Knowledge Engineering and Data Science (KEDS)
Vol 3, No 1, July 2020, pp. 28–39

pISSN 2597-4602
eISSN 2597-4637

Earthquake Magnitude and Grid-Based Location Prediction using Backpropagation Neural Network

Bagus Priambodo [a, 1, *], Wayan Firdaus Mahmudy [a, 2], Muh Arif Rahman [a, 3]

[a] *Faculty of Computer Science, Brawijaya University*
Jl. Veteran no. 8, Malang 65145, Indonesia
[1] *baguspria@student.ub.ac.id *;* [2] *wayanfm@ub.ac.id;* [3] *m_arif@ub.ac.id*
** corresponding author*

ARTICLE INFO

Article history:
Received 30 June 2020
Revised 2 July 2020
Accepted 15 July 2020
Published online 17 August 2020

Keywords:
Neural network
Resilient backpropagation
Prediction
Magnitude
Earthquake

ABSTRACT

Earthquakes, a type of inevitable natural disaster, is responsible for the highest average death toll per year compared to other types of a natural disaster. Even though it is inevitable, but it can be anticipated to minimize damage and casualties, such as predicting the earthquake's magnitude using a neural network. In this study, a backpropagation algorithm is used to train the multilayer neural network to weekly predict the average magnitude of earthquakes in grid-based locations in Indonesia. Based on the findings in this research, the neural network is able to predict the magnitude of earthquakes in grid-based locations across Indonesia with a minimum error rate of 0.094 in 34.475 seconds. This best result is achieved when the neural network is trained for 210 epochs, with 16 neurons used in the input and output layer, one hidden layer consisted of 5 neurons and a learning rate of 0.1. This result showed backpropagation has pretty good generalization capability in order to map the relations between variables when mathematical function is not explicitly available.

I. Introduction

One of the inevitable disasters is a natural disaster. It may come without prior notice and has been responsible for the massive scale of deaths [1]. Centre for Research on Epidemiology of Disasters (CRED) reported an average of 77.144 deaths per year caused by natural disasters since 2000 to 2017 [2]. The natural disaster caused by seismic activities (earthquakes, tsunamis, and volcanic activities) disrupted 3.4 million lives in 2018 [2]. Earthquakes have caused the most deaths every year compared to other types of natural disasters, such as drought, flood, landslide, wildfire, and many more—with a toll of 46.173 lives [2].

Even though it is inevitable, but it can be anticipated to minimize damage and casualties. Past research has been conducted to predict the level of impact caused by earthquakes in real-time [2]. One of those past research is about an early earthquake warning system (EEWS), which will give an alert when it detects an earthquake [3]. Numerous architectures and algorithms have been developed in those studies. Various research is the utilization of neural network trained with backpropagation algorithm and optimized using Levenberg (LOM) to predict hypocenter location, moment magnitude, and the expansion of the earthquake [4], modification of LOM to minimize error on EEWS [3], and utilization of neural tree to predict P and S waves [5].

Until now, to the knowledge of the author, the study of using a backpropagation (BP) algorithm to predict earthquake magnitude and grid-based location in Indonesia has not been conducted yet. This algorithm is chosen because it has been proven to perform well in broad types of problems, such as regression, pattern recognition, and prediction [6][7][8][9]. In this paper, the study aims to measure the performance of neural network trained using backpropagation algorithm in predicting earthquakes magnitude and grid-based location based on earthquakes magnitude and location data in Indonesia recorded from 2000 to 2019.

https://doi.org/10.17977/um018v3i12020p28-39

II. Methods

Numerous studies on the application of neural networks to predict incoming natural disasters have been conducted before. One of them is to predict seismic activities (magnitude and other seismic activities on a large scale—more than 5 Ritcher scale) [6]. The findings of that research show acceptable performance (in terms of accuracy) of a 3-layer perceptron neural network model trained using backpropagation, which achieved an accuracy of 80.55%. The other research done is tsunami forecasting [10] using multi-layered perceptron neural networks and backpropagation algorithm. The study shows that the backpropagation algorithm is a lot faster than many other conventional models, and produced high accuracy in terms of predicting height and travel time of tsunami based on earthquake location and size. Often neural networks are hybridized with other techniques or algorithms. One example is the study done in 2006 [5], which used a neural tree to pick up P and S waves faster, and more accurately: with precision score achieved is 0.96.

Numerous studies showed the performance of neural networks in predicting earthquakes. Neural networks have been applied to an early earthquake warning system (EEWS), which is trained using backpropagation with modified Levenberg-Marquardt algorithm to minimize the error rate in the EEWS [3]. The error rate that was tried to be minimized was the error on seismic data amplitude prediction based on the Chi-chi earthquake in Taiwan in 1999. Other studies showed increased reliability dan responsiveness of EEWS when a neural network is applied [11]. A theoretical study by [4] showed a prediction of hypocenter location, moment magnitude, and rupture size of an earthquake is generated almost instantaneously as soon as the P wave is picked up.

A. Neural Network and Backpropagation

A neural network is a network that consists of connected neurons (to process inputs) whose job is to produce an activation value [12]. This network mimics the structure and the ways of how the human brain works, hoping it can be an artificial intelligence that is almost as intelligent as human—by receiving, adapting, and transferring known and new knowledge and skills, as a lifelong learning action [13]. A neural network consists of an input layer, zero or more hidden layer, and an output layer. Information on neurons will be propagated through each layer, starting from the input layer, all the way to the output layer [14][15]. The propagation is done by calculating the weighted sum on each neuron, which then used as the input value for the activation function used. The result of the activation function is then propagated to neurons in the next layer as inputs [11]. A common example of activation function is sigmoid function (1) and hyperbolic tangent (tanh) function (2) [16].

$$sig(x) = \frac{1}{1+e^{-x}} \tag{1}$$

$$tanh(x) = \frac{e^{2x}-1}{e^{2x}+1} \tag{2}$$

The backpropagation helped a neural network to learn the relationship between variables, without explicitly defining the mathematical function that defines that relationship [17]. A neural network trained using backpropagation is based on gradient descent [14]. In the backpropagation algorithm, each neuron in the hidden and output layer will process its own inputs using the sum product of each neuron's input value and weight of each neuron, respectively, which then processed through an activation function (most popular used is sigmoid activation function) [17]. Then the error will be calculated backward from the output layer to the input layer for the weight update process—this is called: backpropagation. This weight update process usually used gradient descent to direct the weight changes towards the minimum error [18]. During the weight update process, a parameter called the learning rate (η), will determine the "step width" of the updated value, which will finally update the weight in order to upgrade the neural network's performance in terms of its generalization capability. In the backpropagation process, the derivative of activation function and error function is needed to calculate the weight change. In (3) and (4) each is the derivative of sigmoid and tanh function, respectively.

$$sig'(x) = sig(x)(1 - sig(x)) \tag{3}$$

$$tanh'(x) = 1 - tanh^2(x) \tag{4}$$

The algorithm starts with the initialization phase. In this phase, all of the weights between neurons will be randomly initialized (between 0 and 1). The learning rate will also be initialized,

which is usually set to be 0.1. After the initialization phase, the first data will be used as input to the neural network, feedforward is then performed, which is calculating the weighted sum as input to activation function on the next layer. This process is repeated until the output layer has produced the predicted value. After the output layer has produced an output, the neural network will then begin its backpropagation phase.

During the backpropagation phase, the partial derivative of the error function with respect to each weight will be calculated, as the weights are directly contributed to the error. From the output layer point of view, the weight from the hidden layer to the output layer contributed to input for activation function in the output layer, resulting in a contribution to the error. From the hidden layer point of view, the weight from the input layer to the hidden layer contributed to input for activation function in the hidden layer, finally contributes to the error. Therefore, (5) will be used to calculate the gradient/partial derivative of the error with respect to weights between the hidden and output layer, and (6) will be used for the calculation of partial derivative with respect to weights between input and hidden layer.

$$\frac{\partial E}{\partial w_{HO}} = \frac{\partial i_O}{\partial w_{HO}} \frac{\partial o_O}{\partial i_O} \frac{\partial E}{\partial o_O} \tag{5}$$
$$= O_H \cdot sig'(i_O) \cdot \text{MAE}'$$

$$\frac{\partial E}{\partial w_{IH}} = \frac{\partial i_H}{\partial w_{IH}} \frac{\partial o_H}{\partial i_H} \frac{\partial E}{\partial o_H} \tag{6}$$
$$= in \cdot sig'(i_H) \cdot \sum_{x=1}^{n} \frac{\partial E_{O_x}}{\partial o_H}$$
$$= in \cdot sig'(i_H) \cdot \sum_{x=1}^{n} w_{HO_x} \cdot sig'(i_{O_x}) \cdot \text{MAE}'$$

In (5), $\frac{\partial E}{\partial w_{HO}}$ is the partial derivative of error concerning weights between the hidden and output layer. It consists of $\frac{\partial i_O}{\partial w_{HO}}$ which is the partial derivative of input to the output layer (the weighted sum value) for weight, and $\frac{\partial o_O}{\partial i_O}$ is the partial derivative of the activation function in the output layer to input on the output layer, while $\frac{\partial E}{\partial o_O}$ is the partial derivative of error function to the output. When each component is broken down, the final result of the formula is as in (5), the product of the output of the hidden layer (O_H), a derivative of activation function in the output layer (for example the derivative of sigmoid; $sig'(i_O)$) and derivative of the error function (10). Equation (6) is similar to (5), but it showed the partial derivative for weights between input and hidden layer. When the formula is broken down, the components are input value (input from training data; in), the partial derivative of activation function in the hidden layer ($sig'(i_H)$), and the partial derivative of error concerning output of the hidden layer will need further calculation steps. For partial derivative of the error to the output of the hidden layer, it is the sum of all (written as "n" which represents the number of output neurons) partial derivative of the error with respect to particular output neurons from neurons in the hidden layer. Thus, this component is actually the product of weights between the hidden and output layer with $\frac{\partial E_{O_x}}{\partial i_O}$ from (5). The partial derivative for biases is calculated likewise (as in (5) and (6)), but O_H in (5) and in in (6) is changed into 1, because it is the derivative with respect to bias.

$$MAE' = \begin{cases} +1\,;\ y_{pred} > y_{true} \\ -1\,;\ y_{pred} < y_{true} \end{cases} \tag{7}$$

The derivative of the error function used in this study (MAE) is shown in (7). This function can only be differentiated only when the difference in predicted, and the target value is not 0. In the context of (5), only one prediction and target value each used because in (5), it is only differentiated with respect to one output neuron only. Later for programming purposes, to eliminate the possibility of the unhandled case, the derivative will be 1 when the prediction is bigger or equal to the target value.

After partial derivative for each weight (biases included), the weight will then be updated. The weight update will follow the equation written in (8). The new weight for the next iteration ($w_{ij}(t +$

1)) will be the current weight $w_{ij}(t)$ reduced by the product of learning rate (η) and the partial derivative of the error $\frac{\partial E}{\partial w_{ij}}(t)$. This feedforward, backpropagation, and weight update process will be repeated until the maximum epoch has been reached, where one epoch is counted when all of the training data has been fed to the neural network.

$$w_{ij}(t+1) = w_{ij}(t) - \eta \frac{\partial \mathrm{E}}{\partial w_{ij}} \tag{8}$$

B. Feature Extraction of Earthquakes

Earthquake is a geological phenomenon that happened because of the shifting of earth's plates, caused by excessive pressure which the earth's crust cannot handle [19]. This excessive pressure results in energy release in the form of waves that propagate through the earth's crust, causing shockwave people can felt [20]. Those waves are picked up by seismographs. There are two types of waves, P wave (the fastest wave), which will be received first by the seismograph, then followed by a stronger wave, S wave, but slower than the P wave [21].

In seismograph, these two waves will produce data in 3 components: vertical, north-south, and east-west motion. [22]. Based on data of P wave, S wave, epicenter, magnitude, and peak ground acceleration (PGA), an early warning to civilians can be done, and emergency action can be conducted earlier; this can be done with a tool: early earthquake warning system (EEWS).

In this study, the magnitude and location of earthquakes that happened in Indonesia, starting from January 1 2000 until December 31 2019, will be used as an input feature. It was chosen based on a previous study done by [23]. In that study, [23] used four features: earthquake number, location (represented in numbers corresponding to grids), magnitude, and hypocenter depth. Date and time are not used in that study because preliminary statistical analysis on the data used showed date and time is not representative enough as a feature. In fact, it had too much unrelated/unneeded information. Detailed location data—coordinates—is simplified into grids representing an area that has been divided into 16 areas. Thus, the location feature consists of an integer number, ranging from 0 to 15. Based on this, the current study also divides the location data used in this study into 16 grids, represented using an integer from 0 to 15, as seen in Figure 1. The area is divided into 16 rectangles with every rectangle has the almost equal area. There will be 2 rows and 8 columns used, where starting from latitude of -10.909° to 5.907° will be divided into 2 sectors, and from longitude of 95.206° to 140.976° will be divided into 8 sectors.

In another study, the magnitude of past earthquakes is used to predict earthquake magnitude for the following day [6]. In that study, location is not used as a feature because it was assumed already known beforehand. Therefore, magnitude and location (represented in grid numbers) of an earthquake will be used as an input feature in this study.

Magnitude will be normalized using minmax normalization to match the characteristics of sigmoid and tanh function, resulting in the more appropriate value. The result can then be denormalized to get the actual predicted magnitude value by RB algorithm.

$$z = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{9}$$

Fig. 1. Grid-numbering as location feature

In (9), the minmax normalization equation is shown. Variable *x* is the magnitude value that will be normalized, *min(x)* is the minimum magnitude over all the data, and *max(x)* is the maximum magnitude over all of the data. Based on (9), the denormalization equation can be inferred as in (10).

$$x = z \times \{\max(x) - \min(x)\} + \min(x) \tag{10}$$

C. Evaluation Metrics

There are many methods that can be used to evaluate the performance of a neural network. In this study, Mean Absolute Error (MAE) will be used as evaluation metrics. This metric calculates the distance of predicted value and the target value. This metric is used based on a study done in [6] and [23]. The equation for MAE can be seen in (11).

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_{i_{pred}} - y_{i_{true}}\right| \tag{11}$$

In (13), *n* represents the feature count, $y_{i_{true}}$ represents target value of i_{th} feature, and $y_{i_{pred}}$ represents the predicted value of i_{th} feature.

D. Proposed Method

The data is queried from United States Geological Survey (USGS) website and obtained in a comma-separated value (CSV) file. The data collected is earthquake data that happened in Indonesia, specifically inside these boundaries: latitude of -10.909° to 5.907°, and longitude of 95.206° to 140.976°. Earthquake events from January 1 2000 up to January 1 2019 will be used as training data (36,453 records) and earthquake events from January 2 2019 up to December 31 2019 will be used as testing data (2,358 records). The data obtained (total of 38,811 records) is presented as seismic data with 22 columns/attributes, and 4 of them will be taken as features in this study. Those 4 attributes are detailed in Table 1.

After the data is obtained, there is two phases of preprocessing that will be done. First, 'id' attribute will be added as feature to index each data. Then, the data on 'time' column will be changed to 'date', because only the dates will be taken as the feature. The 'latitude' and 'longitude' feature will then be mapped into grid numbers (as explained before) and represented as 'grid' feature. The last feature is taken as it is, which is the 'mag' that represents the magnitude of the event. A snippet of first 5 data after this first phase of preprocessing can be seen in Table 2.

After the preliminary phase of preprocessing has been done, the final phase will be done to create a dataset that is ready to be used by the neural network. In this phase, each event will be grouped into weekly-period data, and for each week, the average magnitude will be calculated for each grid. This will be 'avg_mag' feature. Then, this will be normalized using minmax normalization. The final feature added will be the 'target' feature, which is the 'avg_mag' on the following week.

Table 1. Data attributes details

No	Name	Description	Data Type	Value
1	time	Date and time in milliseconds & UTC when the even occured.	Long integer	[2000-01-02T12:46:58.770Z, 2019-12-31T09:50:41.876Z]
2	latitude	Decimal degrees latitude.	Decimal	[-90,0, 90,0]
3	longitude	Decimal degress longitude.	Decimal	[-180,0, 180,0]
4	mag	Magnitude of the event.	Decimal	[-1,0, 10,0]

Table 2. Preliminary phase of preprocessing on the first 5 data

Id	Date	Grid	Mag
1	1/2/2000	6	4.9
2	1/2/2000	6	4.4
3	1/3/2000	6	4.7
4	1/4/2000	14	4.4
5	1/4/2000	5	3.9

Table 3. Final phase of preprocessing on the first week data

Grid	Avg_mag	Target
1	0.699	0.000
2	0.000	0.000
3	0.000	0.000
4	0.000	0.000
5	0.582	0.000
6	0.627	0.594
7	0.658	0.603
8	0.000	0.637
9	0.000	0.000
10	0.767	0.616
11	0.676	0.621
12	0.651	0.000
13	0.000	0.000
14	0.589	0.548
15	0.562	0.644
16	0.616	0.000

Snippet on first week data that has been preprocessed (earthquake events from January 2 2000 to January 8 2000) can be seen in Table 3.

The neural network built will consist of 16 neurons in the input and output layer and one hidden layer. The reason only one hidden layer is used is based on numerous study that has shown the good result of approximation on any continuous function [17]. Each neuron in the input and output layer represents the average magnitude in each grid. The input layer will receive data of the average magnitude of earthquake events in a week, and the output layer will predict the average magnitude for each grid in the following week. Each neuron in the hidden and output layer will have a bias with uniform-value: 1.

III. Results and Discussions

In order to achieve the best result, there will be four components of the neural network configuration that will be tested. The first one is to find the best maximum epochs allowed, then the learning rate, the number of neurons needed in the hidden layer, and lastly, which activation function will be used: sigmoid or tanh function.

In the maximum Epochs Testing testing, the maximum epochs allowed will be tested from 50, then increased by 10 until no significant error rate change occurred. For current testing, the neural network will use 5 neurons in the hidden layer, trained using the sigmoid function, a learning rate of 0.1, and all of the weights (except the biases) will be initialized with uniform value: 0.5. For each number of maximum epochs tested, it will be tested ten times, and the average will be taken. The average error rate and training duration will be measured to determine how many epochs that will produce the best result. Figure 2 showed the lowest average error rate achieved is 0.093 when the neural network is trained for 210 epochs, while the highest average error rate achieved is 0.094 when it is trained for 160 epochs. Figure 3 showed when a neural network is trained for 50 epochs, it only took 12.93 seconds, but it took 134.49 seconds to complete training for 500 epochs.

Another parameter that may contribute to the performance of the neural network's generalization capability is the learning rate. If it is too big, it may miss the optimal accuracy. If it is too small, the learning process would be too slow. In the learning rate testing, the learning rate will be tested from 0.1, then increased by 0.1 until no significant error rate change occurred. For current testing, the neural network will be trained using sigmoid function in 210 epochs (the best result achieved from the last testing) using 5 neurons in the hidden layer, and all of the weights (except the biases) will be initialized with uniform value: 0.5. For each learning rate tested, it will be tested ten times, and the average will be taken. Figure 4 showed the lowest average error rate achieved is 0.093 when the

neural network is trained using 0.1 as the learning rate, while the highest average accuracy achieved is 0.103 when it is trained using 1.0 as the learning rate. The average training duration is not tested in the current testing phase, as the learning rate does not affect directly to the training duration in terms of computation time.

The next test measures the number of neurons needed in the hidden layer will be tested from 5 neurons, then increased by 2 until no significant error rate change occurred. For current testing, the neural network will be trained using a sigmoid function in 210 epochs (the best result achieved from the last testing) with a learning rate of 0.1, and all of the weights (except the biases) will be initialized with uniform value: 0.5. For each number of neurons tested, it will be tested ten times, and the average will be taken. The average error rate and duration will be measured to determine how many neurons in the hidden layer that will produce the best result. Figure 5 showed the lowest average error rate achieved is 0.093 when the neural network is trained using 5 neurons in the hidden layer, while the highest average error rate achieved is 0.095 when it is trained using 23 neurons in the hidden layer. The average duration for training, as shown in Figure 6 is similar to the result in the last testing: as the number of neurons increased, the training duration also increased. The best training time achieved is 57.23 seconds when trained using 5 neurons only, and the worst result is 75.06 when trained using 23 neurons.

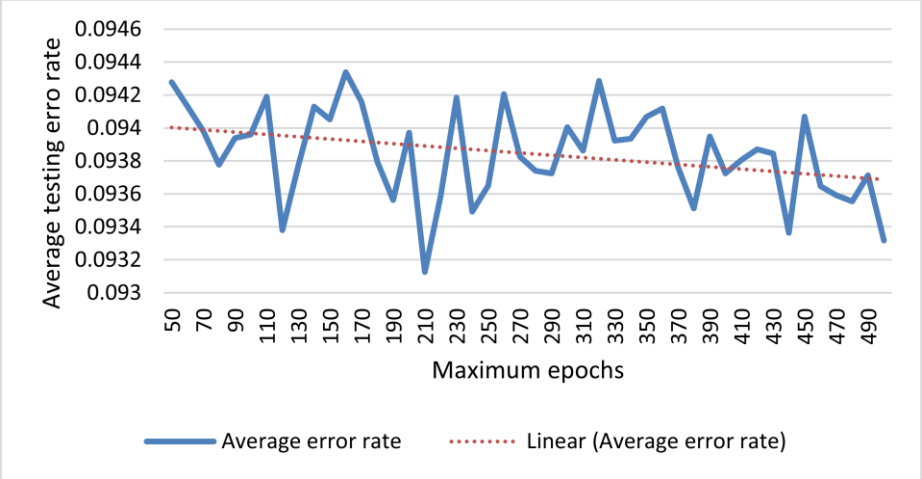

Fig. 2.The average testing error rate for each maximum epochs value

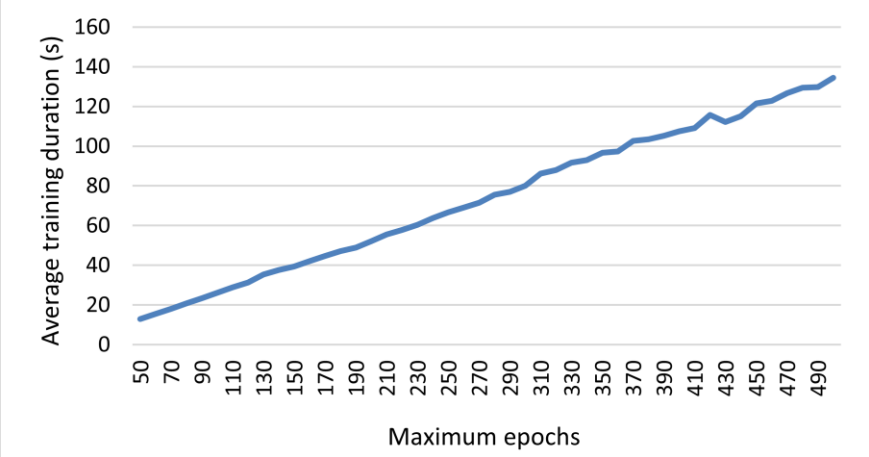

Fig. 3. Average training duration for each maximum epochs value

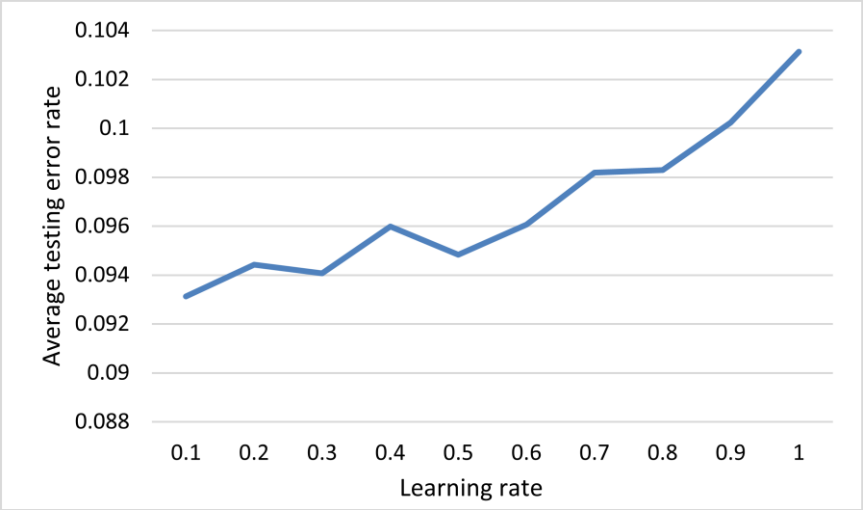

Fig. 4. The average testing error rate for each learning rate value

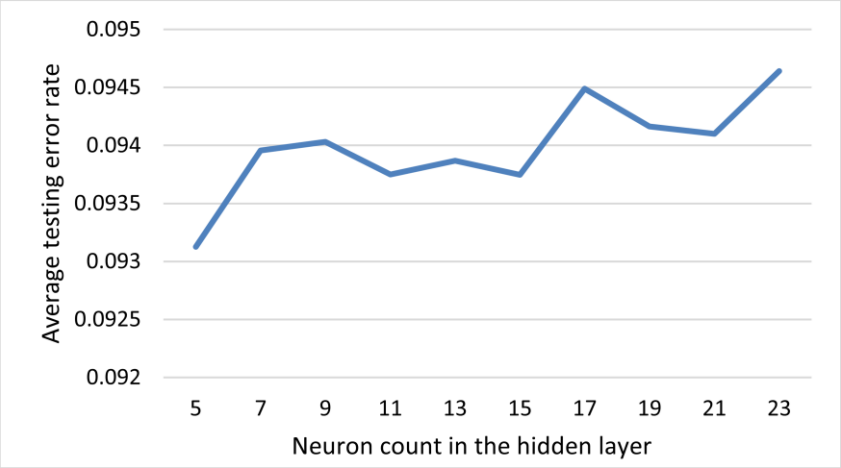

Fig. 5. Average testing accuracy for each neuron count in the hidden layer

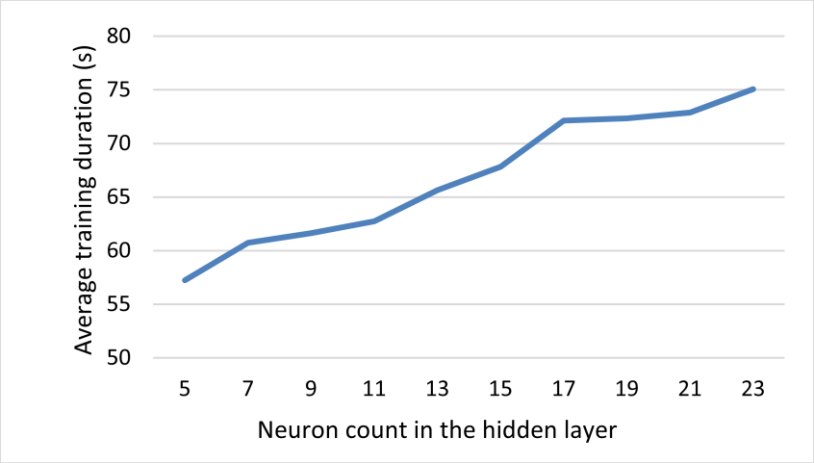

Fig. 6. Average training duration for each neuron count in the hidden layer

The Activation Function Testing used a sigmoid function or tanh function. For current testing, the neural network will be trained in 210 epochs, the learning rate of 0.1, using 5 neurons in the hidden layer (the best result achieved from the last testing), but all of the weights (except the biases) will be initialized with a random value. For each number of maximum epochs tested, it will be tested ten times, and the average will be taken. The sigmoid function produced an average error rate of 0.094, which is better than the tanh function, which produced an average error rate of 0.881. The sigmoid function is also better than tanh function in terms of training duration, which only took 60.81 seconds compared to 60.91 seconds when trained using tanh function. Thus, the sigmoid function will be used as the activation function.

Based on the testing results, when the neural network is trained with sigmoid as the activation function in 210 epochs using 5 neurons in the hidden layer and 0.1 as learning rate, it achieves its best performance with an error rate of 0.094 in 60.81 seconds. As shown in Figure 2, there is a tendency in the beginning that as the number of maximum epochs increased, the average accuracy also increased. This is also shown in Figure 7. Figure 7 shows the error rate of the neural network on the training data for each epoch. It is shown that as it learns the data over and over at each epoch, it became better in terms of error rate. This proves that the backpropagation algorithm is able to adapt to the nature of data more accurately as it received more training [24]. This is where the weight adaptation process in backpropagation takes part. Through the weight adaptation process, it finally produces weights that are well-suited to the training data and reached better generalization capability when tested against new data in the testing phase.

As the learning rate increases, it is shown that the performance of the neural network is getting worse. This may happen because of the step width is too large that the local minimum may be missed. It is also shown that as the number of neurons in the hidden layer increases, the performance is not getting better. This may be the result of overfitting, a condition where the neural network generalization capability became weak, and only achieve a good result when tested to the training data only. Therefore, when the neural network is tested using the testing data, the error rate is higher.

The duration needed for the neural network to predict earthquake event magnitude (for events in 2019) is the total duration from training phase until testing phase; the prediction duration. The average prediction duration is 0.005 seconds. Thus, overall duration needed is 60.81 seconds (average training duration) plus 0.005 seconds (prediction duration): 60.815 seconds. This is the total duration needed to predict 51 rows of data earthquake event magnitude based on training result on 988 rows of data, with 16 features each.

As shown in Table 4, it is a snippet of prediction and target comparison for the first two weeks. In prediction result, there are 12.99% (106 events) of the prediction that missed the target by 1 up to 5.2, and the others (about 87.01%—710 events) missed the target less than 1, with minimum error recorded is 2.56×10^{-4}.

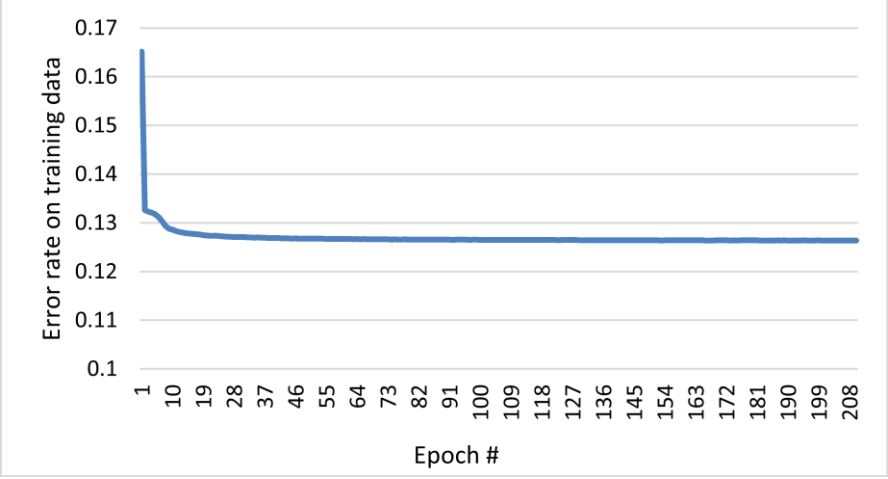

Fig. 7. Error rate during training on each epoch

Table 4. Comparison of prediction and target produced by resilient backpropagation algorithm

Date	Grid	Prediction	Target
1/9/2019	1	4.524	4.767
1/9/2019	2	0.001	0.000
1/9/2019	3	0.001	0.000
1/9/2019	4	0.001	0.000
1/9/2019	5	4.398	4.200
1/9/2019	6	4.571	4.467
1/9/2019	7	0.004	4.400
1/9/2019	8	0.005	0.000
1/9/2019	9	0.002	0.000
1/9/2019	10	4.510	5.000
1/9/2019	11	4.285	4.300
1/9/2019	12	4.317	0.000
1/9/2019	13	4.482	4.540
1/9/2019	14	4.350	4.488
1/9/2019	15	4.308	4.467
1/9/2019	16	4.515	0.000
1/16/2019	1	4.519	4.300
1/16/2019	2	0.001	4.300
1/16/2019	3	0.001	0.000
1/16/2019	4	0.001	0.000
1/16/2019	5	4.386	4.433
1/16/2019	6	4.568	4.688
1/16/2019	7	0.003	0.000
1/16/2019	8	0.004	0.000
1/16/2019	9	0.001	0.000
1/16/2019	10	4.542	4.350
1/16/2019	11	4.302	4.300
1/16/2019	12	4.410	4.200
1/16/2019	13	4.499	4.975
1/16/2019	14	4.364	4.443
1/16/2019	15	4.316	4.450
1/16/2019	16	4.475	0.000

IV. Conclusion

There are two key findings in this study. First, to build input features based on magnitude and location of earthquake event, detailed information of location (such as latitudes and longitudes) needed to be mapped into grids, then the magnitude will be averaged weekly for each grid number. The average magnitude weekly and based on each grid will be the input features. Secondly, the lowest error rate achieved by backpropagation algorithm (when trained in 210 epochs using sigmoid activation function, 5 neurons in the hidden layer and 0.1 as learning rate) to predict the magnitude of earthquake event in the following week is 0.094, with 60.815 seconds needed for the neural network to learn from 988 rows of data and predict 51 rows of data. Based on those key findings, it is recommended to further studying the importance and impact of other input features, configuration of the neural network, and hybridizing with other algorithms, which can further maximize the performance in predicting earthquakes more accurately and in the small-time window to increase preparedness.

Declarations

Author contribution

All authors contributed equally as the main contributor of this paper. All authors read and approved the final paper.

Funding statement

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Conflict of interest

The authors declare no conflict of interest.

Additional information

No additional information is available for this paper.

References

[1] D. Guha-Sapir and F. Vos, "Earthquakes, an Epidemiological Perspective on Patterns and Trends," Advances in Natural and Technological Hazards Research, pp. 13–24, Dec. 2010.

[2] Centre for Research on Epidemiology of Disasters (CRED), "Press release: EMBARGO 11.00 CET, JANUARY 24 24," 2019.

[3] J. W. Lin, C. T. Chao, and J. S. Chiou, "Backpropagation neural network as earthquake early warning tool using a new modified elementary Levenberg-Marquardt Algorithm to minimise backpropagation errors," *Geosci. Instrumentation, Methods Data Syst.*, vol. 7, no. 3, pp. 235–243, 2018, doi: 10.5194/gi-7-235-2018.

[4] M. Böse, F. Wenzel, and M. Erdik, "PreSEIS: A neural network-based approach to earthquake early warning for finite faults," *Bull. Seismol. Soc. Am.*, vol. 98, no. 1, pp. 366–382, 2008, doi: 10.1785/0120070002.

[5] S. Gentili and A. Michelini, "Automatic picking of P and S phases using a neural tree," *J. Seismol.*, vol. 10, no. 1, pp. 39–63, 2006, doi: 10.1007/s10950-006-2296-6.

[6] M. Moustra, M. Avraamides, and C. Christodoulou, "Artificial neural networks for earthquake prediction using time series magnitude data or Seismic Electric Signals," *Expert Syst. Appl.*, vol. 38, no. 12, pp. 15032–15039, Nov. 2011, doi: 10.1016/j.eswa.2011.05.043.

[7] N. R. Sari, W. F. Mahmudy, and A. P. Wibawa, "Backpropagation on neural network method for inflation rate forecasting in Indonesia," *Int. J. Adv. Soft Comput. its Appl.*, vol. 8, no. 3, 2016.

[8] 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, doi: 10.11591/ijeecs.v4.i1.pp240-244.

[9] H. Aini and H. Haviluddin, "Crude Palm Oil Prediction Based on Backpropagation Neural Network Approach," *Knowl. Eng. Data Sci.*, vol. 2, no. 1, pp. 1–9, 2019.

[10] M. Romano *et al.*, "Artificial neural network for tsunami forecasting," *J. Asian Earth Sci.*, vol. 36, no. 1, pp. 29–37, 2009, doi: 10.1016/j.jseaes.2008.11.003.

[11] C. J. Lin, Z. Shen, and S. Huang, "Predicting Structural Response with On-Site Earthquake Early Warning System Using Neural Networks," *Weather*, no. 226, 2011.

[12] J. Schmidhuber, "Deep Learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015, doi: 10.1016/j.neunet.2014.09.003.

[13] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, pp. 54–71, May 2019, doi: 10.1016/j.neunet.2019.01.012.

[14] G. T. Hicham, E. A. Chaker, and E. Lotfi, "Comparative study of neural networks algorithms for cloud computing CPU scheduling," *Int. J. Electr. Comput. Eng.*, vol. 7, no. 6, pp. 3570–3577, 2017, doi: 10.11591/ijece.v7i6.pp3570-3577.

[15] C. Dewi, S. Sundari, and M. Mardji, "Texture Feature On Determining Quantity of Soil Organic Matter For Patchouli Plant Using Backpropagation Neural Network," *J. Inf. Technol. Comput. Sci.*, vol. 4, no. 1, pp. 1–14, 2019.

[16] K. Chandrasekaran and S. P. Simon, "Binary/real coded particle swarm optimization for unit commitment problem," in *International Conference on Power, Signals, Controls and Computation*, Jan. 2012, no. 3, pp. 1–6, doi: 10.1109/EPSCICON.2012.6175240.

[17] A. T. C. Goh, "Back-propagation neural networks for modeling complex systems," *Artif. Intell. Eng.*, vol. 9, no. 3, pp. 143–151, Jan. 1995, doi: 10.1016/0954-1810(94)00011-S.

[18] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: the RPROP algorithm," in *IEEE International Conference on Neural Networks*, 1993, pp. 586–591, doi: 10.1109/ICNN.1993.298623.

[19] K. Mogi, "Earthquake Prediction in Japan," *J. Phys. Earth*, vol. 43, no. 5, pp. 533–561, 1995.

[20] U.S. Geological Survey, "What is an earthquake and what causes them to happen?," *U.S. Department of the Interior*, 2019.

[21] Incorporated Research Institutions for Seismology (IRIS), "Seismic Wave Behavior — Effect on Buildings" .

[22] Incorporated Research Institutions for Seismology (IRIS), "3-Component Seismograph," 2017.

[23] A. S. N. Alarifi, N. S. N. Alarifi, and S. Al-Humidan, "Earthquakes magnitude predication using artificial neural

network in northern Red Sea area," *J. King Saud Univ. - Sci.*, vol. 24, no. 4, pp. 301–313, Oct. 2012, doi: 10.1016/j.jksus.2011.05.002.

[24] I. Wahyuni, N. R. Adam, W. F. Mahmudy, and A. Iriany, "Modeling backpropagation neural network for rainfall prediction in tengger east Java," in *Proceedings - 2017 International Conference on Sustainable Information Engineering and Technology, SIET 2017*, 2018, vol. 2018-Janua, doi: 10.1109/SIET.2017.8304130.