Monthly rainfall prediction based on artificial neural networks with backpropagation and radial basis function

Ian Mochamad Sofian [a,1], Azhar Kholiq Affandi [a,2], Iskhaq Iskandar [a,3,*], Yosi Apriani [b,4]

[a] Department of Physics, University of Sriwijaya, Palembang, Indonesia
[b] Department of Electrical Engineering, University of Muhammadiyah Palembang, Palembang, Indonesia
[1] ian.msof@gmail.com; [2] azharka@unsri.ac.id; [3] iskhaq@mipa.unsri.ac.id; [4] yosi_apriani@um-palembang.ac.id
* corresponding author

ARTICLE INFO

Article history
Received May 24, 2018
Revised June 26, 2018
Accepted June 28, 2018

Keywords
Prediction
Rainfall
BPNN
RBFNN

ABSTRACT

Two models of Artificial Neural Network (ANN) algorithm have been developed for monthly rainfall prediction, namely the Backpropagation Neural Network (BPNN) and Radial Basis Function Neural Network (RBFNN). A total data of 238 months (1994-2013) was used as the input data, in which 190 data were used as training data and 48 data used as testing data. Rainfall data has been tested using architecture BPNN with various learning rates. In addition, the rainfall data has been tested using the RBFNN architecture with maximum number of neurons K = 200, and various error goals. Statistical analysis has been conducted to calculate R, MSE, MBE, and MAE to verify the result. The study showed that RBFNN architecture with error goal of 0.001 gives the best result with a value of MSE = 0.00072 and R = 0.98 for the learning process, and MSE = 0.00092 and R = 0.86 for the testing process. Thus, the RBFNN can be set as the best model for monthly rainfall prediction.

1. Introduction

Indonesia as the largest archipelagic state, is one of most vulnerable countries to the negative impact of global climate change. Global climate change models predict all areas in Indonesia would suffer changes in patterns and intensity of rainfall [1]. In addition, the trend of changes may show significant variations for monthly, seasonal, and even inter-annual time-scale [2]. Therefore, the occurred variations make the rainy season and the dry season become more uncertain and difficult to predict.

The most noticeable negative impacts of changes in patterns and intensity of rainfall are forest fires and floods. In 1994, 1997, 2002 and 2015, forest and peat fires struck South Sumatera causing hundreds of thousands of hectares of concession and conservation land damaged [3], [4]. Recorded presence of fire in South Sumatra reached 40% of total fires throughout Indonesia (2.1 million hectares). In addition, smoke generated from forest fires affects the health of the majority of people with acute respiratory infections. Therefore, this research builds an accurate method for rainfall prediction.

In general, there are three types of prediction methods: physical law [5], statistical analysis [6], and soft computing [7]. while the third one is based on numerical model. The first method involves the study of the rainfall processes in order to model the underlined physical law. However, this method is very difficult to applied because the rainfall is influenced by a number of hydrological parameters and is limited in both the spatial and temporal dimensions [8], [9]. Thus, this method requires a various complicated calculation. The second method (statistical method) is including the Simple Linear Regression (SLR), Exponential Smoothing (ES), Autoregressive Moving Average (ARMA), Autoregressive Integrated Moving Average (ARIMA), and Generalized Autoregressive Conditional

Heteroskedasticity (GARCH). Note that the ARIMA model is used by the Agency for Meteorology, Climatology, and Geophysics (BMKG) for short-term weather predictions. Nevertheless, the statistical method has a limitation in which it is not suitable for nonlinear time series data [10]. It is known that the rainfall contains of nonlinear as well as stationary data. On the other hand, the soft computing method could deal with both linear and nonlinear data. For example, the Artificial Neural Network (ANN) is a soft computing method that has capability to identify nonlinear data pattern with learning approach [11]. This method is quite simple and practical to use in the case of prediction, but also has good accuracy.

Many researchers predicted rainfall using ANN. Abhishek et al. [12], have used ANN to develop weather forecasting. The training parameters (activation function) are applied differently in three different architectures. In view of the Mean Square Error (MSE) value, that overall architecture yields the value of MSE is still somewhat large. Mislan et al. [13], applied ANN techniques with backpropagation algorithm to predict rainfall. The network model is made into 3 architectures. From the test results show that the value of MSE produced by each architecture is still somewhat large but better than previous research. Wahyuni et al. [14], have used ANN with backpropagation algorithm to develop rainfall prediction models. The data used were taken from the period 2005-2014, where 50% was used for the training process, and 50% for the test process. The focus of the research is to search for the most optimal model parameters. For this purpose, this research applies different parameters in each built model, including the number of hidden layers, learning rate, and epoch. The value of MSE is still large.

ANN used in the early research, showed inaccurate prediction results. To get a better prediction with a small error value, it is necessary to present a solution to the problem. In this research, we apply different BPNN algorithms. The difference lies in the great architecture (each 100 neurons in each of six hidden layers). The use of training parameters will vary, such as the activation function between hidden layer connections (*logsig*, *tansig*, and *purelin*), and learning rate (0.05, 0.1, and 0.3). The results obtained from BPNN will be compared with other algorithms, namely Radial Basis Function Neural Network (RBFNN).

2. Method

2.1. Data and Research Area

The data used in this study were obtained from the National Oceanographic and Atmospheric Administration (NOAA) (https://www.esrl.noaa.gov/psd/data/gridded/data.gpcc.html). The data have temporal resolution of monthly and spatial resolution of 0.5° in both latitude and longitude. The data cover a period of January 1994 – December 2013. Meanwhile, the research area is bounded by a coordinate of 2.5°-3° S and 104,5°-105° E. These coordinates cover most of the Palembang City and a small part of the Banyuasin Regency, South Sumatera Province, Indonesia

2.2. Pre-processing Data

The downloaded data cover a globe and are in the form of a NetCDF file. We extracted the data using the Grid Analysis and Display System (GRADS) software by selecting the data on the research area only. Afterward, the selected data were reprocessed using software MATLAB R2008a and MS. Excel to get final (as presented in Table 1).

These final data are, then, become input pattern for the Artificial Neural Network (ANN). Note that prior to use in the process of training and testing ANN, the data were normalized using sigmoid function (1) in order to get the rainfall value between 0-1.

$$x' = \frac{0.8(x-a)}{b-a} + 0.1, \tag{1}$$

here, a is the minimum value and b is the maximum value. Furthermore, x is the original data, while x' is normalized data. The original time series of the nonlinear rainfall data are shown in Fig. 1.

Table 1. Original rainfall data 1994-2013

Year	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003
Jan	110.76	300.34	123.83	110	200.05	289.01	185.62	220.74	245.17	297.44
Feb	166.78	307.72	281.14	100.43	175.38	178.64	218.55	190.27	34.13	359.97
Mar	245.94	317.7	108.03	335.03	210.32	309.25	285.09	294.29	244.66	172.19
...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...
Nov	459.77	324.05	278.11	111.01	313.29	296.76	203.1	450.78	252.01	145.78
Dec	236.01	209.62	224.1	183.41	235.53	183.42	260.11	331.19	324.55	251.56
Year	**2004**	**2005**	**2006**	**2007**	**2008**	**2009**	**2010**	**2011**	**2012**	**2013**
Jan	303.36	100.35	234.75	287.45	185.62	159.8	137.25	267.44	75.67	164.39
Feb	136.85	74.37	285.93	118.89	125.2	304.23	335.05	108.55	182.73	312.35
Mar	369.83	321.65	183.99	193.93	337.41	249.27	212.97	147.3	168.2	395.13
...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...
Nov	293.44	334.26	199.02	153.99	257.94	389.27	294.92	409.46	298.29	192.21
Dec	257.85	220.56	237.65	306.21	308.75	317.3	267.22	236.29	183.17	266.33

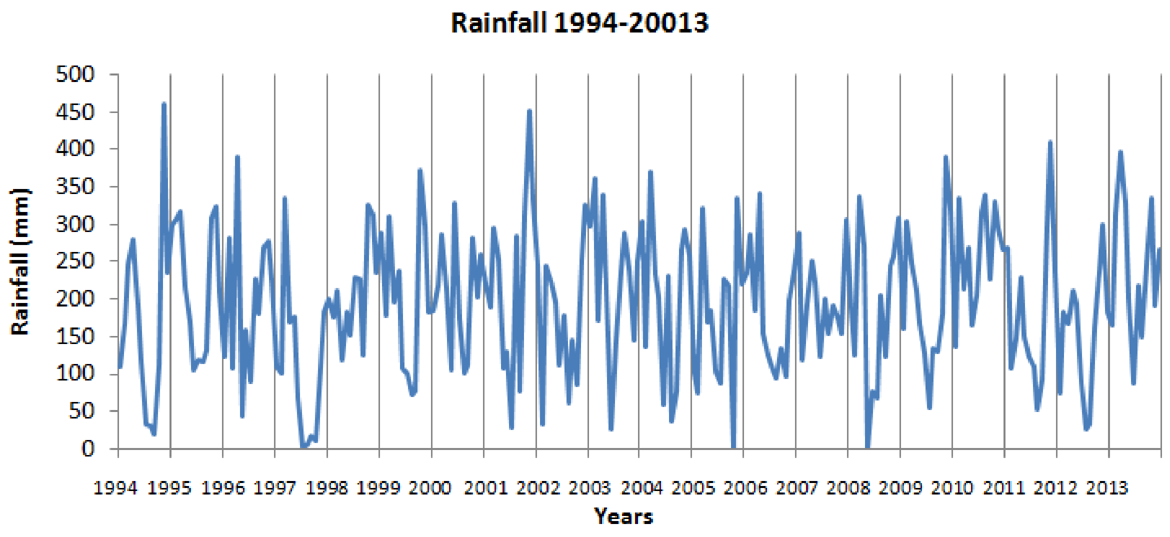


Fig. 1. Time series of the monthly rainfall averaged over the area 2.5°-3° S, 104,5°-105° E

2.3. Define Input Patterns, Training Data, and Testing Data

The ANN-rule determined input data consists of two neurons *P = [p (t-2), p (t-1)]*, while the output data consist of 1 neuron, *p '(t)*. These input data were obtained from the original rainfall data after a normalization using (1). Two *P* neurons showed a pattern of training data, where the first month (Jan 1994) and second month (Feb 1994) became input for the target in the third month (Mar 1994). As for the 4th month target (Apr 1994) then the 2nd month (Feb 1994) and the 3rd month (Mar 1994) into input data. This rule pattern continues to apply until it reaches the 240th month data (Dec 2013) as the target and the 238th month (Oct 2013) as well as the 239th (Nov 2013) as input. Using this technique, we will get 238 data patterns or group (Table 2). In order to get a good result, the final rainfall data were divided into two part, namely the training data and the testing data. In this study, the training data are 80% of the total data (pattern 1-190), while the testing data used the remaining 20% (pattern 191-238).

The data as presented in Table 2 will be applied to both the Backpropagation Neural Network (BPNN) and Radial Basis Function Neural Network (RBFNN) algorithms.

Table 2. Rainfall data after normalization

	Group	Input neuron P=[p(t-2),p(t-1)]		Target neuron T		Group	Input neuron P=[p(t-2),p(t-1)]		Target neuron T
		p(t-2)	*p(t-1)*	*p'(t)*			*p(t-2)*	*p(t-1)*	*p'(t)*
Training	1	0.292482	0.389996	0.527789	Testing	191	0.777281	0.652004	0.338593
	2	0.389996	0.527789	0.587076		192	0.652004	0.338593	0.682901
	3	0.527789	0.587076	0.433147		193	0.338593	0.682901	0.470398
	...	...	...	...		...	...	...	...
	...	...	...	...		...	...	...	...
	188	0.331961	0.323989	0.414713		236	0.358507	0.529964	0.68318
	189	0.323989	0.414713	0.777281		237	0.529964	0.68318	0.434261
	190	0.414713	0.777281	0.652004		238	0.68318	0.434261	0.563281

2.4. BPNN Algorithm

Backpropagation Neural Network (BPNN) is one of the algorithms owned by ANN with supervised learning method [15]. In general, BPNN works to feed forward input signals to a hidden layer which is then forwarded to the output signal display. From the output layer do feedback to input layer accompanied by a change of weight between the layer connections. The BPNN has three layers consisting of an input layer, a hidden layer, and an output layer. In general, the development steps of BPNN for rainfall prediction are detailed as follows:

1) Normalization and segregation of data as shown by points B and C.
2) Designing BPNN (Fig. 2) to determine the number of input data layer, hidden layer, and output layer and define the training parameters used.
3) Testing. This stage is intended to confirm the ability of BPNN during the training process and determine the accuracy of the prediction.

The architecture of BPNN is shown in Fig. 2. It consists of 2 neurons in the input layer, 1 neuron on the output layer, and 6 hidden layers consisting of 100 neurons in each layer. The activation functions used from the input layers to the output layer are respectively *tansig*, *tansig*, *logsig*, *logsig*, *logsig*, *tansig*, *purelin*. Meanwhile, the learning algorithm uses the *traingdx* algorithm. In addition, the training parameters consists of *performFcn* = *mse*, error goal (*eg*) = 0.01, epoch = 2500, momentum constant (*mc*)

= 0.95, learning rate (*lr*) = 0.05 (Model BPNN1), 0.1 (Model BPNN2), and 0.3 (Model BPNN3), respectively.

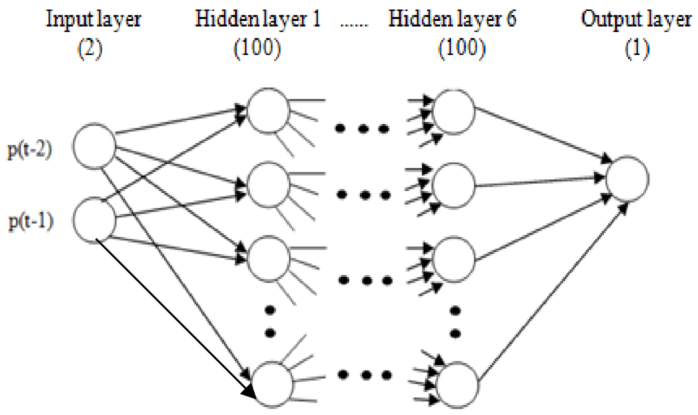


Fig. 2. The architecture of BPNN [2-100-100-100-100-100-100-1]

The training algorithms of the BPNN are as follows:

Step 0: Initialize all weights with small random numbers.

Step 1: If the termination condition has not been met, then it goes to steps 2-8.

Step 2: For each training-data pair, perform steps 3-8.

<u>**Phase 1**</u>: Feedforward propagation

Step 3: Each input unit receives a signal and passes it to a hidden unit above it.

Step 4: Calculate all outputs in the hidden unit z_j (j = 1, 2, ..., p).

$$z\_net_j = v_{j0} + \sum_{i=1}^{n} x_i v_{ji} \; . \tag{2}$$

$$z_j = f(z\_net_j) = \frac{1}{1+e^{-z\_net_j}} \; . \tag{3}$$

Step 5: Calculate all network outputs in unit y_k (k = 1, 2, ..., m).

$$y\_net_k = w_{k0} + \sum_{j=1}^{p} z_j w_{kj} \; . \tag{4}$$

$$y_k = f(y\_net_k) = \frac{1}{1+e^{-y\_net_k}} \tag{5}$$

<u>**Phase 2**</u>: Backpropagation

Step 6: Calculate δ factor of the output unit based on the error in each output unit y_k (k = 1, 2, ..., m)

$$\delta_k = (t_k - y_k) f'(y\_net_k) = (t_y - y_k) y_k (1 - y_k). \tag{6}$$

where δ_k is the unit of error to be used in the changing weights of the subsequent layer (step 7). We than calculate the weight change w_{kj} (to be used later to change the weight w_{kj}) with the learning rate α.

$$\Delta w_{kj} = \alpha \delta_k z_j ; k = 1,2,...,m; j = 0,1,..., p. \quad (7)$$

Step 7: Calculate δ factor of the hidden unit based on the error in each hidden unit z_j (j = 1, 2, ..., p)

$$\delta\_net_j = \sum_{k=1}^{m} (\delta_k w_{kj}). \quad (8)$$

The δ factor of the hidden units is defined as

$$\delta_j = \delta\_net_j f'(z\_net_j) = \delta\_net_j z_j (1 - z_j). \quad (9)$$

Calculate the change of weight v_{ji} (to be used later to change the weight v_{ji}) by the following equation

$$\Delta v_{ji} = \alpha \delta_j x_i ; j = 1,2,..., p; i = 0,1,..., n. \quad (10)$$

<u>**Phase 3**</u>: Change the weight

Step 8: Calculate all the weight changes of the line-weight change leading to the output unit

$$w_{kj}(new) = w_{kj}(old) + \Delta w_{kj} ; k = 1,2,...,m; j = 0,1,..., p. \quad (11)$$

The line-weight change leading to hidden unit is defined as

$$v_{ji}(new) = v_{ji}(old) + \Delta v_{ji} ; j = 1,2,..., p; i = 1,2,..., n. \quad (12)$$

2.5. RBFNN Algorithm

The Radial Basis Function Neural Network (RBFNN) is one of the algorithms owned by the ANN. Its algorithm works based on the theory of overseeing and unattended functioning or supervisory learning that work simultaneously (hybrid). The RBFNN algorithm is similar to the Feed Forward Neural Network (FFNN), in which its architecture has an input layer, a hidden layer, and an output layer [16]. The hidden layer of the RBFNN has a uniqueness, namely the number of layers only 1 with a Gaussian activation function (13) and the activation function of the output layer is linear [17], [18]. It is defined as,

$$R(X^q - C_i) = \exp\left[-\left(\left\| w1_i - X^q \right\| xb1_i \right)^2 \right], \quad (13)$$

where $||w1_i - X^q||$ is the Euclidean distance, c is the center of Gaussian function, and X^q is the input data.

The architecture of RBFNN is shown in Fig. 3. It consists of 2 neurons in the input layer, 200 neurons in the hidden layer, and 1 neuron in the output layer. The output value of the RBFNN is defined as

$$y = \sum_{j=1}^{m} w_{jm} \cdot \varphi, \quad (14)$$

where y is the output value, ϕ is the hidden value, and W denotes the weights.

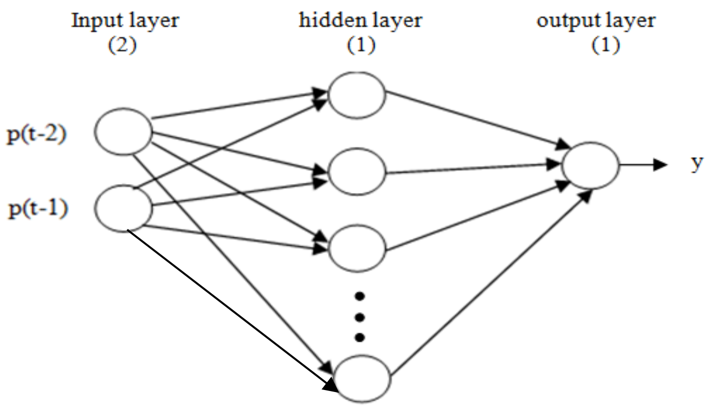


Fig. 3. The architecture of RBFNN [2-1-1]

The RBFNN algorithm consist of several steps as follows [19]:

1. Initialization of the network.
2. The second step is to find the distance D_{ij} between i and j $(i, j = 1, 2, \ldots, Q,$ where Q is the input-output vector, and R is the input variable). The distance is defined as

$$D_{ij} = \sqrt{\sum_{k=1}^{R} (P_{ik} - P_{jk})^2}. \tag{15}$$

3. The third step is to find $a1$, which is defined as,

$$a1_{ij} = e^{-(b1 * D_{ij})^2} \times b1 = \frac{\sqrt{-\ln(0.5)}}{spread}. \tag{16}$$

4. In the fourth step, we calculate the weight and bias. Note that w_{ij} is the new weight, $w_{ij}(t)$ is the weight at t, and α is the learning rate. The weight is defined as,

$$w_{ij} = (t+1) = w_{ij}(t) + \alpha(t)\left[x_i - w_{ij}(t)\right]. \tag{17}$$

2.6. Evaluation of Predictive Accuracy

The model reliability is evaluated using statistical analysis [20]-[26] by calculating several statistical parameters, namely the correlation coefficient (R), the Mean Square Error (MSE), the Mean Bias Error (MBE), and the Mean Absolute Error (MAE). The above mentioned statistical parameters are defined by following equations:

$$R = \frac{n\sum_{i=1}^{n} P_i Q_i - \left(\sum_{i=1}^{n} P_i\right)\left(\sum_{i=1}^{n} Q_i\right)}{\sqrt{n\left(\sum_{i=1}^{n} P_i^2\right) - \left(\sum_{i=1}^{n} P_i\right)^2}\sqrt{n\left(\sum_{i=1}^{n} Q_i^2\right) - \left(\sum_{i=1}^{n} P_i\right)^2}}, \tag{18}$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (P_i - Q_i)^2, \tag{19}$$

$$MBE = \frac{1}{n}\sum_{i=1}^{n} (P_i - Q_i), \tag{20}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|P_i - Q_i\right|, \tag{21}$$

where P is output network, Q is the actual data, and n denotes the number of data patterns.

3. Results and Discussion

In this section, the result from both the training and the testing process of the two algorithms used in this study (e.g. the BPNN and RBFNN) will be discussed. In the BPNN architecture, various levels of learning rate (*lr*) have been applied, i.e. *lr* = 0.05, 0.1, and 0.3, but with similar momentum constant (*mc)* = 0.95. Meanwhile, in the RBFNN architecture, we have applied various error goals (*eg*), i.e. *eg* = 0.001, 0.002, and 0.003. The other parameters used in the RBFNN are the spread = 1, the maximum number of neurons K = 200, and the display number of neurons Ki = 1. Table 3 shows the statistical analysis of the output from both the BPNN and RBFNN. It is shown that the RBFNN algorithm with *eg* = 0.001 has the best accuracy among other models. This architecture has the smallest MSE = 0.00091681.

The BPNN prediction results are lower than the actual value. It can be proved by negative value MBE at lr = 0.05 for -0.0006559 (training) and -0.025391 (testing). In contrast to *lr* = 0.1 and *lr* = 0.3, the predicted BPNN results are higher than the actual value with MBE which is positive. For absolute error (MAE), the average BPNN prediction result is 0.0759 (training) and 0.15639 (testing).

While on RBFNN, the predicted result is slightly lower than the actual value. This can be evidenced by MBE values of negative value (training and testing) on *eg* = 0.001 and *eg* = 0.002. In contrast to eg = 0.003, RBFNN predicted results are slightly higher than actual values with MBEs with positive values, ie 2.27E-14 (training) and 7.49E-16 (testing). For absolute error (MAE), the average RBFNN prediction is 1.1042 (training) and 1,125 (testing).

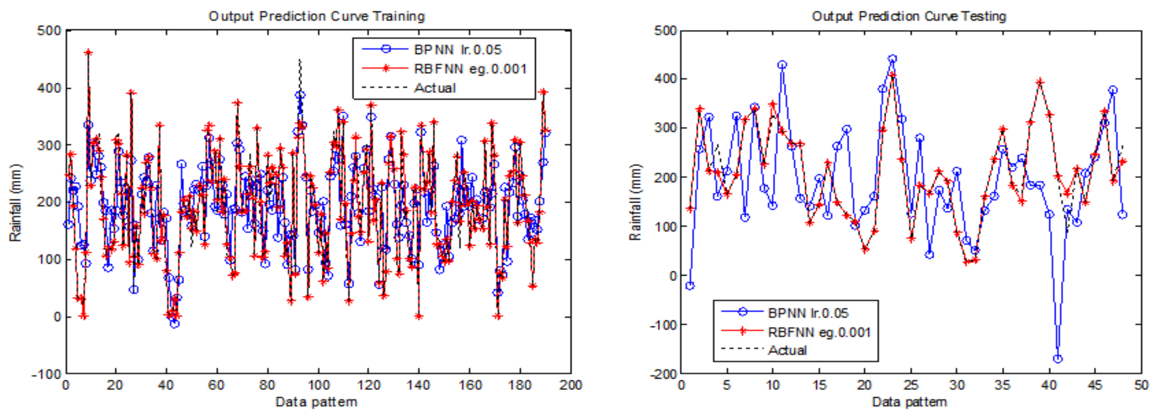
Table 3. The output error results of BPNN and RBFNN

BPNN *Lr*	Training			Testing		
	MSE	MBE	MAE	MSE	MBE	MAE
0.05	0.0099944	-0.0006559	0.07601	0.040461	-0.025391	0.16098
0.1	0.0099796	0.00036352	0.077022	0.027309	0.00051902	0.14321
0.3	0.0099977	0.00036818	0.074668	0.041001	0.020627	0.16498
RBFNN	**Training**			**Testing**		
eg	**MSE**	**MBE**	**MAE**	**MSE**	**MBE**	**MAE**
0.001	0.00071861	-3.90E-14	0.70914	0.00091681[a]	-1.00E-14	0.12958
0.002	0.001782	-6.87E-15	1.063	0.0018136	-7.33E-15	1.378
0.003	0.0027317	2.27E-14	1.5406	0.0029613	7.49E-16	1.8663

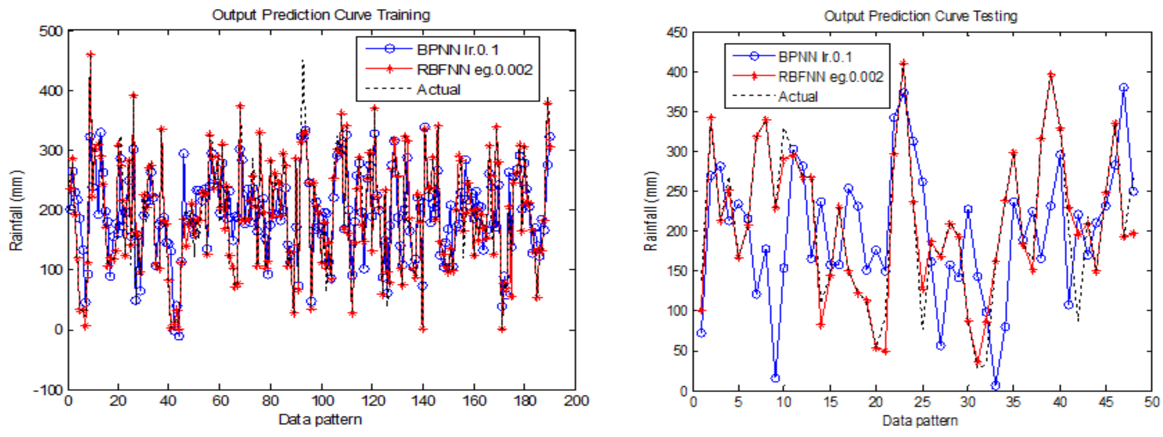
[a]. *The smallest MSE*

In this research, the duration of the iteration time was also investigated. The Iteration time training has met the best performance for each parameter algorithms. In the BPNN with *lr*=0.05, the iteration has been achieved in 240 seconds and reached the epoch 3714. Meanwhile, in the BPNN with *lr*=0.1, the iteration time was 249 seconds and reached the epoch 4070. However, in the BPNN with *lr*=0.3 has demonstrated longer iteration time with 273 seconds and reached the epoch 4651. On the other hand, in the RBFNN with error goal (*eg*) is different, it only takes an average of 15 seconds to achieve its best performance.

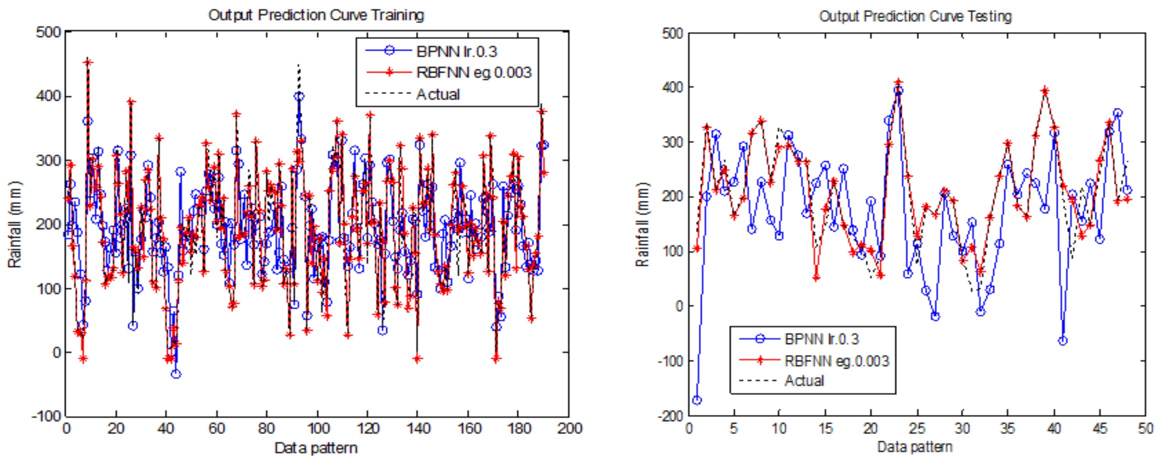
The training and testing results from those two algorithms are presented in Fig. 4. Each figure compares the output from each algorithm with different learning rate (e.g. the BPNN algorithm) and error goal (e.g. the RBFNN algorithm) for the outputs of both the training process and the testing process. In general, the predicted results of RBFNN (red line) are much dense in following actual data patterns or observations (black line) than the BPNN prediction results (blue line).



(a) The outputs from the training process (*left*) and the testing process (*right*) from the BPNN with *lr*=0.05 and the RBFNN with *eg*=0.001



(b) Same as in (a) except for the BPNN with *lr*=0.1 and the RBFNN with *eg*=0.002



(c) Same as in (a) except for the BPNN with *lr*=0.3 and the RBFNN with *eg*=0.003

Fig. 4. The output from the training (*left*) and the testing (*right*) processes for various types of the BPNN and RBFNN algorithms

In order to quantitatively evaluate the model performance, we then calculate the correlation between the observed monthly rainfall and the model outputs from each algorithm with various parameters during the training process (Fig. 5). It is clearly shown that the RBFNN algorithm with *eg*=0.001 has better correlation with the observation compared to the other 3 models of the BPNN algorithm. The correlation coefficient of the RBFNN is R=0.98 and the regression equation is $y=0.97x+6.3$. Meanwhile, the correlation coefficient of the BPNN algorithm with lr = 0.05, 0.1, and 0.3 are 0.80, 0.81, and 0.80, respectively. Therefore, the RBFNN algorithm with *eg*=0.001 is the best model for monthly rainfall prediction.

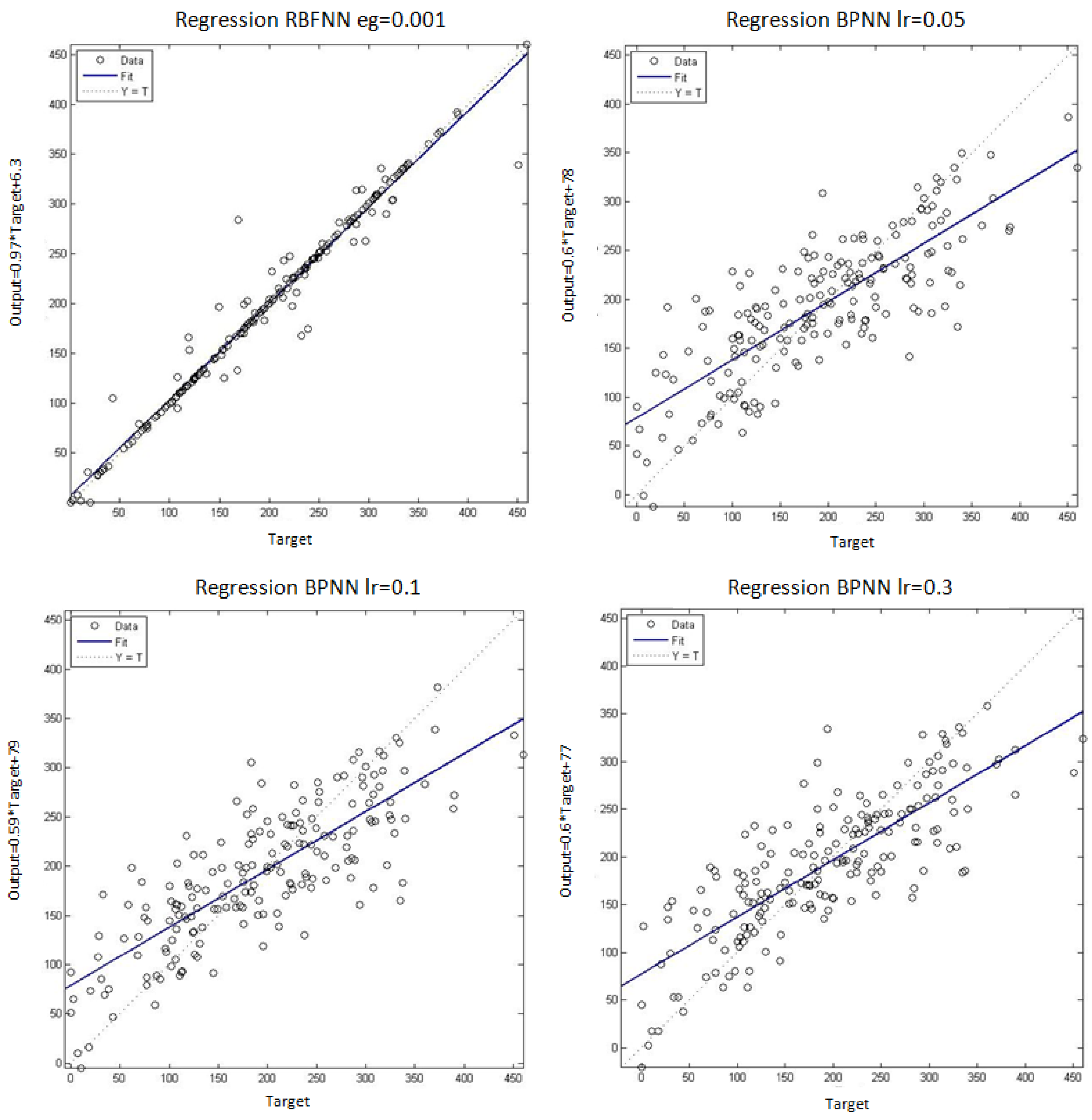


Fig. 5. The regression results between the observed monthly rainfall and the output from training process for the RBFNN with *eg*=0.001 (*upper left panel*), the BPNN with *lr*=0.05 (*upper right panel*), the BPNN with *lr*=0.1 (*lower left panel*), and the BPNN with *lr*=0.3 (*lower right panel*)

Meanwhile, the regression analysis between the observed monthly rainfall and the output from the testing process for both the RBPNN and RBFNN is shown in Fig. 6. The correlation coefficient of the RBFNN (*eg*=0.001) is R=0.86 and the regression equation is $y=0.74x+54$. Meanwhile, the correlation coefficient of the BPNN algorithm with lr = 0.05, 0.1, and 0.3 are 0.39, 0.55, and 0.37, respectively.

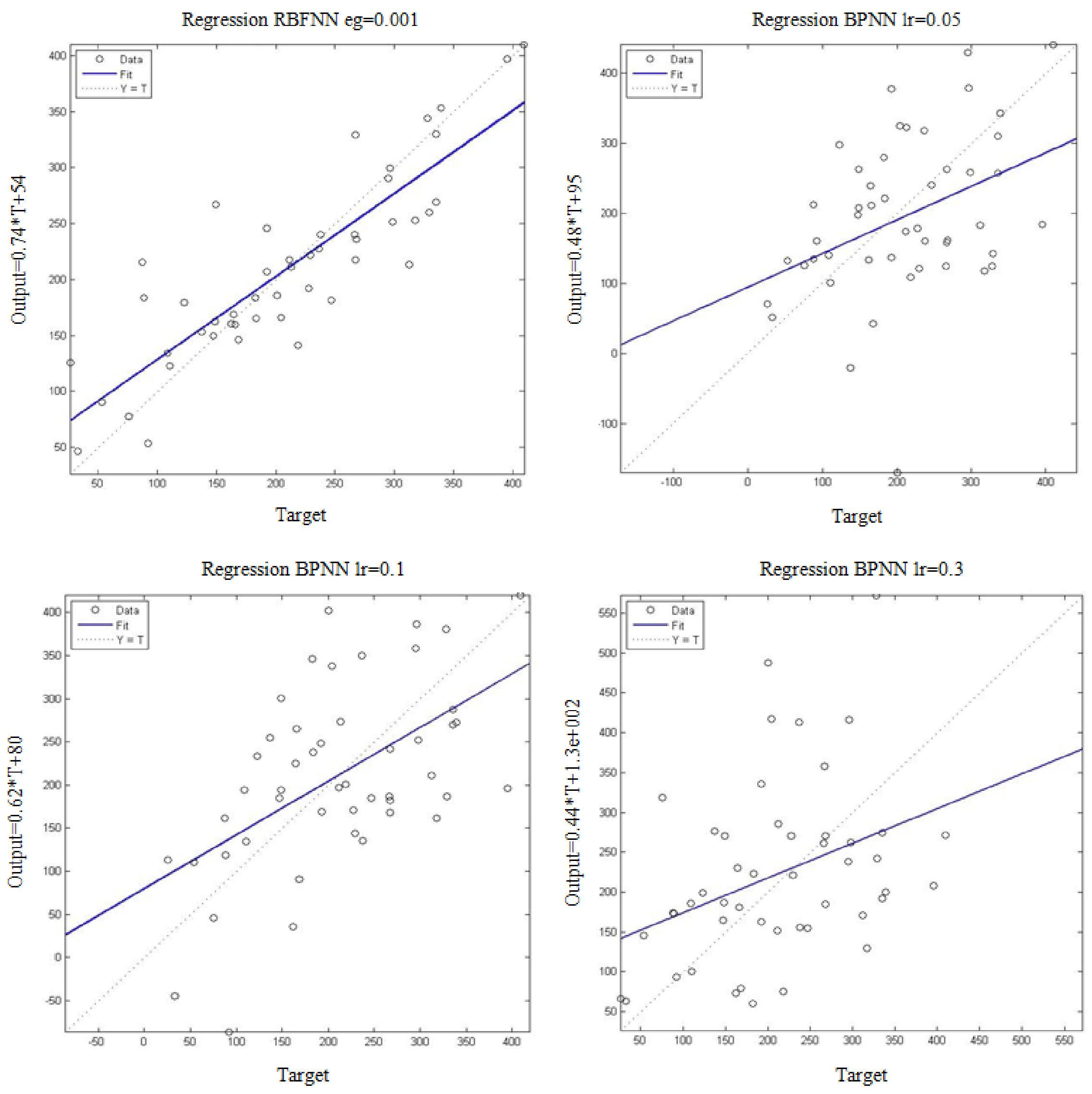


Fig. 6. The regression results between the observed monthly rainfall and the output from testing process for the RBFNN with *eg*=0.001 (*upper left panel*), the BPNN with *lr*=0.05 (*upper right panel*), the BPNN with *lr*=0.1 (*lower left panel*), and the BPNN with *lr*=0.3 (*lower right panel*)

4. Conclusion

In this study, two AAN method of algorithms namely the BPNN and the RBFNN have been used for monthly rainfall prediction. The statistical analysis has been performed to evaluate the prediction accuracy of each algorithm. It is found that the RBFNN algorithm with *eg*=0.001 shows a better results compared to the BPNN algorithm for both learning and testing processes. The correlation coefficients between the model output from the RBFNN algorithm and the observation for learning and testing process are 0.98 and 0.86, respectively. To minimize errors, the architecture of BPNN must have a long hidden layer with a large number of neurons. However, it takes a long time to find the best performance. Meanwhile, the RBFNN only has one hidden layer to find the best performance. This has an impact on the relatively short duration of the iteration time. Therefore, the accuracy of determining the architecture also affects the performance of the duration of the iteration time.

Acknowledgment

This research is supported by The Ministry of Research, Technology and Higher Education through "*Penelitian Berbasis Kompetensi 2018*" for the corresponding author.

References

[1] M. Zikra, Suntoyo, and Lukijanto, "Climate Change Impacts on Indonesian Coastal Areas," *Procedia Earth Planet. Sci.*, vol. 14, pp. 57–63, 2015, doi: https://doi.org/10.1016/j.proeps.2015.07.085.

[2] H. Meinke *et al.*, "Rainfall variability of decadal and longer time scales: Signal or noise?," *J. Clim.*, vol. 18, no. 1, pp. 89–90, 2005, doi: https://doi.org/10.1175/JCLI-3263.1.

[3] L. Tacconi, P. F. Moore, and D. Kaimowitz, "Fires in tropical forests - What is really the problem? Lessons from Indonesia," *Mitig. Adapt. Strateg. Glob. Chang.*, vol. 12, no. 1, pp. 55–66, 2007, doi: https://doi.org/10.1007/s11027-006-9040-y.

[4] M. Ardiansyah, R. Boer, and A. P. Situmorang, "Typology of land and forest fire in South Sumatra, Indonesia Based on Assessment of MODIS Data," in *IOP Conference Series: Earth and Environmental Science*, 2017, vol. 54, doi: https://doi.org/10.1088/1755-1315/54/1/012058.

[5] A. Saxena, N. Verma, and K. C. Tripathi, "Neuro-genetic hybrid approach for rainfall forecasting," *Int. J. Comput. Sci. Inf. Technol.*, vol. 5, no. 2, pp. 1291–1295, 2014, available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.639.5409&rep=rep1&type=pdf.

[6] Haviluddin and I. Tahyudin, "Time series prediction using radial basis function neural network," *Int. J. Electr. Comput. Eng.*, vol. 5, no. 4, pp. 765–771, 2015, available at: https://www.iaescore.com/journals/index.php/IJECE/article/view/5671.

[7] K. W. Wong, P. M. Wong, T. D. Gedeon, and C. C. Fung, "Rainfall prediction model using soft computing technique," *Soft Comput.*, vol. 7, no. 6, pp. 434–438, 2003, doi: https://doi.org/10.1007/s00500-002-0232-4.

[8] M. N. French, W. F. Krajewski, and R. R. Cuykendall, "Rainfall forecasting in space and time using a neural network," *J. Hydrol.*, vol. 137, no. 1–4, pp. 1–31, 1992, doi: https://doi.org/10.1016/0022-1694(92)90046-X.

[9] M. Richard and K. G. Rao, "Artificial neural networks in temporal and spatial variability studies and prediction of rainfall," *ISH J. Hydraul. Eng.*, vol. 20, no. 1, pp. 1–6, 2014, doi: https://doi.org/10.1080/09715010.2013.806400.

[10] U. Yolcu, E. Egrioglu, and C. H. Aladag, "A new linear & nonlinear artificial neural network model for time series forecasting," *Decis. Support Syst.*, vol. 54, no. 3, pp. 1340–1347, 2013, doi: https://doi.org/10.1016/j.dss.2012.12.006.

[11] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial Neural Network: A Tutorial," *Communications*, vol. 29, pp. 31–44, 1996, doi: https://doi.org/10.1109/2.485891.

[12] K. Abhishek, M. P. Singh, S. Ghosh, and A. Anand, "Weather Forecasting Model using Artificial Neural Network," *Procedia Technol.*, vol. 4, pp. 311–318, 2012, doi: https://doi.org/10.1016/j.protcy.2012.05.047.

[13] Mislan, Haviluddin, S. Hardwinarto, Sumaryono, and M. Aipassa, "Rainfall Monthly Prediction Based on Artificial Neural Network: A Case Study in Tenggarong Station, East Kalimantan - Indonesia," *Procedia Comput. Sci.*, vol. 59, no. Iccsci, pp. 142–151, 2015, doi: https://doi.org/10.1016/j.procs.2015.07.528.

[14] I. Wahyuni, N. R. Adam, W. F. Mahmudy, and A. Iriany, "Modeling Backpropagation Neural Network for Rainfall Prediction in Tengger East Java," *2nd Int. Conf. Sustain. Inf. Eng. Technol. (SIET 2017)*, no. 11, pp. 170–175, 2017, doi: https://doi.org/10.1109/SIET.2017.8304130.

[15] S. C. Joshi and A. N. Cheeran, "MATLAB based back-propagation neural network for automatic speech recognition," *Int. J. Adv. Res. Electr. Electron. Instrum. Eng.*, vol. 3, no. 7, pp. 10498–10504, 2014, doi: https://doi.org/10.15662/ijareeie.2014.0307016.

[16] Y. U. Zhijun, "RBF neural networks optimization algorithm and application on tax forecasting," *Telkomnika*, vol. 11, no. 7, pp. 3491–3497, 2013, doi: https://doi.org/10.11591/telkomnika.v11i7.2199.

[17] H. Q. Zhang and J. B. Li, "Prediction of tourist quantity based on RBF neural network," *J. Comput.*, vol. 7, no. 4, pp. 965–970, 2012, doi: https://doi.org/10.4304/jcp.7.4.965-970.

[18] M. Awad, H. Pomares, I. Rojas, O. Salameh, and M. Hamdon, "Prediction of tme series using RBF neural networks: a new approach of clustering.," *Int. Arab J. Inf. …*, vol. 6, no. 2, pp. 138–144, 2009, available at: http://www.ccis2k.org/iajit/PDF/vol.6,no.2/5PTSURNNNAC138.pdf.

[19] Haviluddin and A. Jawahir, "Comparing of ARIMA and RBFNN for short-term forecasting," *Int. J. Adv. Intell. Informatics*, vol. 1, no. 1, pp. 15–22, 2015, doi: https://doi.org/10.1292/ijain.v1i1.10.g8.

[20] A. Agustin, W. Mardiansyah, D. Setiabudidaya, and I. Iskandar, "WindSat and RAMA Buoy: a comparison of ocean-atmosphere data," in *MATEC Web of Conferences*, 2017, vol. 101, doi: https://doi.org/10.1051/matecconf/201710104005.

[21] A. Comrie, "Comparing neural networks and regression models for ozone forecasting," *J. Air Waste Manag. Assoc.*, vol. 47, no. 6, pp. 653–663, 1997, doi: https://doi.org/10.1080/10473289.1997.10463925.

[22] D. G. Fox, "Judging air quality model performance," 1981, vol. 62, no. 5, pp. 599–609, doi: https://doi.org/10.1175/1520-0477(1981)062<0599:JAQMP>2.0.CO;2.

[23] M. Kolehmainen, H. Martikainen, and J. Ruuskanen, "Neural networks and periodic components used in air quality forecasting," *Atmos. Environ.*, vol. 35, no. 5, pp. 815–825, 2001, doi: https://doi.org/10.1016/S1352-2310(00)00385-X.

[24] E. Walker, L. H. Slørdal, C. Guerreiro, F. Gram, and K. E. Grønskei, "Air pollution exposure monitoring and estimation. Part II: model evaluation and population exposure," *J. Environ. Monit.*, vol. 1, no. 4, pp. 321–326, 1999, doi: https://doi.org/10.1039/A902776I.

[25] C. J. Willmott *et al.*, "Statistics for the evaluation and comparison of models," *J. Geophys. Res. Ocean.*, vol. 90, no. C5, pp. 8995–9005, 1985, doi: https://doi.org/10.1029/JC090iC05p08995.

[26] C. . Willmott, "Some comments on the evaluation of model performance," 1982, vol. 63, no. 11, pp. 1309–1313, doi: https://doi.org/10.1175/1520-0477(1982)063<1309:SCOTEO>2.0.CO;2.