Determining the Neuron Weights of Fuzzy Neural Networks Using Multi-Populations Particle Swarm Optimization for Rainfall Forecasting

M. Chandra C. Utomo, Wayan F. Mahmudy and Syaiful Anam
Universitas Brawijaya
ccahyo@hotmail.com

***Abstract*—Rainfall trends forecasting is essential for several fields, such as airline and ship management, flood control and agriculture and it can be solved by Fuzzy Neural Networks (FNN) approach. However, one of the challenges in implementing the FNN algorithm is to determine the neuron weights. In comparison to Gradient Descent approach, Particle Swarm Optimization (PSO) has been the common approach used to determine neuron weights that result in a more accurate output. However, one of the weaknesses of PSO approach is it tends to convergence after iteration. To overcome this weakness, this study uses a multi-population mechanism to improve the result of PSO approach. The result shows that FNN optimized by PSO with the multi-population mechanism provided a better result than FNN optimized by standard PSO approach and by Gradient Descent approach. Besides, FNN optimized by PSO with multi-population mechanism is capable to produce a better result than the standard Multi-layer Neural Networks optimized by PSO.**

***Index Terms*—Multi-Population; Particle Swarm Optimization; Rainfall Forecasting; Time-series Forecasting.**

I. INTRODUCTION

Rainfall information and forecast have a significant role in some aspects of airline management, shipping, flood control, agricultural, drainage, and meteorological services worldwide. Rainfall rate is a stochastic process, which relies on weather parameters such as temperature average, surface pressure, relative humidity and wind speed [1]. However, the forecast on time series problems is a challenge due to the presence of a frequent increase in error rates almost every time. This is due to the upsurge in crisp-less conditions in the climates and season changes [2].

Vague conditions need special solutions, which we call special 'IF-THEN'. Special 'IF-THEN' can solve the complex cases caused by the crisp-less conditions [3]. Beside special 'IF-THEN', we need the learning algorithm to solve the cases caused by the changes of conditions [4]. Then, to simulate a stochastic process, many scientists worldwide have developed a statistical model of stochastic weather generator to generate a random rainfall rate, which usually brings drawbacks when finding the similarities to their weather data [5]. However, the disadvantages rely on some tacit assumptions in most cases of the system. The uncontrolled atmosphere on any basis may also cause random (vague) behaviors of the system. Furthermore, the behaviors initialization of the random, sensitive, and nonlinear equation can result in inaccurate rainfall forecast as well as making it difficult to solve [3].

II. RELATED WORKS

Predictions of rainfall trends have been generally made using linear regression analysis. Iriany et al. [6] carried out a study using Generalized Space-Time Autoregressive (GSTAR) model by engaging the Ordinary Least Square (OLS) method and approach with the Seemingly Unrelated Regression (SUR) system to make a forecast on Tengger area, East Java. However, this study can only predict the rainfall trend instead rainfall rate.

Optimization of regression approach has been generally made using Particle Swarm Optimization approach. Zhao and Wang [7] applied a support vector regression optimized by particle swarm optimization to forecast the rainfall in Guangxi area. However, this study used only one month of data sets.

For rainfall forecasting, Particle Swarm Optimization has been used to optimize the linear regression using Sugeno Fuzzy Inference System by Utomo and Mahmudy [8] and standard Multi-layer Neural Networks by Sulaiman et al. [9]. The two approaches have their own advantages, in which Fuzzy Inference System is capable to process crisp-less conditions, while Multi-layer Neural Networks is capable to process the cases caused by the changes of conditions. Multi-layer Neural Networks is also a powerful algorithm for malware detection by Huda et al. [10]. Wahyuni et al. [11] proposed an approach of a dynamic system for rainfall prediction in Tengger, Indonesia.

We proposed a hybrid Fuzzy Neural Networks to obtain both advantages and then optimized it using Particle Swarm Optimization with the multi-population mechanism. The Particle Swarm Optimization is necessary to determine the neuron weights after taking into consideration the challenges in building the Fuzzy Neural Networks [12]. We also used Utomo and Mahmudy [13] studies. The multi-population mechanism is necessary to improve the observation of Particle Swarm Optimization in wide area.

Another reason for proposing our method is that Fuzzy Neural Networks are usually used to solve trends forecasting case. For example, Shen, Shen and Chang [14] proposed a Fuzzy Neural Networks for water flow estimation in the drains during wet weather. Another study was conducted by Corani and Guariso [15] that proposed a hybrid fuzzy model and artificial neural networks for river flood prediction.

III. RESEARCH METHOD

The Fuzzy Neural Networks (FNN) algorithm is as shown in Figure 1. The figure shows a flowchart of FNN algorithm that consists of four steps: Firstly, the algorithm received several forecasting parameters. While PSO strategy is optimizing the FNN approach, the datasets are read at this stage. Second, the parameters or datasets are processed by fuzzy logic, which uses fuzzy rules and fuzzy sets. Third, the parameters or datasets are processed by Multi-layer Neural Networks, and the fourth step involves the derivation of the forecasting result. The descriptions of developing the FNN containing fuzzy rules, fuzzy sets, Multi-layer Neural Networks and the optimization of FNN using Particle Swarm Optimization with multi-population mechanism are presented in this subsection.

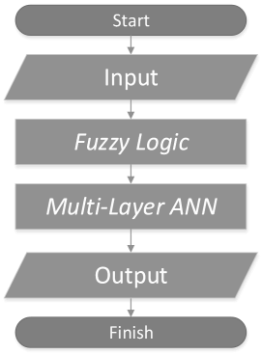

Figure 1: Fuzzy Neural Networks Flowchart

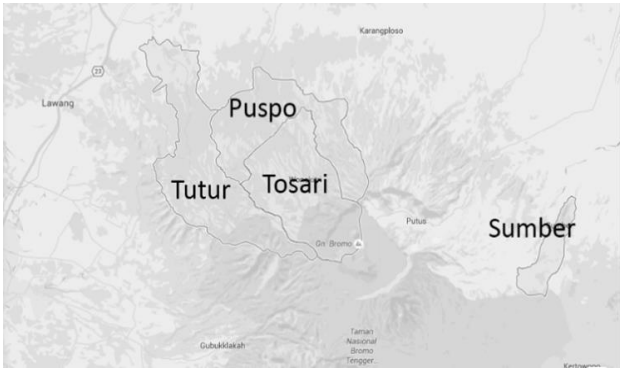

Figure 2: Puspo, Sumber, Tosari, and Tutur Locations on Google Maps

A. Datasets

The rainfall trends datasets used in this study were from Iriany et al. [6] and comparative analysis was carried out among the datasets. The trends were collected from four stations at Tengger area, East Java, namely Puspo, Sumber, Tosari, and Tutur. Each location can be seen in Figure 2. We used these stations because the areas can affect each other. The collection was conducted and averaged each 10 days between the period from 2005 to 2014. Further, the daily data were recorded [16] in a time-series format [17]. The rainfall pattern can be seen as depicted in Figure 3. The rainfall trends data sets were obtained and recorded by the Body of Meteorology, Climatology and Geophysical (*Badan Meteorologi, Klimatologi, dan Geofisika (BMKG)).*

B. Fuzzy Rules

Drawn from a study by Utomo and Mahmudy [13], we derived a sample of the fuzzy rules as shown in Table 1. All of the Fuzzy Rules has 16-column parameters and 16-row

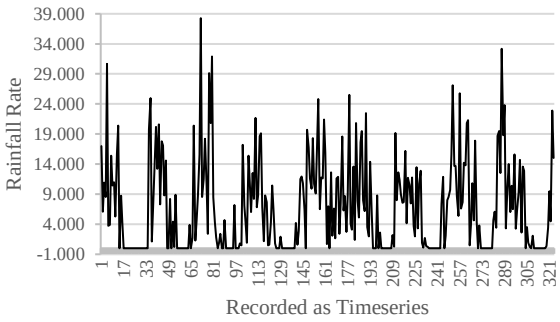

Figure 3: Rainfall Pattern

rules in total. The (*t*-1) means one day ago, (*t*-2) means two days ago, (*t*-18) means eighteen days ago, and (*t*-36) means thirty-six days ago. We used these parameters because the(*t*-36) is often the same as (*t*-0), and it is often significantly different from (*t*-0), and if (*t*-2) is nearly the same as (*t*-1), then (*t*-0) is often different. The table shows that the alphabet S represents a sunny condition, and alphabet R represents rainy condition. The variables in the table have AND relationship, therefore the first row can be read as,

IF *(t-1) = sunny* **AND** *(t-2) = sunny* **AND** *(t-18) = rain*
AND *(t-36) = sunny*
THEN *(t-0) =* $f_1\ (z_{1\text{-}17})$

According to each condition on the rows, the result obtained from an equation on the (*t*-0) column. The equation on the (*t*-0) column is a zero order equation (or called as a linear regression equation) as in Equation (1), where P is Puspo, S is Sumber, O is Tosari, U is Tutur, and z_1 until z_{17} is a linear regression constant variable that can be seen in the reference [13].

Table 1
Fuzzy Rules

t-0	$P_{t\text{-}1}$	$P_{t\text{-}2}$	$P_{t\text{-}18}$	....	$U_{t\text{-}36}$
$f_1\ (z_{1\text{-}17})$	S	S	R	....	S
$f_2\ (z_{1\text{-}17})$	R	R	S	....	R
$f_3\ (z_{1\text{-}17})$	S	R	R	....	S
$f_4\ (z_{1\text{-}17})$	R	S	S	....	R
$f_5\ (z_{1\text{-}17})$	S	S	S	....	R
....	....	....	....	....	....
$f_{16}\ (z_{1\text{-}17})$	R	R	R	....	S

$$f(z_{1-17}) = z_1 + z_2 * P_{t-1} + z_3 * S_{t-1} + z_4 * O_{t-1} + \ldots + z_{16} * O_{t-36} + z_{17} * U_{t-36} \quad (1)$$

C. Fuzzy Sets

The fuzzy sets are required for normalization that converts rainfall rate into the fuzzy rate. From the study of Utomo and Mahmudy [13], we produced the fuzzy set as shown in Figure 4. Figure 4 shows sunny line as Equation (2) and rainy line as Equation (3) to determine the association of the fuzzy rules that have been mentioned above. The 16 inputs variables in each column are processed by the fuzzy rules in each row. If the value of the fuzzy rules is 0 (sunny), then the normalization process uses Equation (2), otherwise, if the value of the fuzzy rules is 1 (rain) then the normalization process uses Equation (3), where x is the input variable.

$$f(x) = \begin{cases} 1 & x \le 0 \\ \dfrac{40 - x}{40 - 0} & 0 < x < 40 \\ 0 & 40 \le x \end{cases} \quad (2)$$

$$f(x) = \begin{cases} 0 & x \le 0 \\ \dfrac{x - 0}{40 - 0} & 0 < x < 40 \\ 1 & 40 \le x \end{cases} \quad (3)$$

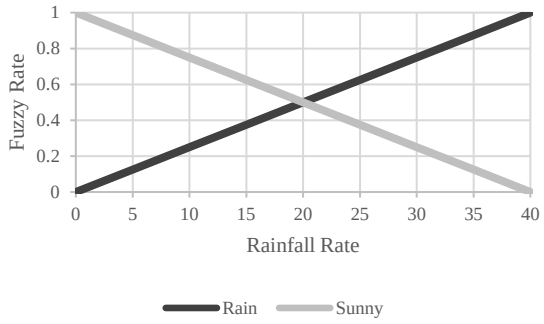

Figure 4: Fuzzy Set

D. Multi-Layer Neural Networks

The simple example of Multi-Layer Neural Networks architecture can be seen in Figure 5. As shown in Figure 5, the architecture has two input neurons (X1, X2) with one bias neuron (B1) on input layer, three hidden neurons (Z1, Z2, Z3) with one bias neuron (B2) on a hidden layer, and lastly one output neuron (Y1) on output layer. Bias neuron always has the value of one. This study optimizes the number of hidden layer and hidden neurons (Z1, Z2, …., Zn) to be used. Between the neuron, there is one or are some lines following the name as V01, V02, V03, V11, V12, V13, V21, V22, V23, W01, W21, W21, and W31. Each line has a weight to process the value to the next neuron. The processing example follows Equation (4) that looks like Equation (1). The Y1 neuron can be changed to Z1, Z2, and Z3 neuron following the previous neuron and weight. This study also optimizes the weight of each line.

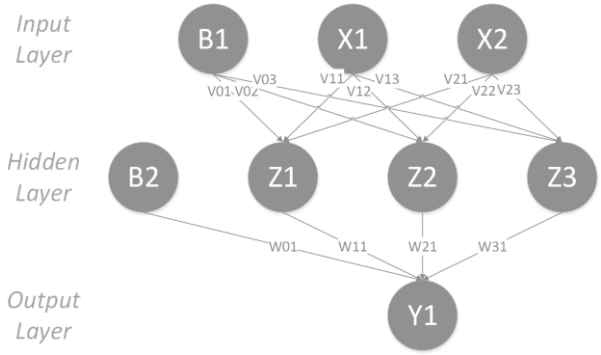

Figure 5: The simple example of Multi-Layer Neural Networks

$$Y1 = B2 * W01 + Z1 * W11 + Z2 * W21 + Z3 * W31 \quad (4)$$

According to Fausett's study [4], one hidden layer is usually sufficient in most case but in some case, it is better to have two hidden layers. Based on this opinion, this study compared the differences in performance between the usage of one hidden layer and two hidden layers. According to Haykin's study [12], the number of the hidden neuron is between 2 to 9. According to Heaton's study [18], the number of hidden neuron should be based on the following rules:

i. between the amount of output layer to the amount of input layer;
ii. 2/3 of the input neuron then plus with the output neuron,
iii. not over than 2 times the amount of input neuron.

Based on the above conditions, this study attempts to find the best requirement when using hidden neuron and determine the weights of the neuron using Particle Swarm Optimization.

Fuzzy Rules has 16-row rules that have 16 output values. Based on the16 output values, we used 16 input neurons on multi-layer neural networks. Taking into consideration of 1 as the result value of a requirement, we used 1 output neuron on multi-layer neural networks. By using 16 input neurons, 1 output neuron, and according to a study conducted by Heaton [18], we obtained 12 hidden neurons. By using 16 input neurons, one hidden layer with 12 hidden neurons, and 1 output neuron, we determined the neuron weights using Particle Swarm Optimization.

E. Optimization

One challenge of building an FNN accurately was determining neuron weights. To deal with this issue, we employed Particle Swarm Optimization (PSO). PSO works by representing several solutions of neuron weights as an array [8]. The solution was defined as particles and several solutions were defined as particle sizes. We also used a multi-population mechanism to improve the result.

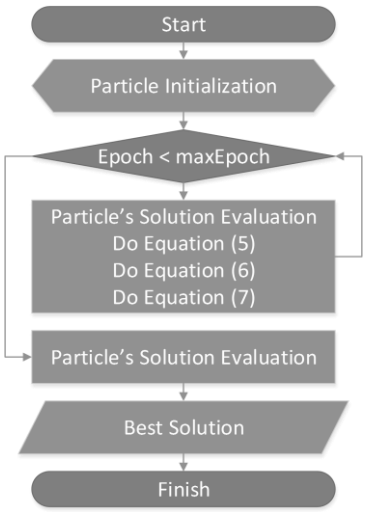

Figure 6: PSO Algorithm FlowChart

The PSO algorithm follows the sequence as shown in Figure 6. To obtain the new solution, we update each particle with Equation (5), Equation (6), and Equation (7) in each iteration,

$$V_i(t) = \theta V_i(e-1) + C_1 R_1 \left(pBest_i - X_i(e-1)\right) + C_2 R_2 \left(gBest_i - X_i(e-1)\right) \quad (5)$$

$$X_i(e) = V_i(e) + X_i(e-1) \quad (6)$$

$$\theta(i) = \theta_{max} - \left(\frac{\theta_{max} - \theta_{min}}{e_{max}}\right) e \quad (7)$$

where:
(X) is an array of the solution.
(V) is an array of solution modifier (of called velocity).
(i) is an array's index.
(e) is an iteration progress.
(θ) is an inertia moment.
(C_1) and (C_2) is a modifier's constant.
(R_1) and (R_2) is a random floating value.
(*pBest*) is an array of the best solution from the (X) itself.
(*gBest*) is an array of the best solution from its population.

This study represents the requirement of using hidden layer, hidden neuron in each hidden layer, and weights on each neuron in an array. Since this study determines one or two hidden layers only, the first index of the array contains the amount of neuron on the first layer and the second index of the array contains the amount of neuron of the second layer. Both of them cannot contain zero value. However, if one of them contains zero value, it means FNN algorithm uses one hidden layer only. Each index of the array on the next layer contains the weight for each neuron. The ordered of the weight is according to the ordered of the neuron. The length of the array is two plus the amount of weight requirement for FNN algorithm. Then, to measure the effectiveness of the FNN algorithm, we use RMSE equation as Equation (8),

$$y = \sqrt[2]{\sum_{t=1}^{n}(x'_t - x_t)^2 / n} \quad (8)$$

where:
(y) is the RMSE result.
(x) is the real rainfall rate.
(x') is the prediction of rainfall rate.
(t) is an index of datasets.
(n) is a datasets size.

F. Multi-Populations Mechanism

In the multi-population mechanism, several particles were created as many as determined particle sizes and several populations were created as many as determined population sizes. In this study, we use double populations and each population has 70 particles. After each of the several iterations, a migration mechanism was performed. In the migration mechanism, the best particle from each population was switched. In this study, the migration mechanism was performed for each of the 10 iterations in 10000 iterations. Alongside the migration mechanism, random injection mechanism was also performed. In the random injection mechanism, several particles were selected and replaced with new initialization. In this study, 2% of the particle size was selected randomly and replaced with a new random initialization.

IV. Results and Analysis

Because Particle Swarm Optimization is one of the heuristic algorithms, which provides a stochastic result, we run it for100 times to get an average result [19].

A. Hidden Layer Requirement Test

According to Fausett [4], we tested the performance of using one and two hidden layers. To measure the performance of each requirement, we used Root Mean Squared Error equation [13]. A better performance was provided when the minimum RMSE was obtained. The result of the hidden layer requirement test is shown in Table 2.

The table shows that using two hidden layers with RMSE as 9.607892 is better than using 1 hidden layer with RMSE as 9.608295. But in our opinion, using 1 hidden layer is better than 2 hidden layers. This is because RMSE as 9.607892 using 2 hidden layers is not significantly different from RMSE as 9.608295 using 1 hidden layer. The minimum of using hidden layer results in a lighter algorithm and this is the reason for using two hidden layers while having closed RMSE with one hidden layer. Therefore, we preferred to use one hidden layer only.

Table 2
Hidden Layer Requirement Test Result

Hidden Layer	RMSE
1	9.608295
2	9.607892

B. Ranged Hidden Neuron Requirement Test

In this section, we tried to find the best requirement of using hidden neuron. According to Haykin [12], we tried to find the best requirement between 2 to 9 of the hidden neurons. According to Heaton [18], we tried to find the best requirement between 1 to16, and between 1 to 32 hidden neurons and compared them using 12 hidden neurons.

To measure the best requirement on each distribution, we used mode. In statistics, the mode is the value that most frequently occur or repetitive in an array or range of data. It means that the most frequently occurring or repetitive is the most proven best requirement. To measure the performance of each requirement, we used Root Mean Squared Error equation. The minimum RMSE was obtained, the better performance was provided. The result of hidden neuron requirement test is shown in Table 3.

Table 3 shows that the mode of 2 – 9 range requirement is 8, the mode of 1 – 16 range requirement is 12, and the mode of 1 – 32 range requirement is 26. When we looked at the minimum RMSE and compared them to 12 and 26 hidden neurons with RMSE as 9.608905 and 9.609291, the requirement of using 8 hidden neurons with RMSE as 9.607668 is better than the previous condition. It is abnormal because conceptually, the usage of more hidden neuron results in the achievement of minimum RMSE. We assumed that the wider the range of data, the more difficult to optimize using PSO. It is proven with the comparison with 12 range requirement only. In this section, we describe the comparison of several hidden neuron requirements individually.

Table 3
Best Hidden Neuron Requirement Test Result from Ranged of Data

Range of Data	Mode	RMSE
2 – 9	8	9.607668
1 – 16	12	9.608905
12	12	9.606
1 – 32	26	9.609291

.

C. Individually Hidden Neuron Requirement Test

In this section, we test the performance of using 12 hidden neurons than 9, 16, and 32 hidden neurons respectively. To measure the performance of each requirement, we used Root Mean Squared Error equation. The minimum RMSE has been obtained and the better performance was provided. The result

of neuron weights optimization test is shown in Table 4 and Figure 7.

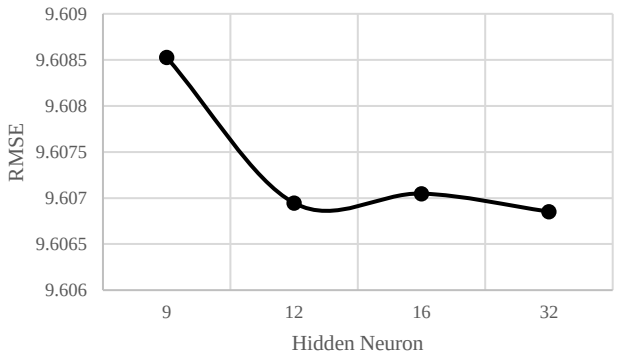

Figure 7: Best Hidden Neuron Requirement Test Result Individually

Table 4 shows that using 12 hidden neurons with a RMSE as 9.605500 is better than using 9, 16, and 32 hidden neurons with RMSE as 9.605731, 9.605515, and 9.605516 respectively. Conceptually, the usage of the more hidden neuron, the minimum of RMSE must be obtained. However, the test result in Table 4 shows that using 16 and 32 hidden neurons provides a worse solution than using 12 hidden neurons. We assumed that the use of more hidden neuron results in the difficulty to determine the neuron weights caused by its complexity. The complexity can cause the lack of proper values to be used. Besides, as shown in Figure 7, RMSE used together with all of the 12 hidden neurons is not significantly different from the RMSE using 16 and 32 hidden neurons since PSO is a stochastic algorithm [19]. In addition, as shown in Figure 8, the pattern between 12, 16, and 32, except 9 hidden neurons requirement were identified similarly. Therefore, we preferred to use 12 hidden neurons.

Table 4
Best Hidden Neuron Requirement Test Result Individually

Hidden Neuron	RMSE
9	9.608526
12	9.606946
16	9.607048
32	9.606853

D. Comparison

To know the value of this study, we compared it with a standard multi-layer neural network optimized by PSO strategy and FNN optimized by Gradient Descent strategy. The RMSE comparison of each approach is shown in Table 5.

Table 5
RMSE Comparison

Approach	RMSE
FNN-PSO with Multi-PopSize	9.605
FNN-PSO with Single-PopSize	9.614
Multi-layer ANN-PSO	9.614
Multi-layer ANN-GD	10.104
FNN-GD	14.765

Table 5 shows that the proposed FNN optimized by PSO (FNN-PSO with Multi-PopSize) (with a RMSE as 9.6053003) is better than FNN optimized by Gradient Descent (FNN-GD) (with a RMSE 14.764963). This study proved that PSO strategy was capable of optimizing FNN approach and performed better than Gradient Descent strategy. The proposed FNN optimized by PSO (FNN-PSO with Multi-PopSize) is also better than the standard multi-layer neural networks optimized by PSO (Multi-layer ANN-PSO) (with a RMSE as 9.614299). Therefore, PSO with multi-populations mechanism is better to optimize FNN approach (FNN-PSO with Multi-PopSize) than PSO with a single population (FNN-PSO with Single-PopSize) (with a RMSE as 9.6137196).

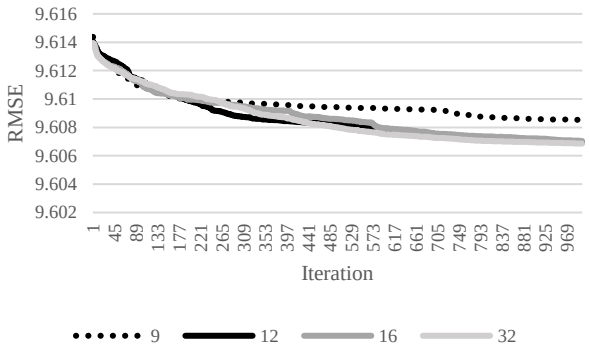

Figure 8: Hidden Neuron Requirement per Iteration Test Result

However, based on Figure 8, we were not satisfied with the results. Figure 8 shows that the PSO with multi-populations was converged after 1000 iteration and it was difficult to provide a better result. In the following study, we tried to use another evolutionary algorithm to optimize FNN weight for rainfall trends forecasting.

V. Conclusion

The comparison between the standard Multi-layer Neural Networks optimized by PSO with RMSE as 9.61, the hybrid FNN optimized by PSO with RMSE as 9.6 shows that it was capable of providing a better result.

The comparison with the Gradient Descent strategy with RMSE as 14.76 and PSO strategy with RMSE as 9.6 proved that the optimization of FNN algorithm performed better. This is because PSO strategy processes several solutions than the Gradient Descent strategy that processes only one solution in the same iteration. Besides, the comparison with simple PSO strategy with a RMSE as 9.61 and PSO with a multi-population mechanism with a RMSE as 9.6 was better to optimize FNN approach. In the next study, we will use another evolutionary algorithm to optimize FNN algorithm for rainfall trends forecasting problem.

References

[1] J. Patel and F. Parekh, "Forecasting Rainfall Using Adaptive Neuro-Fuzzy Inference System (ANFIS)," *Int. J. Appl. Innov. Eng. Manag.*, vol. 3, no. 6, pp. 262–269, Jun. 2014.

[2] W. T. Zaw and T. T. Naing, "Modeling of Rainfall Prediction Over Myanmar using Polynomial Regression," presented at the International Conference on Computer Engineering and Technoloogy (ICCET)., 2009, pp. 316–320.

[3] P. Guhathakurta, "Long-range monsoon rainfall prediction of 2005 for the districts and sub-division Kerala with artificial neural network," *Curr. Sci.*, vol. 90, no. 6, pp. 773–779, 2006.

[4] L. V. Fausett, *Fundamentals Of Neural Network: Architecture, Algorithms, and Applications*, International Editions. Prentice-Hall, 1994.

[5] D. S. Wilks, "Multisite generalization of a daily stochastic precipitation generation model," *J. Hydrol.*, vol. 210, no. 1, pp. 178–191, 1998.

[6] A. Iriany, W. F. Mahmudy, S. Handoyo, A. D. Sulistyono, and S. K. Nisak, "GSTAR-SUR Model for Rainfall Forecasting in Tengger Region, East Java," in *Planning for Environmental Sustainability for the Well Being of Future Humanity*, Malang, Indonesia, 2015.

[7] S. Zhao and L. Wang, "Support Vector Regression Based on Particle Swarm Optimization for Rainfall Forecasting," presented at the 3rd International Joint Conference on Computational Science and Optimization (CSO), 2010, pp. 484–487.

[8] M. C. C. Utomo and W. F. Mahmudy, "Optimization of Fuzzy's Rules for Rainfall Forecasting using Particle Swarm Optimization," *Int. J. Eng. Inform.*, no. 2016.

[9] J. B. Sulaiman, H. Darwis, and H. Hirose, "Monthly Maximum Accumulated Precipitation Forecasting Using Local Precipitation Data and Global Climate Modes," *J. Adv. Intell. Intell. Inform.*, vol. 18, no. 6, pp. 999–1006, Nov. 2014.

[10] F. A. Huda, W. F. Mahmudy, and H. Tolle, "Android malware detection using backpropagation neural network," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 4, no. 1, 2016.

[11] I. Wahyuni, P. F. E. Adipraja, W. F. Mahmudy, and A. Iriany, "Rainfall Prediction in Tengger Indonesia: A System Dynamic Approach," *Int. J. Intell. Eng. Syst.*, 2017.

[12] S. Haykin, *Neural Networks. A Comprehensive Foundation*, 2nd ed. Singapore: Pearson Prentice Hall, 2005.

[13] M. C. C. Utomo and W. F. Mahmudy, "Optimization of Sugeno Fuzzy Inference System's Rules for Rainfall Forecasting," *IAENG*, no. 2016.

[14] J. Shen, W. Shen, J. Chang, and N. Gong, "Fuzzy Neural Network for Flow Estimation in Sewer Systems During Wet Weather," *Water Environ. Res.*, vol. 78, no. 2, pp. 100–109, Feb. 2006.

[15] G. Corani and G. Guariso, "Coupling Fuzzy Modeling and Neural Networks for River Flood Prediction," *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 35, no. 3, pp. 382–390, Aug. 2005.

[16] K. G. Abistado and C. N. Arellano, "Weather Forecasting Using Artificial Neural Network and Bayesian Network," *J. Adv. Intell. Intell. Inform.*, vol. 18, no. 5, pp. 812–817, Sep. 2014.

[17] J. B. Yabuuchi and J. Watada, "Fuzzy Autocorrelation Model with Confidence Intervals of Fuzzy Random Data," *J. Adv. Intell. Intell. Inform.*, vol. 18, no. 2, pp. 197–203, Mar. 2014.

[18] J. T. Heaton, *Introduction to Neural Networks for Java*, 2nd ed. Heaton Research, Inc., 2008.

[19] D. Novitasari, I. Cholissodin, and W. F. Mahmudy, "Hybridizing PSO With SA for Optimizing SVR Applied to Software Effort Estimation," *Telkomnika*, vol. 14, p. 1, 2016.