and Z. Zhang, "A robust reliability prediction method using Weighted Least Square Support Vector Machine equipped with Chaos Modified Particle Swarm Optimization and Online Correcting Strategy," *Applied Soft Computing Journal*, Vol. 85, P. 105873, 2019. https://doi.org/10.1016/j.asoc.2019.105873
[14] B. Bai, J. Zhang, X. Wu, G. wei Zhu, and X. Li, "Reliability prediction-based improved dynamic weight particle swarm optimization and back propagation neural network in engineering systems," *Expert Systems with Applications*, Vol. 177, No. November 2020, P. 114952, 2021. https://doi.org/10.1016/j.eswa.2021.114952
[15] P. Singh, S. Chaudhury, and B. K. Panigrahi, "Hybrid MPSO-CNN: Multi-level Particle Swarm optimized hyperparameters of Convolutional Neural Network," *Swarm and Evolutionary Computation*, Vol. 63, No. February, P. 100863, 2021. https://doi.org/10.1016/j.swevo.2021.100863
[16] A. P. Piotrowski, J. J. Napiorkowski, and A. E. Piotrowska, "Population size in Particle Swarm Optimization," *Swarm and Evolutionary Computation*, Vol. 58, No. May, P. 100718, 2020. https://doi.org/10.1016/j.swevo.2020.100718
[17] R. S. Wahono, N. S. Herman, and S. Ahmad, "Neural network parameter optimization based on genetic algorithm for software defect prediction," *Advanced Science Letters*, Vol. 20, No. 10–12, Pp. 1951–1955, 2014. https://doi.org/10.1166/asl.2014.5641
[18] Z. Alameer, M. Abd, A. A. Ewees, H. Ye, and Z. Jianhua, "Forecasting gold price fluctuations using improved multilayer perceptron neural network and whale optimization algorithm," *Resources Policy*, Vol. 61, Pp. 250–260, 2019. https://doi.org/10.1016/j.resourpol.2019.02.014
[19] M. Chayama and Y. Hirata, "When univariate model-free time series prediction is better than multivariate," *Physics Letters A*, Pp. 1–7, 2016. https://doi.org/10.1016/j.physleta.2016.05.027
[20] F. fMohammad and R. Gupta, "ARIMA and NAR based prediction model for time series analysis of COVID-19 cases in India," *Journal of Safety Science and Resilience*, Vol. 1, No. June, Pp. 12–18, 2020. https://doi.org/10.1016/j.jnlssr.2020.06.007