

## Implementation of recurrent neural network for the forecasting of USD buy rate against IDR

Lady Silk Moonlight<sup>1</sup>, Bambang Riyanto Trilaksono<sup>2</sup>, Bambang Bagus Harianto<sup>3</sup>, Fiqqih Faizah<sup>4</sup>

<sup>1</sup>D3 Aeronautical Communication, Politeknik Penerbangan Surabaya, Surabaya, Indonesia

<sup>2</sup>School of Electrical Engineering and Informatics, Bandung Institute of Technology, Bandung, Indonesia

<sup>3</sup>D3 Air Navigation Engineering, Politeknik Penerbangan Surabaya, Surabaya, Indonesia

<sup>4</sup>D3 Airport Electrical Engineering, Politeknik Penerbangan Surabaya, Surabaya, Indonesia

### Article Info

#### Article history:

Received Jan 27, 2022

Revised Dec 27, 2022

Accepted Jan 2, 2023

#### Keywords:

Backpropagation through time

Forecasting

Foreign exchange

Recurrent neural network

### ABSTRACT

This study implements a recurrent neural network (RNN) by comparing two RNN network structures, namely Elman and Jordan using the backpropagation through time (BPTT) programming algorithm in the training and forecasting process in foreign exchange forecasting cases. The activation functions used are the linear transfer function, the tan-sigmoid transfer function (Tansig), and the log-sigmoid transfer function (Logsig), which are applied to the hidden and output layers. The application of the activation function results in the log-sigmoid transfer function being the most appropriate activation function for the hidden layer, while the linear transfer function is the most appropriate activation function for the output layer. Based on the results of training and forecasting the USD against IDR currency, the Elman BPTT method is better than the Jordan BPTT method, with the best iteration being the 4000<sup>th</sup> iteration for both. The lowest root mean square error (RMSE) values for training and forecasting produced by Elman BPTT were 0.073477 and 122.15 the following day, while the Jordan backpropagation RNN method yielded 0.130317 and 222.96 also the following day.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



### Corresponding Author:

Lady Silk Moonlight

D3 Aeronautical Communication, Politeknik Penerbangan Surabaya

Jemur Andayani 1 No 73 Surabaya, Indonesia

Email: lady@potekbangsby.ac.id

## 1. INTRODUCTION

The role of foreign exchange rates can be seen in the increasing daily exchange rate transactions. In foreign exchange trading, exchange rates change almost daily. The volatility of currency fluctuations is a problem for multinational and international companies, as entrepreneurs can experience unpredictable losses. This fluctuating exchange rate movement makes it important to develop the science of forecasting in estimating foreign exchange rates with low risk.

In this study, an artificial neural network (ANN) is used because ANN performs grouping, classification, and pattern recognition. The advances that have been made by ANN in artificial intelligence (AI) are speech recognition, image recognition, robotics, forecasting, and others [1]. Neural network backpropagation algorithm in estimating the number of data on infected cases/deaths of the coronavirus disease (COVID-19) outbreak in Qatar, Spain, and Italy, with a high correlation coefficient ( $\geq 0.99$ ) [2]. In a previous study, compared with autoregressive integrated moving average (ARIMA) and ANN on COVID-19 prevalence forecasting, the key is that ANN can build a model that can predict three variables at the same time at an acceptable prediction level [3]. Recurrent neural networks (RNN) are used in predicting household

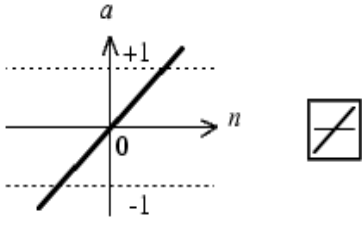
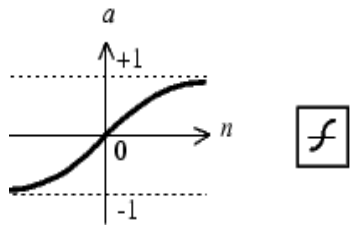
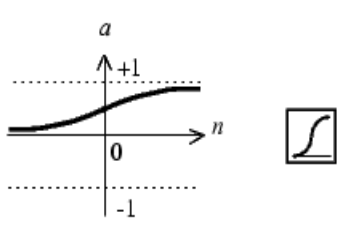
electricity use, with an average error percentage of 1.5% for replay predictions, and 4.6% for the maximum error [4]. In this study, the backpropagation through time (BPTT) algorithm is used, namely, the backpropagation algorithm which involves data (t-1) on the current data (t), adapted to the RNN architecture, namely Jordan and Elman.

**2. METHOD**

**2.1. Activation function**

The activation functions applied to the hidden and output layers are linear transfer functions, tan-sigmoid transfer functions (Tansig), and log-sigmoid transfer functions (Logsig). The activation function is described in Table 1. The activation function will be applied to the BPTT method on the hidden and output layers in each network architecture. The results of applying the activation function are compared and the most suitable is selected for this forecasting system. In the linear activation function, the output value is proportional to the input. In the Tansig activation function, the output value is in the range of -1 to 1. Meanwhile, in the Logsig activation function, the output value is in the range of 0 to 1 [5]–[7].

Table 1. Activation functions

Linear Transfer Function (Linear)	Tan-Sigmoid Transfer Function (Tansig)	Log-Sigmoid Transfer Function (Logsig)
 <p><math>a = \text{purelin}(n)</math></p> <p><math>a = n</math></p>	 <p><math>a = \text{tansig}(n)</math></p> <p><math>a = \frac{2}{(1+e^{-2n})} - 1</math></p>	 <p><math>a = \text{logsig}(n)</math></p> <p><math>a = \frac{1}{(1+e^{-n})}</math></p>
The output value is proportional to the input	Output values are in the range of -1 to 1.	The output value is in the range of 0 to 1.

**2.2. Recurrent neural network**

The simple recurrent network is a variation of the multi-layer perceptron (MLP), often referred to as the Elman network because it was discovered by Jeff Elman. The main difference in this structure is that there are several nodes adjacent to the input layer that are connected to the hidden layer like any other input node. These nodes contain the contents of one of the layers that existed when the previous pattern was trained. At each step, the input is propagated in the standard feedforward manner and then a learning rule (usually backpropagation) is used. The result of a fixed back connection in the context unit is a copy of the result from the previously hidden unit. Therefore, the network can maintain a sequence of states and allow it to perform some tasks such as sequence prediction that are beyond the capabilities of the standard MLP.

Recurrent neural network (RNN) has an architecture where there are several feedback connections from a neuron to the neuron itself or neurons in the previous layer. With the feedback connection, the network can maintain short-term memory. Memory can have an impact on the way of input brings the past back into the network. In RNN, the state of a layer is improved not only from the external input of the network but also from the results of the previous activation of the forward neuron. Feedback will be updated by adapting weights through learning [8]–[12].

**2.3. Recurrent neural network architecture**

In this study, the RNN architectures that were applied are the Jordan RNN and Elman RNN architectures, as shown in Figures 1(a) and (b). The architectures are then compared to see system performance in the case of foreign exchange forecasting. This architecture is applied to the training and forecasting processes. The difference between the two architectures is that in the Jordan architecture there is the feedback that comes from the output at the output layer to the input at the hidden layer. While the Elman architecture, the network has feedback that comes from the output on the hidden layer to the input on the hidden layer. From the application of the two RNN architectures above, the Elman architecture is best used in forecasting foreign exchange [13]–[15].

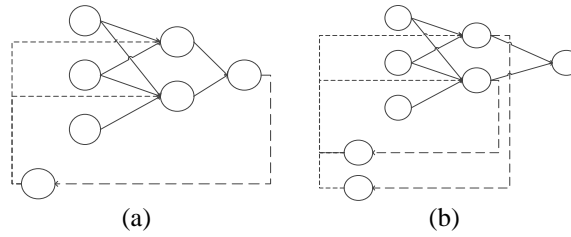


Figure 1. RNN architecture (a) Jordan and (b) Elman

#### 2.4. Backpropagation through time

Initially, the backpropagation used by Jeff Elman with  $y_i(t-1)$  was only an additional input. The error in a layer  $\delta_j(t)$  is used to correct the weight of the additional input. Because errors can be propagated back further, the backpropagation used in recurrent networks is called backpropagation through time and is a simple extension of backpropagation in general. The following are the steps of the backpropagation through time algorithm in a RNN [16]–[22]:

- 1) Creating a matrix for input data so that it can become training data.
  - 2) Normalizing training data as a preprocessing process for use in the learning process. Preprocessing will change the data in intervals of 0 to 1.
  - 3) Initialize the initial weight randomly in the interval 0 to 1. The initial weight includes the weight of the input unit to the hidden unit and the weight of the hidden unit to the output unit.
  - 4) Set the training data input value (normalized result) into the input layer.
  - 5) Perform the feedforward process for the first data to the last data. The feedforward process is the process of calculating the neuron output signal in the hidden layer and the output layer using the sigmoid activation function.
- Calculation of all incoming signals to hidden neurons from neuron input ( $u$ ), bias, and previous time output feedback ( $t-1$ ), shown in (1):

$$x\_in_j^{(n)} = v_{j0}^{(n)} + v_{jx}x_j^{(n-1)} + \sum_{i=1}^{NU} (v_{ji}u_i)^{(n)} \quad (1)$$

where  $x\_in_j^{(n)}$  is the incoming signal of hidden neurons ( $j^{th}$ ) at the time ( $n$ );  $v_{j0}^{(n)}$  is bias weights towards hidden neurons ( $j^{th}$ ) at the time ( $n$ );  $v_{jx}$  is the weight of the feedback to the hidden neurons; ( $j^{th}$ );  $x_j^{(n-1)}$  is hidden neuron output signal ( $j^{th}$ ) at the time ( $n-1$ );  $NU$  is the number of input neurons;  $v_{ji}^{(n)}$  is the weight of the input neuron ( $i^{th}$ ), towards the hidden neuron ( $j^{th}$ ) at the time ( $n-1$ );  $u_i^{(n)}$  is the input neuron signal  $i^{th}$  at the time ( $n$ ).

- The calculation of the hidden neuron of the output signal uses the sigmoid activation function. The sigmoid activation function is the shown in (2):

$$f(x) = \frac{1}{(1+e^{-x})} \quad (2)$$

- Hidden neuron output signal (activation) at the time  $n$  shown in (3):

$$x(n) = f(x\_in)(n) \quad (3)$$

- Calculation of all incoming signals to the output neurons from hidden neurons ( $x$ ), and bias, shown in (4):

$$y\_in_j^{(n)} = w_{j0}^{(n)} + w_{jx}y_j^{(n-1)} + \sum_{i=1}^{NX} (w_{ji}x_i)^{(n)} \quad (4)$$

where  $y\_in_j^{(n)}$  is signal enters the output neuron ( $j^{th}$ ) at the time  $n$ ;  $w_{j0}^{(n)}$  are weights that are biased towards the output neuron ( $j^{th}$ ) at the time ( $n$ );  $w_{jx}$  is the weight of the feedback that goes to the output neuron ( $j^{th}$ );  $y_j^{(n-1)}$  is output neuron signal is output ( $j^{th}$ ) at the time ( $n-1$ );  $NX$  is the number of hidden neurons;  $w_{ji}^{(n)}$  is weights of hidden neurons ( $i^{th}$ ) towards the output of neurons ( $j^{th}$ );  $x_i^{(n)}$  is hidden neuron output signal ( $i^{th}$ )

- The calculation of the output neuron output signal uses the sigmoid activation function, shown in (5). The neuron output signal is output (activation) at the time  $n$ .

$$y(n) = f(y_{in})(n) \quad (5)$$

- 6) Perform the backward process starting from the last data from the feedforward process to the first data. Calculate the error signal.
- The error calculation in the output unit is shown in (6):

$$e_j^{(n)} = (t_j^{(n)} - y_j^{(n)}) f'(y_{in_j}^{(n)}) \quad (6)$$

with

$$f'(y_{in_j}^{(n)}) = \frac{1}{1+e^{-y_{in_j}^{(n)}}} \left( 1 - \frac{1}{1+e^{-y_{in_j}^{(n)}}} \right) \quad (7)$$

or

$$f'(y_{in_j}^{(n)}) = f(y_{in_j}^{(n)}) (1 - f(y_{in_j}^{(n)})) \quad (8)$$

where  $e_j^{(n)}$  is error neuron output ( $j^{th}$ ) at the time  $n$ ;  $NY$  is the number of output neurons;  $t_j^{(n)}$  is the target ( $j^{th}$ ) at the time  $n$ ; and  $y_j^{(n)}$  is the output neuron signal is output ( $j^{th}$ ) at the time ( $n-1$ )

- Calculation of weight correction at the time  $n$  is shown in (9), (10), and (11):

$$\text{Output - Hidden: } \Delta w_{ji}^{(n)} = -\eta e_j^{(n)} x_i^{(n)} \quad \text{Output - Bias: } \Delta w_{j0}^{(n)} = -\eta e_j^{(n)} \quad (9)$$

$$\text{Output - Output (Feedback): } \Delta w_{jx}^{(n)} = -\eta e_j^{(n)} y_j^{(n-1)} \quad (10)$$

$$\text{Output - Hidden (Feedback): } \Delta w_{jx}^{(n)} = -\eta e_j^{(n)} x_i^{(n-1)} \quad (11)$$

where  $\eta$  is the *learning rate*;  $e_j^{(n)}$  is error neuron output to  $j^{th}$  at the time ( $n$ );  $x_i^{(n)}$  is The neuron output signal is hidden  $i^{th}$  at the time ( $n$ );  $y_j^{(n-1)}$  is the neuron output signal is output  $j^{th}$  at the time ( $n-1$ );  $x_i^{(n-1)}$  is the neuron output signal hidden  $i^{th}$  at the time ( $n-1$ ).

- Summation of input delta in the hidden unit shown in (12):

$$\delta_j^{(n)} = \sum_{i=1}^{NI} (w_{ij} e_i)^{(n)} + \sum_{k=1}^{NK} v_{jk} \varepsilon_k^{(n+1)} + \sum_{l=1}^{NL} v_{jl} e_l^{(n+1)} \quad (12)$$

where  $\delta_j^{(n)}$  is delta input neuron hidden ( $j^{th}$ );  $NI$  is the number of Feedforward neurons where the connection comes from neuron ( $j^{th}$ );  $w_{ij}^{(n)}$  is the weight of hidden neurons ( $j^{th}$ ) leads to the output neurons to ( $i^{th}$ ) at the time ( $n$ );  $e_i^{(n)}$  is error output neuron ( $i^{th}$ ) which has a Feedforward connection to hidden neuron ( $j^{th}$ ) at the time  $n$ ;  $NK$  is the number of hidden neurons that have feedback connections to hidden neurons to ( $j^{th}$ );  $v_{jk}$  is the weight of the feedback from hidden neurons ( $k^{th}$ ) to hidden neurons ( $j^{th}$ );  $\varepsilon_k^{(n+1)}$  is error hidden neurons ( $k^{th}$ ) which have a feedback connection to hidden neurons  $j^{th}$  at the time ( $n+1$ );  $NL$  is the number of output neurons that have feedback connections to hidden neurons ( $j^{th}$ );  $v_{jl}$  is the weights of the output neuron feedback 1 towards the hidden neurons ( $j^{th}$ ); and  $e_l^{(n+1)}$  is error output neuron 1 which has a feedback connection to hidden neuron  $j^{th}$  at the time ( $n+1$ ).

- Error calculation in hidden units at the time  $n$  shown in (13):

$$\varepsilon_j^{(n)} = f'(x_{in_j}^{(n)}) \delta_j^{(n)} \quad (13)$$

where  $\varepsilon_j^{(n)}$  is hidden neuron error ( $i^{th}$ ) at the time ( $n$ ); and  $\delta_j^{(n)}$  is delta input neuron hidden ( $j^{th}$ ) at the time ( $n$ ).

- Calculation of the weight correction at the time ( $n$ ) shown in (14), (15), and (16):

$$\text{Hidden - Input: } \Delta v_{ji}^{(n)} = -\eta \varepsilon_j^{(n)} u_i^{(n)} \quad (14)$$

$$\text{Hidden - Bias: } \Delta v_{j0}^{(n)} = -\eta \varepsilon_j^{(n)} \quad (15)$$

$$\text{Hidden - Hidden (Feedback): } \Delta v_{jx}^{(n)} = -\eta \varepsilon_j^{(n)} x_j^{(n-1)} \quad (16)$$

where  $\eta$  is the learning rate;  $\varepsilon_j^{(n)}$  is an error of hidden neuron unit ( $i^{\text{th}}$ ) at the time ( $n$ );  $u_i^{(n)}$  an output signal of input neuron ( $i^{\text{th}}$ ) at the time ( $n$ ); and  $x_j^{(n-1)}$  is the output signal of the hidden neuron ( $j^{\text{th}}$ ) at the time ( $n-1$ ).

- 7) Weights update is done after feedforward and backward are finished or after one iteration. Fix (update) the weight and the bias values for each iteration.
- Correction of the weights on the hidden unit shown in (17), (18), and (19).

$$\text{Hidden - Input: } \Delta v_{ji} = \frac{\sum_{n=1}^{NT} (\Delta v_{ji}^{(n)})}{NT} \quad (17)$$

$$\text{Hidden - Bias: } \Delta v_{j0} = \frac{\sum_{n=1}^{NT} (\Delta v_{j0}^{(n)})}{NT} \quad (18)$$

$$\text{Hidden - Feedback: } \Delta v_{jx} = \frac{\sum_{n=1}^{NT} (\Delta v_{jx}^{(n)})}{NT} \quad (19)$$

- Correction of the weights on the output unit shown in (20), (21), and (22).

$$\text{Output - Hidden: } \Delta w_{ji} = \frac{\sum_{n=1}^{NT} (\Delta w_{ji}^{(n)})}{NT} \quad (20)$$

$$\text{Output - Bias: } \Delta w_{j0} = \frac{\sum_{n=1}^{NT} (\Delta w_{j0}^{(n)})}{NT} \quad (21)$$

$$\text{Output - Feedback: } \Delta w_{jx} = \frac{\sum_{n=1}^{NT} (\Delta w_{jx}^{(n)})}{NT} \quad (22)$$

- Each hidden unit improves its weight, as shown in (23).

$$v_{ji}^{\text{next\_epoch}} = v_{ji}^{\text{now\_epoch}} + \Delta v_{ji} \quad (23)$$

- Each output unit improves its weight, as shown in (24).

$$w_{ji}^{\text{next\_epoch}} = w_{ji}^{\text{now\_epoch}} + \Delta w_{ji} \quad (24)$$

- 8) Calculate the root mean square error (RMSE) value for each iteration shown in (25).

$$RMSE = \sqrt{\frac{\sum_{n=1}^{NT} (t^{(n)} - y^{(n)})^2}{NT}} \quad (25)$$

where  $NT$  is the amount of *time*;  $t^{(n)}$  is target at the time ( $n$ ); and  $y^{(n)}$  is the output signal of output neuron  $j^{\text{th}}$  at the time ( $n-1$ );

Repeat steps 5-8 for the next iteration or epoch until the error value meets the minimum error or the number of iterations has met the maximum epoch.

## 2.5. Linear data normalization (Min-Max)

Input data normalization aims to adjust the value of the data range with the activation function of the BPTT algorithm. So that the input range that meets the requirements is the input data value from 0 to 1 or from -1 to 1. Therefore, the resulting output will be in the range of 0 to 1. Then to get the actual value of the output, a denormalization process needs to be done again [23], [24]. Normalize data using (26):

$$y = \frac{x_i - x_{min}}{x_{max} - x_{min}} \quad (26)$$

where  $y$  is the normalized value;  $x_i$  is the ( $i^{th}$ ) data;  $x_{min}$  is data with a minimum value; and  $x_{max}$  is data with a maximum value. While denormalization of data using (27):

$$x_i = y(x_{max} - x_{min}) + x_{min} \tag{27}$$

### 3. RESULTS AND DISCUSSION

#### 3.1. Literature review

A machine learning model that uses feed-forward neural networks has a very good performance in predicting future prices including cryptocurrencies, both long-term and short-term [25]. In the review [26] the author compares several methods for predicting cryptocurrency price volatility. The result of the review is a multi-layer Perceptron artificial neural network, by maximizing the posterior value, namely Bayesian neural networks (BNN) is the method with the best accuracy [27], besides BNN, the ridge regression method has the lowest RMSE [28]. Research in predicting the future value of cryptocurrency volatility has been developed with various hybrid systems [29], one of which has significant efficiency, namely the system using the long short term memory (LSTM) and gated recurrent units (GRU) approaches that focus on predicting the movement of abbreviations. of the cryptocurrency market [30]. In addition to these models, multiple-input deep neural network (MICDL) model by applying convolutional neural networks (CNN) and LSTM is the most accurate model in performance prediction [31].

The application of particle swarm optimization (PSO) to the backpropagation ANN method to predict stock market indexes has been proven to increase the accuracy of stock market predictions, especially NASDAQ-100 stocks, or other financial data analysis [32]. In the case of stock market predictions on the NASDAQ-100 data index, the hybrid model used, namely the adaptive neuro-fuzzy inference system (ANFIS) and genetic algorithm (GA) proved to be better in performance than using only the ANFIS standard, but using a hybrid model that it takes time longer [33].

#### 3.2. Process flow

The process flow of the foreign exchange forecasting system starts with the user entering data on the USD buy exchange rate against the IDR. Then the system normalizes these values. The results of the normalization process are included in the training process to produce output values. The output results are denormalized to get the value of the forecast results. The process flow of this foreign exchange forecasting system is illustrated in the form of business process model and notation (BPMN) in Figure 2.

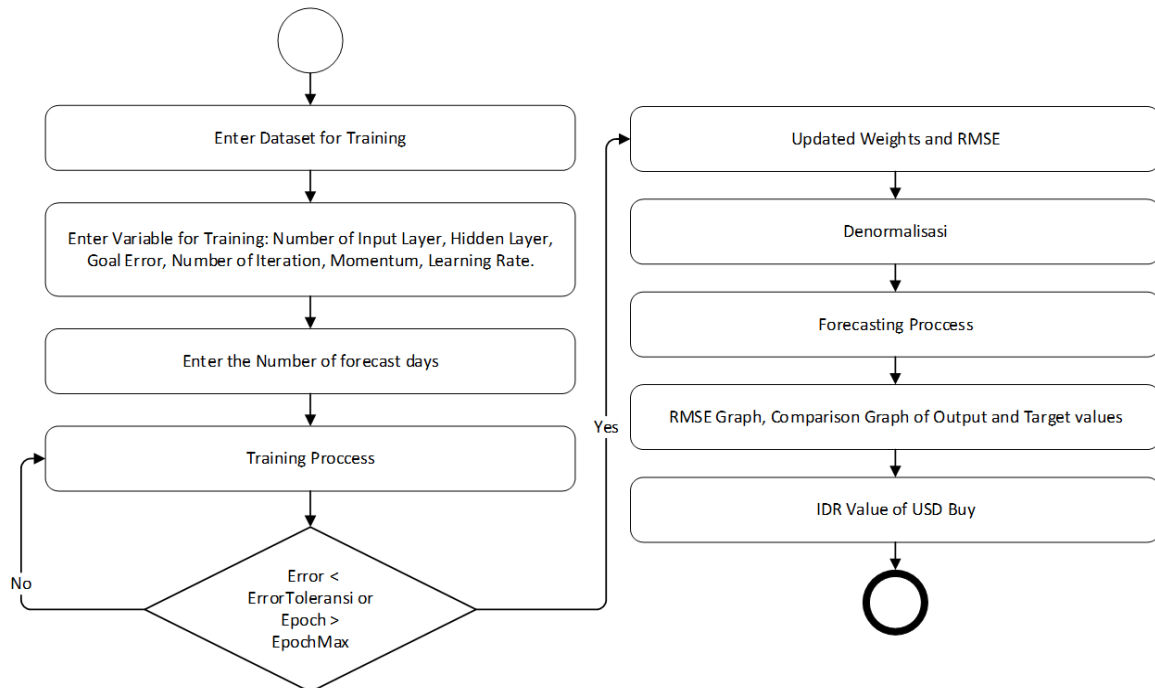


Figure 2. BPMN system

### 3.3. Data

#### 3.3.1. Input data

In the training process, the data used is the USD Buy rate against IDR from December 1<sup>st</sup>, 2020 to February 28<sup>th</sup>, 2021. This system allows data to be retrieved in the TXT (.txt) format. The data input matrix in the training process is described in Table 2 with a total of 90 data [34].

Table 2. Input data

No	Date	USD Buy
1	12/01/2020	14.107,11
2	12/02/2020	14.093,18
3	12/03/2020	14.106,12
...	...	...
90	2/28/2021	14.157,86

#### 3.3.2. Normalization data result

Before entering the training process, the USD Buy rate data is processed first using the linear normalization function. The purpose of the normalization process is to convert the initial data into data that is ready to be trained, with a value range of 0 to 1. The normalized data is shown in Table 3. The opposite of the normalization process is the denormalization process. The denormalization process is returning the value of the output result to a USD buy rate which can be used as the result of the predicted rate.

Table 3. Normalized data

No	Date	USD Buy
1	12/01/2020	0.7444685894903217
2	12/02/2020	0.7100701303832487
3	12/03/2020	0.742023903595419
...	...	...
90	2/28/2021	0.8697896088502582

#### 3.3.3. Training data

Because the number of days is 7 days, the number of histories used as the input layer is 7 input layers representing the 1<sup>st</sup> to the 7<sup>th</sup> previous days, or (t-7). The training data used are the buy USD rate (t-7), (t-6), (t-5), (t-4), (t-3), (t-2), (t-1), and with a target value of USD Buy(t). The data that will be processed as the training data are listed in Table 4.

Table 4. Training data

No	Date	USD Buy (t-7)	...	USD Buy (t-1)	USD Buy (t)
1	12/09/2020	0.7100701303832487	...	0.7100701303832487	0.7100701303832487
2	12/10/2020	0.742023903595419	...	0.7100701303832487	0.6265310154089306
3	12/11/2020	0.7542967206637698	...	0.6265310154089306	0.5577340971947848
...	...	...	...	...	...
82	2/28/2021	0.5159768866060851	...	0.8697896088502582	0.8697896088502582

### 3.4. Evaluation of activation function

The activation function is a function used to calculate the output value of each neuron in each layer. In this study, the activation functions being compared are the linear transfer function (Linear), the tan-sigmoid transfer function (Tansig), and the log-sigmoid transfer function (Logsig), which will be applied alternately to the hidden layer and output layer. Table 5 is the result of a comparison of the root mean square error (RMSE) values at the end of the training process with a total of 4000 iterations, using the BPTT-Elman RNN method.

Table 5. Comparison of the implementation of the activation function at the hidden layer and the output layer

Hidden	Logsig	Logsig	Logsig	Tansig	Tansig	Tansig	Linear	Linear	Linear
Output	Logsig	Tansig	Linear	Logsig	Tansig	Linear	Logsig	Tansig	Linear
RMSE	0.08567	0.08156	0.07664	0.14826	0.08686	-	0.38269	-	-

Figure 3 (see in appendix) is a comparison of the graphs of changes between target and output values. A graph of changes in the target value is shown in Figure 3(a). The graphs of changes in output values are Figure 3(b) shows output graph with Logsig activation function in the hidden and output layer, and Figure 3(c) shows an output graph with Logsig activation function in the hidden layer and Tansig activation function in the output layer, Figure 3(d) shows output graph with Logsig activation function in the hidden layer and linear activation function in the output layer, Figure 3(e) shows output graph with Tansig activation function in the hidden layer and Logsig activation function in the output layer, Figure 3(f) shows output graph with Tansig activation function in the hidden and output layer, and Figure 3(g) shows output graph with linear activation function in the hidden layer and Logsig activation function in the output layer. From the results of the implementation of the activation function of the linear transfer function, Tan-sigmoid transfer function, and log-sigmoid transfer function, it can be concluded that the log-sigmoid transfer function is most appropriate for use in the hidden layer, and the linear transfer function is most appropriate for the output layer, with a value RMSE is 0.07664 in the 4000<sup>th</sup> iteration.

### 3.5. Evaluation of network architecture performance

The following tests were conducted to compare the performance of the Elman and Jordan RNN architectures with the BPTT algorithm. The parameters used are the number of input layer units=7, the number of hidden layers=7; minimum error=0.001; momentum=0; learning rate=0.01; maximal epoch=10,000; forecasting for the next 7 days; and used the log-sigmoid transfer function in the hidden layer, and the linear transfer function in the output layer. The number of input layers is 7 layers, adjusted for the number of days in 1 week. The minimum number of hidden layers is the input layer, in this study we use 7 input layers. Table 6 represents the changes in the RMSE value, as well as a graph of the comparison of output and targets.

Table 6. Comparison of RMSE training value with BPTT RNN architecture

Iteration Number	Elman RNN	Jordan RNN
1000	0.082832	0.137871
2000	0.077480	0.148271
3000	0.074951	0.132359
4000	0.073477	0.130317
5000	0.110103	0.131779
6000	0.115928	0.131267
7000	0.123828	0.130434
8000	0.138169	0.148271
9000	0.123078	0.148271
10000	0.147858	0.148879
Best Iteration	<b>4000</b>	<b>4000</b>
Best RMSE value	0.073477	0.130317

Figure 4 is the graphical result of the training and forecasting process using the BPTT-Elman RNN method, where the RMSE change graph is shown in Figure 4(a), and the graph of the output in the training process and forecasting values for the next 7 days is shown in Figure 4(b). Figure 5 (see in appendix) is the graph result of the training and forecasting process using the BPTT-Jordan RNN method, where the RMSE change graph is shown in Figure 5(a), and a graph output in the training process and forecasting values for the next 7 days is shown in Figure 5(b). From the test results above, it is known that the best iteration of the Elman RNN-BPTT method is the 4000<sup>th</sup> iteration with the lowest RMSE value of 0.073477. While the best iteration of the Jordan backpropagation RNN method is the 4000<sup>th</sup> iteration with the lowest RMSE value of 0.130317. So, it can be concluded that the Elman backpropagation RNN method is better than the Jordan RNN-BPTT method in the training process.

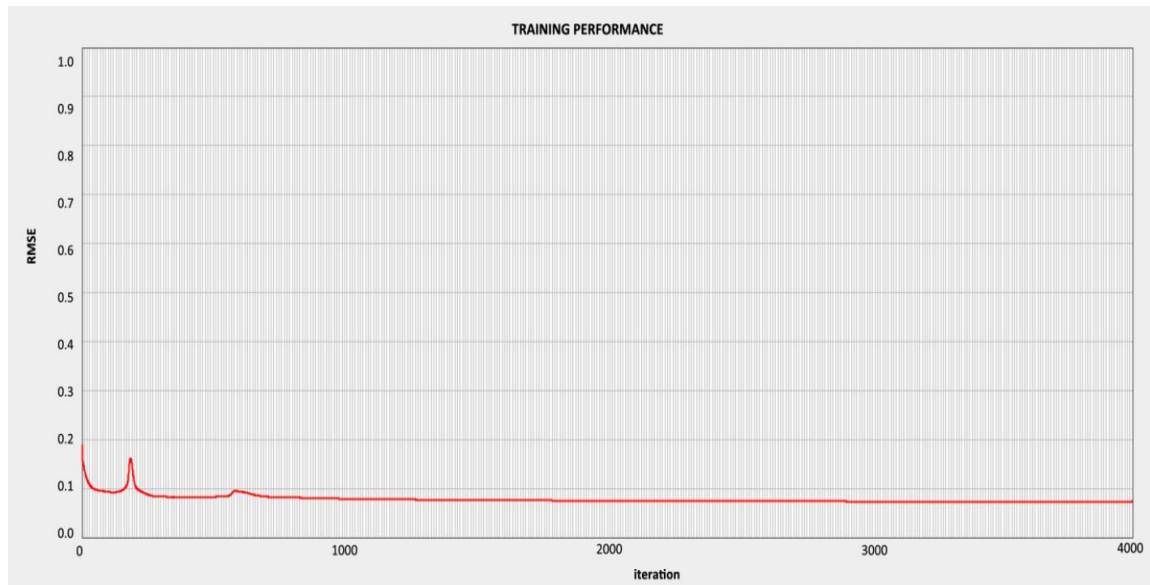
### 3.6. Forecasting performance evaluation

The forecasting process is carried out after the training process. In the forecasting process, the process of denormalizing the resulting output value is carried out. Furthermore, in the forecasting test, the USD buy value is calculated for the next 7 days, then it is compared with the actual USD buy rate, namely March 1<sup>st</sup> to 7<sup>th</sup> 2021, to determine the RMSE value for the forecasting process. Table 7 displays forecasting results for the next 7 days compared to the actual USD buy rate.

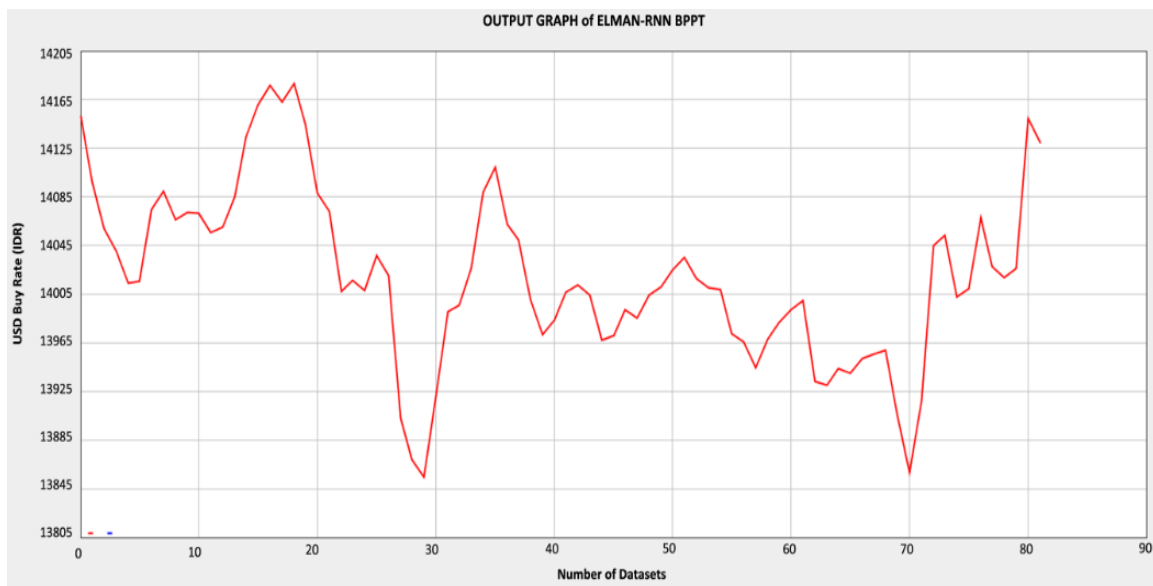
From the calculation of RMSE forecasting USD buy for the next 7 days, it is known that the lowest RMSE value is on the next very first following day by the Elman backpropagation RNN Method with an RMSE value of 122.15 and the Jordan backpropagation RNN method with an RMSE value 222.96. So, it can



be concluded that the Elman backpropagation RNN method is better than the Jordan backpropagation RNN method in the case of the foreign exchange forecasting system.



(a)



(b)

Figure 4. Graph of Elman-RNN BPTT results (a) RMSE graph and (b) output results graph

Table 7. RMSE forecasting process

NO	USD BUY	FORECASTING RESULT		DIFFERENCE		RMSE	
		ELMAN	JORDAN	ELMAN	JORDAN	ELMAN	JORDAN
1	14.228,50	14.106,35	14.005,54	122,15	222,96	122,15	222,96
2	14.235,47	14.090,06	14.005,00	145,41	230,47	134,28	226,75
3	14.262,33	14.069,06	14.004,25	193,27	258,08	156,44	237,65
4	14.227,51	14.065,50	14.003,87	162,01	223,64	157,85	234,23
5	14.299,15	14.076,26	14.003,80	222,89	295,35	172,83	247,66
6	14.299,15	14.067,96	14.003,71	231,19	295,44	183,85	256,24
7	14.299,15	14.062,39	14.003,54	236,76	295,61	192,30	262,23

#### 4. CONCLUSION

In this foreign exchange forecasting system research, trials were carried out by implementing activation functions, such as the linear transfer function, the tan-sigmoid transfer function, and the log-sigmoid transfer function. From these trials, it can be concluded that the log-sigmoid transfer function is most appropriate for use in the hidden layer, and the linear transfer function is most suitable for the output layer, with an RMSE value of 0.07664 in the 4000<sup>th</sup> iteration. In addition, the backpropagation through time (BPTT) method is also implemented on the RNN network architecture, namely Elman and Jordan for the training process. From these trials, it was concluded that the best iteration was the 4000<sup>th</sup> iteration with the lowest RMSE value of 0.073477 for Elman and 0.130317 for Jordan in the training process. So, it can be concluded that the BPTT-Elman RNN method is better than the BPTT-Jordan RNN method in the training process. For testing the forecasting process, the BPTT method was applied to Elman and Jordan, then a comparison was made of the RMSE results. From the results of the RMSE buy USD forecasting calculations in the next 7 days, the lowest RMSE value is obtained the following day with the BPTT-Elman RNN method of 122.15 and with the BPTT-Jordan RNN method of 222.96. So, it can be concluded that the BPTT method applied to the Elman RNN architecture is better than the Jordan RNN in the case of forecasting the USD currency against IDR.

#### APPENDIX

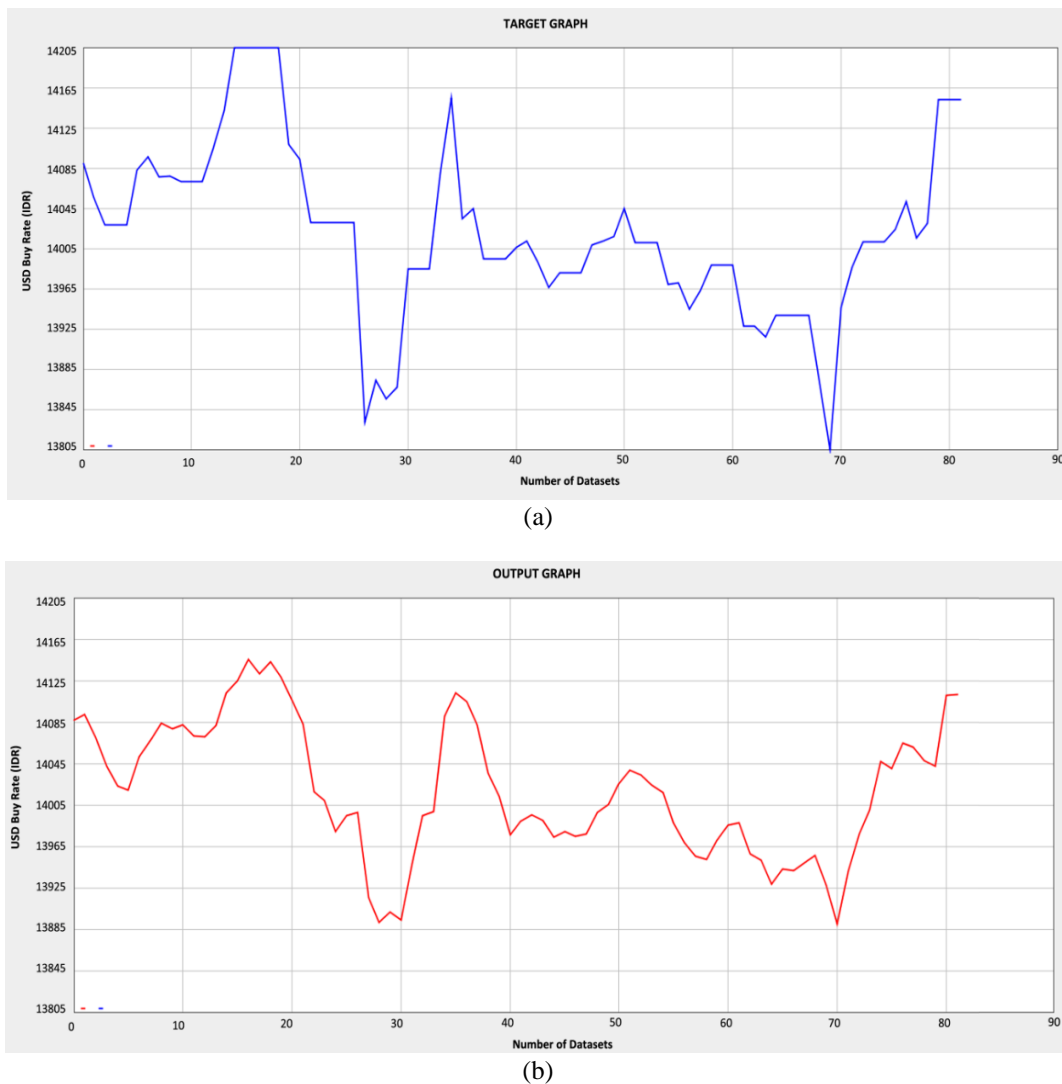
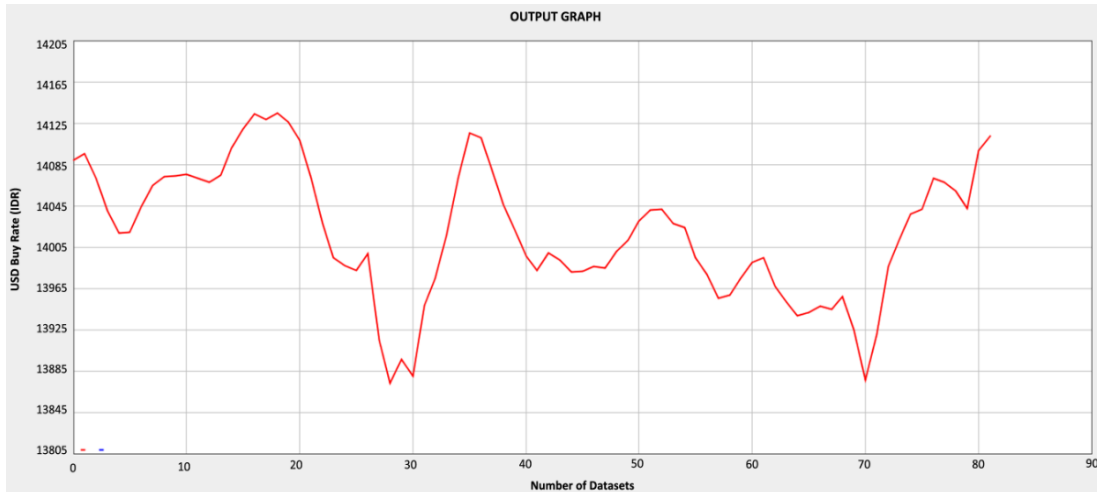
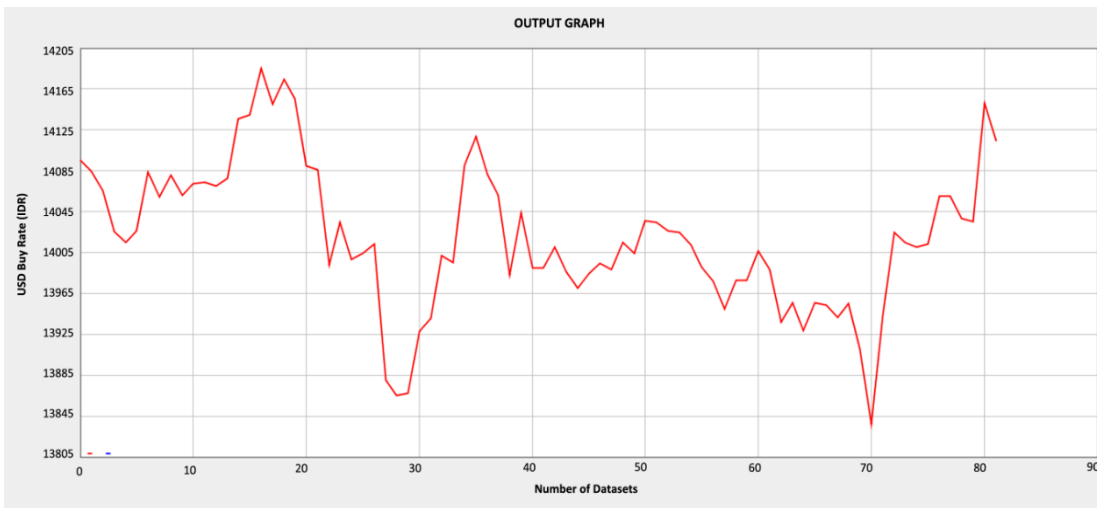


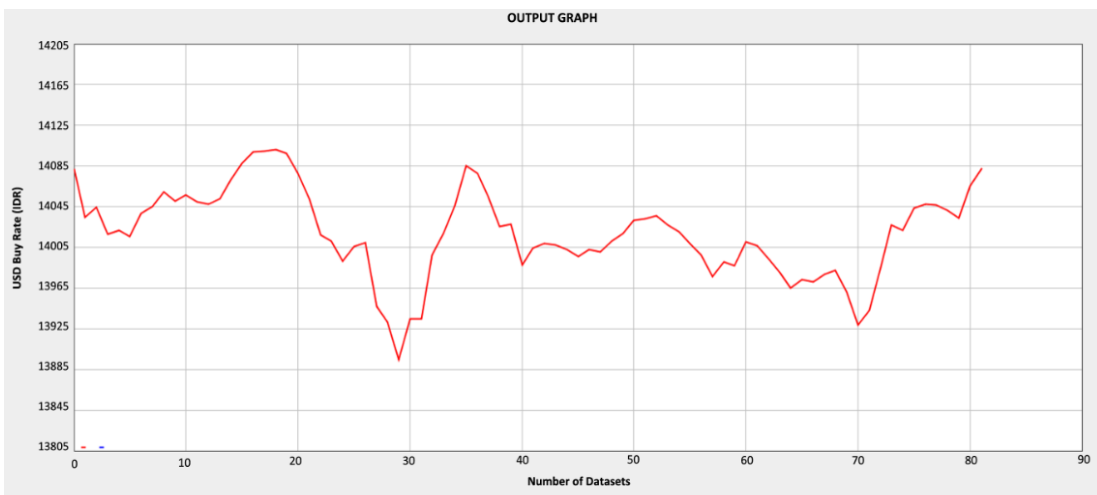
Figure 3. Graphs of changes between target and output values (a) target graph (b) output graph with Logsig activation function in the hidden and output layer (*continue*)



(c)

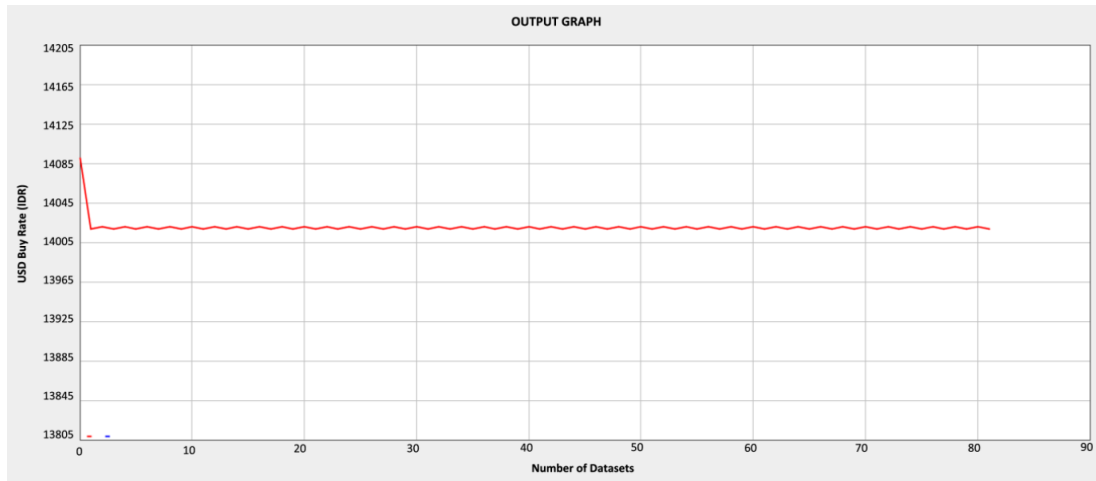


(d)

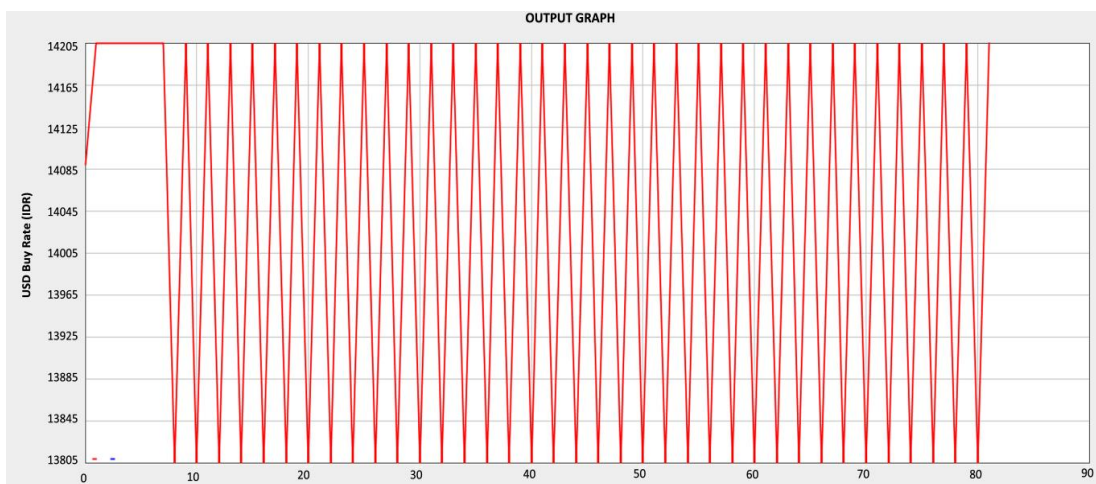


(e)

Figure 3. Graphs of changes between target and output values (c) output graph with Logsig activation function in the hidden layer and Tansig activation function in the output layer, (d) output graph with Logsig activation function in the hidden layer and linear activation function in the output layer, and (e) output graph with Tansig activation function in the hidden layer and Logsig activation function in the output layer  
(continue)

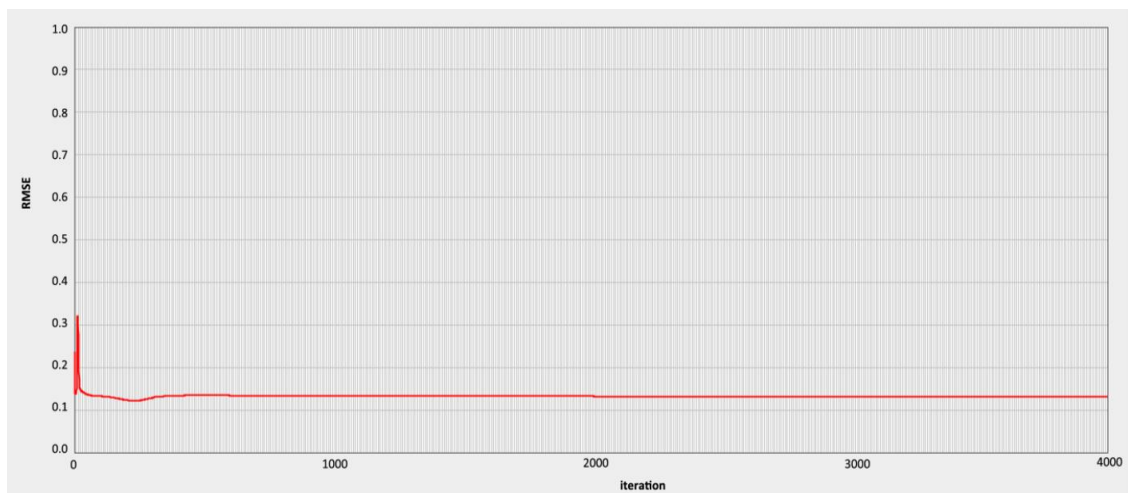


(f)



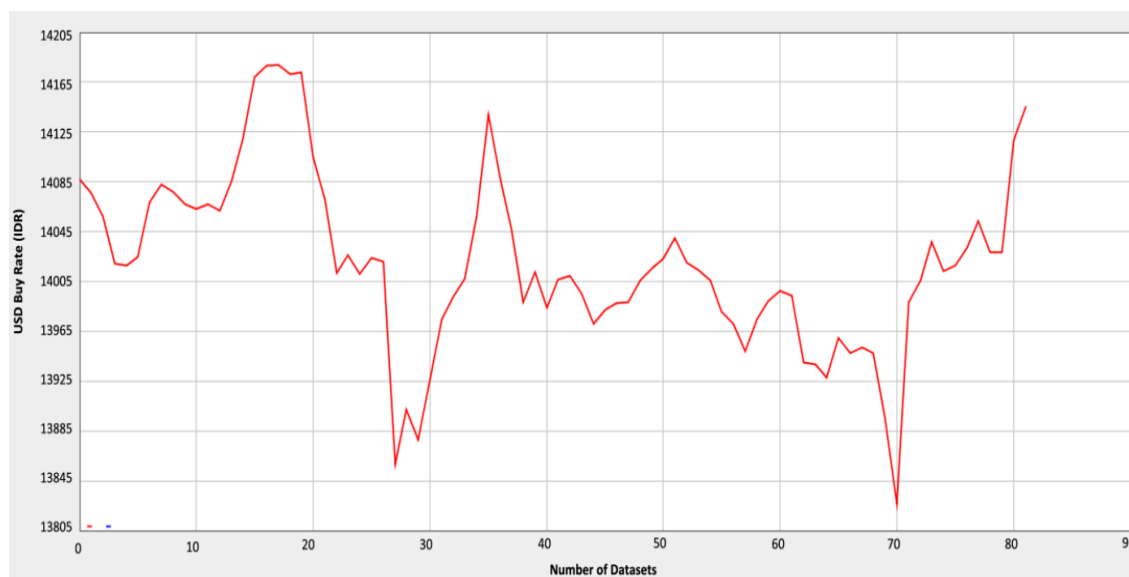
(g)

Figure 3. Graphs of changes between target and output values (f) output graph with Tansig activation function in the hidden and output layer and (g) output graph with linear activation function in the hidden layer and Logsig activation function in the output layer



(a)

Figure 5. Graph of Jordan-RNN BPTT results (a) RMSE graph (*continue*)



(b)

Figure 5. Graph of Jordan-RNN BPTT results (b) Output results graph




## REFERENCES

- [1] N. S. Gill, "Artificial neural networks applications and algorithms," *XenonStack*, 2021. <https://www.xenonstack.com/blog/artificial-neural-network-applications> (accessed Nov. 03, 2021).
- [2] M. Shawaqfah and F. Almomani, "Forecast of the outbreak of COVID-19 using artificial neural network: Case study Qatar, Spain, and Italy," *Results in Physics*, vol. 27, Aug. 2021, doi: 10.1016/j.rinp.2021.104484.
- [3] G. Toğa, B. Atalay, and M. D. Toksari, "COVID-19 prevalence forecasting using autoregressive integrated moving average (ARIMA) and artificial neural networks (ANN): case of Turkey," *Journal of Infection and Public Health*, vol. 14, no. 7, pp. 811–816, Jul. 2021, doi: 10.1016/j.jiph.2021.04.015.
- [4] A. Marvuglia and A. Messineo, "Using recurrent artificial neural networks to forecast household electricity consumption," *Energy Procedia*, vol. 14, pp. 45–55, 2012, doi: 10.1016/j.egypro.2011.12.895.
- [5] S. Sharma, "Activation functions in neural networks," *Towards Data Science*, 2017. <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6> (accessed Nov. 05, 2021).
- [6] "Multilayer shallow neural network architecture," *MathWorks*. <https://www.mathworks.com/help/deeplearning/ug/multilayer-neural-network-architecture.html> (accessed Nov. 08, 2021).
- [7] Q. W. Luthuli and K. A. Folly, "Short term load forecasting using artificial intelligence," in *2016 IEEE PES PowerAfrica*, Jun. 2016, pp. 129–133, doi: 10.1109/PowerAfrica.2016.7556585.
- [8] A. A. Firdaus, R. T. Yunardi, E. I. Agustin, T. E. Putri, and D. O. Anggriawan, "Short-term photovoltaics power forecasting using Jordan recurrent neural network in Surabaya," *Telecommunication Computing Electronics and Control (TELKOMNIKA)*, vol. 18, no. 2, Apr. 2020, doi: 10.12928/telkomnika.v18i2.14816.
- [9] K. E. ArunKumar, D. V. Kalaga, C. M. S. Kumar, M. Kawaji, and T. M. Brenza, "Forecasting of COVID-19 using deep layer recurrent neural networks (RNNs) with gated recurrent units (GRUs) and long short-term memory (LSTM) cells," *Chaos, Solitons and Fractals*, vol. 146, May 2021, doi: 10.1016/j.chaos.2021.110861.
- [10] J. Brownlee, "Time series prediction with LSTM recurrent neural networks in python with Keras," *Machine Learning Mastery*, 2016. <https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/> (accessed Nov. 08, 2021).
- [11] K. Park, J. Kim, and J. Lee, "Visual field prediction using recurrent neural network," *Scientific Reports*, vol. 9, no. 1, Jun. 2019, doi: 10.1038/s41598-019-44852-6.
- [12] A. Hassan and A. Mahmood, "Convolutional recurrent deep learning model for sentence classification," *IEEE Access*, vol. 6, pp. 13949–13957, 2018, doi: 10.1109/ACCESS.2018.2814818.
- [13] P. Tenti, "Forecasting foreign exchange rates using recurrent neural networks," *Applied Artificial Intelligence*, vol. 10, no. 6, pp. 567–582, Dec. 1996, doi: 10.1080/088395196118434.
- [14] J. Park, D. Yi, and S. Ji, "Analysis of recurrent neural network and predictions," *Symmetry*, vol. 12, no. 4, Apr. 2020, doi: 10.3390/sym12040615.
- [15] A. Alamsyah, B. Prasetyo, M. F. Al Hakim, and F. D. Pradana, "Prediction of COVID-19 using recurrent neural network model," *Scientific Journal of Informatics*, vol. 8, no. 1, pp. 98–103, May 2021, doi: 10.15294/sji.v8i1.30070.
- [16] A. Y. Alanis, N. Arana-Daniel, and C. Lopez-Franco, *Artificial neural networks for engineering applications*. Academic Press, 2019.
- [17] Z. Berradi and M. Lazaar, "Integration of principal component analysis and recurrent neural network to forecast the stock price of Casablanca stock exchange," *Procedia Computer Science*, vol. 148, pp. 55–61, 2019, doi: 10.1016/j.procs.2019.01.008.
- [18] K. N. Nabi, M. T. Tahmid, A. Rafi, M. E. Kader, and M. A. Haider, "Forecasting COVID-19 cases: A comparative analysis between recurrent and convolutional neural networks," *Results in Physics*, vol. 24, May 2021, doi: 10.1016/j.rinp.2021.104137.




- [19] A. K. Rout, P. K. Dash, R. Dash, and R. Bisoi, "Forecasting financial time series using a low complexity recurrent neural network and evolutionary learning approach," *Journal of King Saud University-Computer and Information Sciences*, vol. 29, no. 4, pp. 536–552, Oct. 2017, doi: 10.1016/j.jksuci.2015.06.002.
- [20] T. P. Lillicrap and A. Santoro, "Backpropagation through time and the brain," *Current Opinion in Neurobiology*, vol. 55, pp. 82–89, Apr. 2019, doi: 10.1016/j.conb.2019.01.011.
- [21] L. S. Moonlight, F. Faizah, Y. Suprpto, and N. Pambudiyatno, "Comparison of backpropagation and kohonen self organising map (KSOM) methods in face image recognition," *Journal of Information Systems Engineering and Business Intelligence*, vol. 7, no. 2, Oct. 2021, doi: 10.20473/jisebi.7.2.149-161.
- [22] L. S. Moonlight and A. Setiyo Prabowo, "Forecasting system for passenger, airplane, luggage and cargo, using artificial intelligence method-backpropagation neural network at Juanda International Airport," *Warta Ardhia*, vol. 45, no. 2, Jun. 2021, doi: 10.25104/wa.v45i2.358.99-110.
- [23] S. Loukas, "Everything you need to know about Min-Max normalization: A Python tutorial," *Towards Data Science*, 2020. <https://towardsdatascience.com/everything-you-need-to-know-about-min-max-normalization-in-python-b79592732b79> (accessed May 22, 2021).
- [24] J. Han and M. Kamber, "Data mining: concepts and techniques," 3<sup>rd</sup> Edition, *Morgan Kaufmann*, vol. 10, pp. 559–569, 2006.
- [25] S. E. Charandabi and K. Kamyar, "Using a feed forward neural network algorithm to predict prices of multiple cryptocurrencies," *European Journal of Business and Management Research*, vol. 6, no. 5, pp. 15–19, Sep. 2021, doi: 10.24018/ejbmr.2021.6.5.1056.
- [26] S. E. Charandabi and K. Kamyar, "Survey of cryptocurrency volatility prediction literature using artificial neural networks," *Business and Economic Research*, vol. 12, no. 1, Jan. 2022, doi: 10.5296/ber.v12i1.19301.
- [27] H. Jang and J. Lee, "An empirical study on modeling and prediction of bitcoin prices with bayesian neural networks based on blockchain information," *IEEE Access*, vol. 6, pp. 5427–5437, 2018, doi: 10.1109/ACCESS.2017.2779181.
- [28] R. Miura, L. Pichl, and T. Kaizoji, "Artificial neural networks for realized volatility prediction in cryptocurrency time series," in *Advances in Neural Networks ISNN 2019*, Springer International Publishing, 2019, pp. 165–172.
- [29] S. E. Charandabi and K. Kamyar, "Prediction of cryptocurrency price index using artificial neural networks: a survey of the literature," *European Journal of Business and Management Research*, vol. 6, no. 6, pp. 17–20, Nov. 2021, doi: 10.24018/ejbmr.2021.6.6.1138.
- [30] P. Jaquart, D. Dann, and C. Weinhardt, "Short-term bitcoin market prediction via machine learning," *The Journal of Finance and Data Science*, vol. 7, pp. 45–66, Nov. 2021, doi: 10.1016/j.jfds.2021.03.001.
- [31] I. E. Livieris, N. Kiriakidou, S. Stavroyiannis, and P. Pintelas, "An advanced CNN-LSTM model for cryptocurrency forecasting," *Electronics*, vol. 10, no. 3, Jan. 2021, doi: 10.3390/electronics10030287.
- [32] F. Ghashami, K. Kamyar, and S. A. Riazi, "Prediction of stock market index using a hybrid technique of artificial neural networks and particle swarm optimization," *Applied Economics and Finance*, vol. 8, no. 3, Apr. 2021, doi: 10.11114/aef.v8i3.5195.
- [33] F. Ghashami and K. Kamyar, "Performance evaluation of ANFIS and GA-ANFIS for predicting stock market indices," *International Journal of Economics and Finance*, vol. 13, no. 7, Jun. 2021, doi: 10.5539/ijef.v13n7p1.
- [34] BI, "Foreign Exchange Rates," *Bank Indonesia*. <https://www.bi.go.id/id/statistik/informasi-kurs/transaksi-bi/default.aspx> (accessed Apr. 20, 2021).

## BIOGRAPHIES OF AUTHORS







**Lady Silk Moonlight**    was born in Surabaya, Indonesia, in 1987. She received a Bachelor of Engineering degree in Informatics Engineering from Trunojoyo University in 2009 and a Master of Engineering degree in Informatics from Bandung Institute of Technology (ITB) in 2013. Her research interests include information technology, artificial intelligence, information systems, computer science, digital image processing, and computer networks. She is a lecturer at D3 Aeronautical Communication, Politeknik Penerbangan Surabaya, Indonesia. She can be contacted at email: lady@poltekbangsby.ac.id.







**Bambang Riyanto Trilaksono**    obtained his first degree in Electrical Engineering from Electrical Engineering in the Bandung Institute of Technology (ITB), and a Master's and Ph.D. degree in Electrical Engineering, Waseda University, Japan. He is currently a Professor in Control and Computer System Research Group in ITB. His research interest includes robust control, intelligent control and intelligent systems, control applications, robotics, and embedded control systems. He can be contacted at email: briyanto@lkk.ee.itb.ac.id.



**Bambang Bagus Harianto**     is a lecturer at D3 Air Navigation Engineering, Politeknik Penerbangan Surabaya, Indonesia. He received a B.Eng. degree in D4 Air Telecommunication and Navigation Engineering from Sekolah Tinggi Penerbangan Indonesia Curug, Indonesia, and a Master's degree in Telecommunication Multimedia Engineering from the Institut Teknologi Sepuluh Nopember in Surabaya Indonesia. He is currently a Doctoral student at Universitas Negeri Surabaya (UNESA), Indonesia. He can be contacted at email: bambangfarzardy@gmail.com.



**Fiqqih Faizah**     received a Bachelor's degree in Electrical Engineering from Jember University in 2008 and a Master's degree in Electrical Engineering from Bandung Institute of Technology (ITB) in 2014. She is a lecturer at D3 Airport Electrical Engineering, Politeknik Penerbangan Surabaya. She can be contacted at [fiqqihfaizah@poltekbangsby.ac.id](mailto:fiqqihfaizah@poltekbangsby.ac.id).