Available online at http://scik.org

Commun. Math. Biol. Neurosci. 2021, 2021:87

https://doi.org/10.28919/cmbn/6683

ISSN: 2052-2541

COMPARATION OF ELMAN NEURAL NETWORK, LONG SHORT-TERM MEMORY, AND GATED RECURRENT UNIT IN PREDICTING DENGUE HEMORRHAGIC FEVER AT DKI JAKARTA

BEVINA D. HANDARI*, IMANNUEL M. S. NIMAN, ABDULLAH HASAN, JUSUP R. P. PURBA, GATOT F. HERTONO

Department of Mathematics, Universitas Indonesia, Depok 16424, Indonesia

Copyright © 2021 the author(s). This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract: Dengue hemorrhagic fever (DHF) is a seasonal disease that has quickly spread throughout the world in the past few years. It is caused by the dengue virus that is transmitted by a biological vector in the form of mosquitoes. In DKI Jakarta, the capital of Indonesia, there were 970 DHF cases at the beginning of 2020, thus placing Jakarta in the red zone for the spread of DHF. Besides, DHF can emerge due to various weather and climate factors such as humidity, rainfall, and temperature. With a significant increase in the number of potentially fatal DHF cases in DKI Jakarta, preventing DHF outbreaks is recommended. This research aims to predict the number of DHF cases in DKI Jakarta using weather and DHF case data. Three machine learning models were employed to predict DHF case numbers: Elman neural network (ENN), long short-term memory (LSTM), and gated recurrent unit (GRU). ENNs have a simplified recurrent neural network (RNN) structure, LSTM is a modified RNN with long-range memory and an activation function for deciding which information should be retained or discarded, and GRUs are modified RNNs that are slightly simpler than LSTMs. These methods were implemented in three different data sets as follows: one with 90% training data and 10% testing data, another with 80% training data and 20% testing data, and finally, one with 70% training data and 30% testing data. A grid search is used to determine the best hyperparameter from all three

*Corresponding author

E-mail address: bevina@sci.ui.ac.id

Received August 23, 2021

methods. This will determine the best model for predicting the number of DHF cases in DKI Jakarta (excluding Kepulauan Seribu Regency) according to the data used, RMSE values, and the simulation results. Based on these criteria, LSTM is better suited to predicting the number of DHF cases than ENN or GRU in almost every district in DKI Jakarta.

Keywords: dengue hemorrhagic fever; machine learning; recurrent neural network; elman neural network; long short-term memory; gated recurrent unit.

2010 AMS Subject Classification: 68T05, 92B99.

1. INTRODUCTION

Dengue hemorrhagic fever (DHF) is a disease that is transmitted by mosquitoes. Over the past few years, DHF has rapidly spread throughout the world [1]. Every year, around 400 million people are infected by this climate-sensitive disease, among whom 22,000 die. According to CDC reports [2], this disease is common in more than 100 countries, including Indonesia, where cases are either "frequent or continuous." Indonesia has the second highest number of cases out of the 30 countries where DHF is endemic [3]. The sheer number of DHF incidents has placed DKI Jakarta, the capital of Indonesia, in the red zone of DHF [4].

DHF is caused by the dengue virus and spread by the *Aedes aegypti* mosquito. The dengue virus is commonly found in tropical and subtropical regions, especially cities and suburbs. Indonesia has a tropical climate that is conducive to the growth of mosquito vectors [4]. In severe cases, DHF can cause a decrease in red blood cell (thrombocyte) count, the loss of blood plasma, and severe hemorrhages that can be fatal. DHF emerges due to various environmental factors such as humidity, rainfall, temperature, and so on. For example, increasing temperature and precipitation make *Aedes* larvae mature into pupae more rapidly, thereby increasing the *Aedes* population. On the other hand, a temperature above 35°C combined with low humidity will decrease the *Aedes* population [5].

The Indonesian government has made several efforts to reduce the number of DHF incidents in Indonesia, such as controlling the insects that act as vectors through fumigation, use of bio-larvicides, etc. The budget dedicated to DHF control is a key issue for the government. If an

outbreak is known through accurate prediction of case numbers, the budget can then be safely controlled.

Certain research that predicted DHF cases by considering climate factors were published in the past, including the results of developing a DHF prediction model in China using long short-term memory (LSTM) by calculating measurements based on related weather variables [5]. In addition, there is a DHF prediction model in Yogyakarta that employs climate and surveillance data as predictor variables [6]. A neural network is used to predict the number of DHF sufferers in Semarang [7]. An extreme learning machine in the form of a neural network with exactly one hidden layer is used to predict the number of DHF cases in Tembalang Municipality, Semarang using weather factors [8].

Murphy [9] defines machine learning as a method that can automatically recognize patterns in data and then use them to predict new data. In machine learning, predictive analysis can be conducted with a supervised learning method whose task is to provide predictions by extracting the necessary knowledge or information from the processed data [10]. The neural network is a machine learning model that attempts to replicate the nervous system to simulate learning mechanisms in living things. In its application, the simple neural network (NN) model has a limited ability in certain complex problems such as mining time-series data, text data, and audio data [11]. One type of NN is the recurrent neural network (RNN) which is a neural network that contains at least one loop connection, so that activation is applied repeatedly on the loop, and that uses time-series input data. The simplest RNN architecture, commonly used by researchers for making predictions, is the Elman Neural Network (ENN) [12]. ENNs are often used for predicting time-series data. The ENN is employed to predict wind speeds [13] and [14] implemented ENNs to predict the optimum fermentation time for black tea.

One of the modified versions of the RNN is Long Short-Term Memory (LSTM) [15]. This model is classified in deep learning based on the complexity of its architecture [16] and can solve problems beyond the capability of a standard RNN [15]. It has been proven to be capable of determining influenza and foot-and-mouth disease trends among others [17].

Cho et al. [18] proposed research on a Gated Recurrent Unit (GRU) that could adaptively remember and forget information based on input data obtained from gates. GRU is a modified RNN that is slightly simpler than LSTM and offers comparable performance. It is significantly faster to compute across multiple datasets. However, it is not certain which one is better [19]. The main difference between RNN and both LSTM and GRU is that the architecture of the former is supported by the gating role of the hidden state. These three machine learning methods are part of predictive analysis that employs supervised learning methods that can predict knowledge or information required from scientific data [10].

Based on the urgency for predicting the number of DHF cases accurately and the suspicion of a relation between weather factors and the widespread incidence of DHF, this research employs several machine learning methods, namely ENN, GRU, and LSTM, to predict DHF incidents based on weather and DHF incidence data.

The data used in this research originates from DHF incidents in DKI Jakarta (excluding Kepulauan Seribu Regency) obtained from the Epidemiology Surveillance section of the Health Ministry of DKI Jakarta [20] and we obtain a weather data from the Meteorology, Climatology and Geophysics Board (BMKG). It includes temperature, rainfall, and humidity data.

The paper is categorized as follows: Section 2 discusses the methods used, section 3 covers the results and discussion, and finally, section 4 provides a conclusion.

2. METHODS

2.1. Elman Neural Network

The basic concept of the Elman neural network (ENN) model was developed by Jeffrey L. Elman in 1990 [21]. An ENN has one of each input, hidden, output, and context layers. The context layer in an ENN stores the results of the previous calculation in the hidden layer. These results form the memory in the network. This memory is required for implicitly representing the time required for calculation, thereby yielding a better prediction result. The diagram of an ENN is as follows:

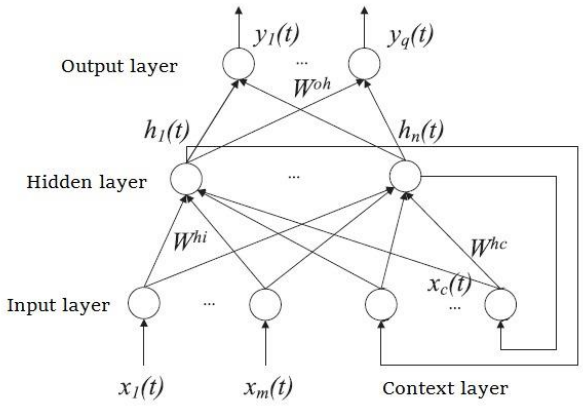

FIGURE 1. ENN Architecture ([21] has been reprocessed)

According to Figure 1, there is a connection between every neuron in the hidden layer and every neuron in the context layer, and vice versa. A neuron in the context layer stores the result of the calculation from one neuron in the hidden layer. Therefore, the weight of each connection from the hidden layer to the context layer remains one during the learning process. The context layer does not conduct an activation process on the calculation results from the hidden layer and only propagates from the input layer to the hidden layer.

Based on Figure 1., an ENN has an input layer that consists of m neurons, where $x_i(t)$, $i \in \{1,2,\dots,m\}$ denotes the i^{th} element of the t^{th} input pattern vector. The hidden layer consists of n neurons, where $h_i(t)$, $i \in \{1,2,\dots,n\}$ denotes the i^{th} element from the activation result vector in the hidden layer. The output layer consists of q neurons, where $y_i(t)$, $i \in \{1,2,\dots,q\}$ denotes the i^{th} element from the activation result vector in the output layer, and the context layer has the same number of neurons n as the hidden layer. The notation $x_c(t)$ denotes a vector that stores the activation results of the hidden layer on the calculation of the $(t-1)^{\text{th}}$ input pattern. The notations W^{hi}, W^{hc}, and W^{oh} denote the connection weight sets from the input layer to the hidden layer, the context layer to the hidden layer, and the hidden layer to the output layer respectively.

Given an input pattern $\boldsymbol{x}(t) \in R^{m\times 1}$ and a context layer vector $\boldsymbol{x_c}(t) \in R^{n\times 1}$, the input network in the hidden layer $\boldsymbol{net^h}(t) \in R^{n\times 1}$ is defined as follows:

$$\boldsymbol{net^h}(t) = W^{hi}(t)\boldsymbol{x}(t) + W^{hc}(t)\boldsymbol{x_c}(t), \tag{1}$$

where $W^{hi} \in R^{n\times m}$, $W^{hc} \in R^{n\times n}$.

The input network on the hidden layer is then activated with the activation function f_{act}. Based on the experiments from [22], the activation function on the hidden layer of ENN is the sigmoid function. The activation result on the hidden layer is the vector $\boldsymbol{h}(t) \in R^{n\times 1}$, where:

$$h_i(t) = f_{act}\left(net_i^h(t)\right), \forall i \in \{1,2,\dots,n\}. \tag{2}$$

The activation results on the hidden layer, $\boldsymbol{h}(t)$, are then propagated to the output layer to form an output network, $\boldsymbol{net^o}(t) \in R^{q\times 1}$, which is defined as follows:

$$\boldsymbol{net^o}(t) = W^{oh}(t)\boldsymbol{h}(t), \tag{3}$$

with $W^{oh} \in R^{q\times n}$. In addition, $\boldsymbol{h}(t)$ is stored in the context layer, therefore:

$$\boldsymbol{x_c}(t+1) = \boldsymbol{h}(t). \tag{4}$$

In prediction problems, the best activation function for the output layer of an ENN is the identity function [22]. Consequently, the output of the ENN model, $\boldsymbol{y}(t)$, can be stated as follows:

$$\boldsymbol{y}(t) = \boldsymbol{net^o}(t). \tag{5}$$

In this research, the learning process on the ENN uses the Mean Sum Squared Error (RMSE) function:

$$E(t) = \text{RMSE} = \sqrt{\frac{\sum_{k=1}^{q}\left(y_{d_k}(t) - y_k(t)\right)^2}{q}}, \tag{6}$$

where $\boldsymbol{y}(t)$ denotes the output of the ENN on the t^{th} input pattern, and $\boldsymbol{y_d}(t)$ denotes the target value on the t^{th} input pattern.

Changing the weight during the learning process of an ENN occurs on the weight of the connection from the input layer to the hidden layer, denoted by $W^{hi} \in R^{n\times m}$, the weight of the connection from the context layer to the hidden layer, denoted by $W^{hc} \in R^{n\times n}$, and the weight of the connection from the hidden layer to the output layer, denoted by $W^{oh} \in R^{q\times n}$. Weight changes W^{hi}, W^{hc}, and W^{oh} are calculated by counting the gradient of the error function $E(t)$ with respect to W^{hi}, W^{hc}, and W^{oh} respectively, which are determined as follows:

$$W^{oh}(t+1) = W^{oh}(t) - \eta \cdot \left(\boldsymbol{y}(t) - \boldsymbol{y_d}(t)\right) \cdot \boldsymbol{h}^T(t), \tag{7a}$$

$$W^{hi}(t+1) = W^{hi}(t) - \eta \cdot \left(\delta f'_{act}(t) \odot \left(W^{oh}(t)\right)^T \cdot \left(\boldsymbol{y}(t) - \boldsymbol{y_d}(t)\right)\right) \cdot \boldsymbol{x}^T(t), \tag{7b}$$

$$W^{hc}(t+1) = W^{hc}(t) - \eta \cdot \left(\delta f'_{act}(t) \odot \left(W^{oh}(t)\right)^T \cdot \left(\boldsymbol{y}(t) - \boldsymbol{y_d}(t)\right)\right) \cdot \boldsymbol{x}_c^T(t), \tag{7c}$$

where the constant η is the learning rate, $\delta f'_{act}(t) = [f'_{act}(net_1^h(t)) \cdots f'_{act}(net_n^h(t))]^T$, and $\odot$ denotes a Hadamard operation. The weight of the connection between the hidden and context layers is always one; the weight of the connection from the context layer to itself is always α.

The number of input patterns that are processed before changing the weight is called the batch size. Obtaining a model with a sufficiently small error function $E(t)$ usually requires a learning process in which the entire input pattern is undertaken more than once. The number of times a learning process is repeated in the entire input pattern is called an epoch. The combination of batch size and epoch determines how much the weight changes during a learning process.

2.2. Gated Recurrent Unit

A Gated Recurrent Unit (GRU) consists of feed-forward and back-propagation processes. The feed-forward process in a GRU has four gate functions (update gate, reset gate, candidate hidden state, and hidden state as outputs) which continue the error calculation using a loss function. The weight parameter that minimizes the loss function during the training stage is determined during back-propagation [23]. The loss/error function used in this research is Root Mean Squared Error (RMSE). It is employed to measure the difference between the predicted and actual values of a model. According to [24], there are several optimization algorithms generally employed to reduce the loss function in a neural network, such as Stochastic Gradient Descent (SGD); its derivatives include Adagrad, RMSProp, AdaDelta, and Adam. This study uses the Adaptive Moment Estimation (Adam) optimization method as shown in [25].

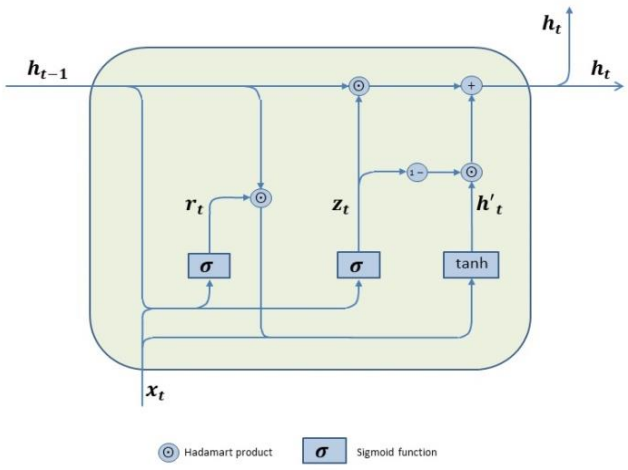

FIGURE 2. GRU Architecture

Figure 2 shows the architecture of a GRU that illustrates the feed-forward process which begins with an input process for the update gate with an input vector on timestep t, denoted by $\mathbf{x}(t) \in R^{n\times 1}$, where n is the number of features and input vectors of the hidden state from timestep $t-1$,, where $\boldsymbol{h}(t-1) \in R^{d\times 1}$. The output of this stage is provided by a layer that is fully connected to the sigmoid function as the activation function, and the output vector is the hidden state $\boldsymbol{h}(t) \in R^{d\times 1}$ on timestep t, where $t = 1,2,\ldots,n$.

The update gate $\boldsymbol{z}(t)$ aims to determine how much information from the previous timestep must be carried into the next timestep as follows:

$$\boldsymbol{z}(t) = \sigma\big(W^{\boldsymbol{xz}}\,\mathbf{x}(t) + W^{\boldsymbol{hz}}\boldsymbol{h}(t-1)\big), \tag{8}$$

where $W^{\boldsymbol{xz}} \in R^{d\times n}$ and $W^{\boldsymbol{hz}} \in R^{d\times d}$ are the weight matrices of $\mathbf{x}(t)$ and $\boldsymbol{h}(t-1)$ respectively on the update gate, and $\boldsymbol{\sigma}$ is a sigmoid activation function. When the vector $\boldsymbol{z}(t)$ approaches one, the model will store a significant amount of information in the previous timestep; when the vector $\boldsymbol{z}(t)$ approaches zero, the model will ignore information in the previous timestep. The reset gate $\boldsymbol{r}(t)$ decides how much information from the past must be ignored. The value of $\boldsymbol{r}(t)$ is determined as follows:

$$\boldsymbol{r}(t) = \boldsymbol{\sigma}\left(W^{\boldsymbol{xr}}\mathbf{x}(t) + W^{\boldsymbol{hr}}\boldsymbol{h}(t-1)\right), \tag{9}$$

where $W^{\boldsymbol{xr}} \in R^{d\times n}$ and $W^{hr} \in R^{d\times d}$ are the weight matrices of $\mathbf{x}(t)$ and $\boldsymbol{h}(t-1)$ respectively on the reset gate. The candidate hidden state $\boldsymbol{h}(t)'$ uses the results of the reset gate to store relevant information from the past. The value of $\boldsymbol{h}(t)'$ is determined as follows:

$$\boldsymbol{h}(t)' = tanh(W^{xh}\,\mathbf{x}(t) + (W^{hh}\boldsymbol{h}(t-1)) \odot \boldsymbol{r}(t)), \tag{10}$$

where $W^{\boldsymbol{xr}} \in R^{d\times n}$ and $W^{hr} \in R^{d\times d}$ are the weight matrices of $\mathbf{x}(t)$ and $\boldsymbol{h}(t-1)$ respectively on the candidate hidden state, $\odot$ denotes a Hadamard product [26], and $\tanh$ is the activation function. When $\boldsymbol{r}(t)$ approaches one, $\boldsymbol{h}(t)'$ stores a temporary output value; when $\boldsymbol{r}(t)$ approaches zero, $\boldsymbol{h}(t)$ tends to only accept or process the input. Finally, the hidden state $\boldsymbol{h}(t)'$ stores the output on timestep *t* to pass to the next GRU. This process updates the output value on the new hidden state ($\boldsymbol{h}(t)$) from the previous hidden state ($\boldsymbol{h}(t-1)$) and stores the results in $\boldsymbol{h}(t)'$. The output on timestep *t* is determined by $\boldsymbol{h}(t)$ as follows:

$$\boldsymbol{h}(t) = \boldsymbol{z}(t)\odot\boldsymbol{h}(t-1) + \left(\mathbf{1} - \boldsymbol{z}(t)\right)\odot\boldsymbol{h}(t)'. \tag{11}$$

Every time $\boldsymbol{z}(t)$ approaches one, the information for the previous hidden state $\boldsymbol{h}(t-1)$ is retained. However, when $\boldsymbol{z}(t)$ approaches zero, $\boldsymbol{h}(t)$ will approach $\boldsymbol{h}(t)'$, indicating a significant change in the output value at timestep t. This allows the GRU to overcome long-term dependencies for a time series in the long term, accelerating the computation process in a network [19].

After obtaining output $\boldsymbol{h}(t)$, calculate the loss function on timestep *t* as follows:

$$E(t) = \frac{1}{2}\left(\boldsymbol{y}(t) - \boldsymbol{h}(t)\right)^{\mathbf{2}}. \tag{12}$$

In the equation above, $\boldsymbol{h}(t)$ is the predicted output and $\boldsymbol{y}(t)$ is the actual output. After the feed-forward phase, commence back-propagation to find the gradient error on the weight of the model and then fix the weight. The GRU and LSTM models use the Adam method for weight change optimization; therefore, the discussion of back-propagation for GRU can be found in the explanation of the LSTM model, with the only difference being found in their weight values.

2.3. Long-Short-Term Memory

The LSTM model is derived from a modified RNN model with long-term memory. It was established by [15]. The difference between the LSTM and RNN models is found in the use of a combination of activation functions that determines which information should be stored or discarded. This LSTM model has three gate functions which are denoted by input gate i_t, forget gate f_t, and output gate o_t. Below is a diagram of LSTM architecture and its related mathematical equations [27]:

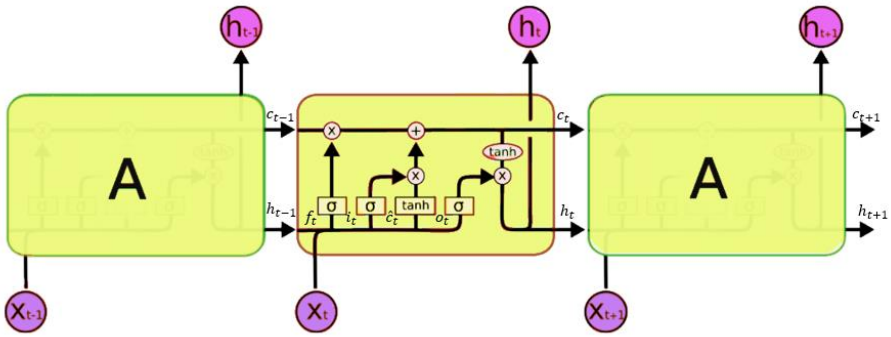

FIGURE 3. LSTM Architecture

(Source: [27] has been reprocessed)

(13a) $$i_t = \sigma(W_i.x_t + U_i.h_{t-1} + b_i),$$

(13b) $$f_t = \sigma\left(W_f.x_t + U_f.h_{t-1} + b_f\right),$$

(13c) $$o_t = \sigma(W_o.x_t + U_o.h_{t-1} + b_o),$$

(13d) $$\tilde{c}_t = tanh(W_{\tilde{c}}.x_t + U_{\tilde{c}}.h_{t-1} + b_{\tilde{c}}),$$

(13e) $$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t,$$

(13f) $$h_t = o_t \odot tanh(c_t).$$

In the series of equations above, i_t is the input gate variable at time *t*, f_t is the forget gate variable at time *t*, o_t is the output gate variable at time *t*, c_t is the cell state variable at time *t*, h_t is the hidden state variable at time *t*, W_i is the weight variable on the input gate, W_f is the weight variable on the forget gate, W_o is the weight variable on the output gate, $W_{\tilde{c}}$ is the weight variable on the cell state, x_t is an input variable, y_t is a target variable, U_i is the hidden state weight variable

on the input gate, U_f is the weight variable of the hidden state on the forget gate, $U_{\tilde{c}}$ is the weight variable of the hidden state on the cell state, b_o is the weight variable bias on the output gate, b_f is the weight variable bias on the forget gate, b_i is the weight variable bias on the input gate, $b_{\tilde{c}}$ is the weight variable bias on the cell state, and U_o is the hidden state weight variable on the output gate [10].

As with GRU, LSTM begins with a feed-forward process and ends with back-propagation. The inputs for a LSTM are $\boldsymbol{x}(t) = [x_1(t)\, x_2(t)\, \cdots\, x_n(t)\,]^T$, $W = \left[W_{\tilde{c}}\, W_i\, W_f\ W_o\,\right]^T$, $U = \left[U_{\tilde{c}}\, U_i\, U_f\ U_o\,\right]^T$, $\boldsymbol{b} = \left[b_{\tilde{c}}\, b_i\, b_f\ b_o\,\right]^T$, and $H(t) = [h_1(t)\, h_2(t)\, \cdots\, h_m(t)\,]^T$. The feed-forward phase counts forward from the first hidden state to the last hidden state by using equations (13a) through (13f) for every timestep *t*. With regard to GRU, this stage aims to obtain the error value from feed-forward at each t^{th} timestep to obtain the total error as shown below.

(14) $$E(total) = E(1) + E(2) + \ldots + E(n).$$

The back-propagation phase aims to obtain a gradient change for every weight in the LSTM model for use in updating the weight in the LSTM model. This model has 12 weights ($W_i, W_f, W_o, W_{\tilde{c}}, U_i, U_f, U_o, U_{\tilde{c}}, b_i, b_f, b_o, b_{\tilde{c}}$) from which the error change gradient must be determined, whereas the GRU model requires six error change gradients ($W^{\boldsymbol{xz}}, W^{\boldsymbol{hz}}, W^{\boldsymbol{xh}}, W^{hh}, W^{\boldsymbol{xr}}, W^{hr}$). These gradients are obtained from a partial derivative of $E(t)$ on each weight for every value of *t = 1,2, …n*.. Refer to [11] for the rules of differential chains for calculating the gradients of error functions.

As with GRU, the Adam optimization method for weight changes will be used. Adam is a stochastic optimization method that conceals the drawbacks of the Stochastic Gradient Descent (SGD) optimization method, that has drawbacks in adaptive learning and momentum. The equations of the Adam optimization are as follows: [25].

(15a) $$\theta_t = \theta_{t-1} - \alpha . \frac{\widehat{m_t}}{\sqrt{\widehat{V_t} + \epsilon}},$$

(15b) $$g_t = \nabla_\theta f_t(\theta_{t-1}),$$

(15c) $$\widehat{m_t} = \frac{m_t}{(1-\beta_1^t)},$$

(15d) $$\widehat{v_t} = \frac{v_t}{(1-\beta_2^t)},$$

(15e) $$m_t = \beta_1 . m_{t-1} + (1-\beta_1).g_t,$$

(15f) $$v_t = \beta_2 . v_{t-1} + (1-\beta_2).g_t^2.$$

Here, m_t is the second bias variable, v_t is the first bias variable, $\widehat{v_t}$ is the fixed first bias variable, $\widehat{m_t}$ is the fixed second bias variable, g_t is the loss function gradient on θ_{t-1}, θ_t is the weight on the t^{th} epoch, $f(\theta)$ is a stochastic function with parameter θ, β_1 is the first momentum variable, α is the learning rate variable, and β_2 is the second momentum variable. In this research, the parameter values used are the same as those proposed by [25] for machine learning problems, that is, $\alpha = 0{,}001, \beta_1 = 0{,}9,\ \beta_2 = 0{,}999,$ and $\epsilon = 10^{-8}$. β_1^t, and β_2^t are the notation for β_1 and β_2 respectively to the power of t, with the initial parameter values of $m_0 = 0$ and $v_0 = 0$.

Suppose the gradients of the 12 weights above were obtained. Then, continue the process using Adam to obtain new weights that minimize the error. For example, the weight gradient obtained is on weight W_o, that is, δW_o; according to the Adam method, it is calculated as follows:

(16a) $$m_1 = \beta_1 . m_0 + (1-\beta_1).\delta W_o,$$

(16b) $$v_1 = \beta_2 . v_0 + (1-\beta_2).(\delta W_o)^2,$$

(16c) $$\widehat{m_1} = \frac{m_1}{(1-\beta_1^1)},$$

(16d) $$\widehat{v_1} = \frac{v_1}{(1-\beta_2^1)},$$

(16e) $$W_{o_{NEW}} = W_o - \alpha . \frac{\widehat{m_1}}{\sqrt{\widehat{v_1}+\epsilon}}.$$

Repeat this process for the remaining 11 weights. The feed-forward and back-propagation processes must be repeated until they reach the desired level of accuracy. For GRU and LSTM, the tanh activation function is used and is written as follows:

(17) $$tanh\ (z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}, \qquad z \in R, \qquad -1 < h(z) < 1.$$

Both methods also use the sigmoid activation function as follows:

$$\Phi(z) = \frac{1}{1+e^{-z}} = \frac{e^{z}}{e^{z}+1}, \qquad z \in R, \qquad 0 < \Phi(z) < 1. \tag{18}$$

For GRU and LSTM, the loss function uses RMSE as stated in equation (6).

3. RESULTS AND DISCUSSION

The data used is DHF incidents data from DKI Jakarta, excluding Kepulauan Seribu Regency. Moreover, daily weather data was utilized: specifically, the total rainfall, average relative humidity, and average air temperature from weather stations in Cengkareng (Jakarta Barat), Kemayoran (Jakarta Pusat), Pondok Betung (Jakarta Selatan), Halim (Jakarta Timur), and Tanjung Priok (Jakarta Utara). The data is taken from daily weather samples from January 2009 to September 2017. DHF incidents data is measured from the number of infected patients [28].

The implementation process begins with pre-processing of DHF incidents and weather data. It consists of mean imputation, conversion of daily data to weekly data, development of input pattern and target data, and data normalization. Mean imputation on weather data is only necessary due to the presence of empty or unusual (outlier) data. The outliers in question have a value of -999 or 8888 and are replaced with means from other data that are neither empty nor outliers. Then, the results of the mean imputation are converted from daily data to weekly data to reduce complex patterns in daily data, thereby improving the model's performance in making predictions [29]. DHF incidents and rainfall data conversion involves the addition of data for seven days; converting average air temperature and humidity data involves the calculation of a seven-day average for the said data. The data from the results of imputation and conversion consists of 455 weekly observations. This data is then arranged in the form of input patterns and target vectors.

Each input pattern, as a predictor variable, consists of a DHF incidents value, a rainfall value, an average air temperature value, and an average humidity value, while each target vector is a DHF incidents value. Time lag analysis was performed to determine the cross-correlation value between the number of weekly DHF incidents and predictor variables in the range of one to eight weeks

earlier based on the mosquito life cycle [30]. The time lag between predictor variables with weekly case totals is determined from the highest correlation value as performed by [31]. The linear correlation of two types of data with N observation values, such as $\boldsymbol{w} = [w(0), w(1), \dots, w(N-1)]^T$ and $\boldsymbol{h} = [h(0), h(1), \dots, h(N-1)]^T$, is measured using linear correlation coefficients as follows [31]:

$$\boldsymbol{\rho} = \frac{\frac{1}{N}\sum_{i=0}^{N-1}(w(i)-\bar{w})\cdot\left(h(i)-\bar{h}\right)}{\sqrt{\left(\frac{1}{N}\sum_{i=0}^{N-1}(w(i)-\bar{w})^2\right)\cdot\left(\frac{1}{N}\sum_{i=0}^{N-1}\left(h(i)-\bar{h}\right)^2\right)}}, \tag{19}$$

where ρ is the linear correlation coefficient, $\bar{w} = \frac{1}{N}\sum_{i=0}^{N-1} w(i)$ and $\bar{h} = \frac{1}{N}\sum_{i=0}^{N-1} h(i)$. Based on the results of the correlation coefficient calculations, lag is chosen based on the highest absolute value as shown in Table 1 below.

TABLE 1. Results of Lag Selection based on Linear Correlation

Lag Value of Cumulative DHF incidents with Predictor Variables				
Regency	**Avg. Temperature**	**Avg. Rainfall**	**Avg. Humidity**	**Weekly Dengue Cases**
East Jakarta	8 weeks	7 weeks	8 weeks	1 week
West Jakarta	8 weeks	7 weeks	7 weeks	1 week
Central Jakarta	8 weeks	8 weeks	7 weeks	1 week
South Jakarta	5 weeks	7 weeks	6 weeks	1 week
North Jakarta	1 week	6 weeks	8 weeks	2 weeks

Normalization aims to place weather and case data in similar value intervals. If the value interval used has a sufficiently large difference, the learning process becomes overly sensitive to at least one data type and is unable to yield a good model [11]. The normalization function used is the z-score function which is as follows:

$$f_{normalization}(x_i) = \frac{x_i - \mu}{\sigma}, \tag{20}$$

where $f_{normalization}(x_i)$ denotes the z-score normalization function for a value x_i,

$\mu = \frac{1}{T}\sum_{i=1}^{T} x_i$, $\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{T}(x_i - \mu)^2}$, and T denotes the amount of training data [32]. The opposite of normalization is called denormalization.

Simulations were conducted in each region using three methods (ENN, GRU, and LSTM) with the number of epochs set at 300 and 400, each using the following training/testing data compositions: 70/30, 80/20, and 90/10. This was done to obtain the RMSE values for training and testing data as well as the denormalized DHF case data. RMSE for denormalized data is obtained for seeing the prediction results from all three methods on actual DHF case data. For the ENN method, the number of neurons in the hidden layer tested are 4, 8, 16, 32, and 64, and the learning rate is 0.001. For the LSTM method, the number of units is 400 and 450, learning rate is 0.001, drop-out parameters are 0.0 and 0.4, batch sizes are 32 and 64, and recurrent dropout parameters are 0.0 and 0.4. For the GRU method, the number of units is 300 and 350, learning rate is 0.001, drop-out parameters are 0.0 and 0.3, batch sizes are 32 and 64, and recurrent dropout rates are 0.0 and 0.4. The results of the simulation based on the best RMSE of all experiments are obtained using a grid search as shown in Tables 1 to 6. The simulation results with 300 epochs on all three data compositions are shown in Tables 2 to 4 as follows:

TABLE 2. RMSE for Each District with 300 Epochs and Data Composition of 70/30

Method	RMSE	District				
		West Jakarta	South Jakarta	East Jakarta	North Jakarta	Central Jakarta
ENN	Training	0.12062	0.12577	0.14100	0.13818	0.12904
	Testing	0.22722	0.29351	0.41531	0.30613	0.21945
	Denormalization	18.63241	17.02371	12.87465	8.57163	9.43615
LSTM	Training	0.08301807	0.053544	0.0429068	0.063513	0.0906752
	Testing	0.13725576	0.0640291	0.133380044	0.0954041	0.1297555
	Denormalization	17.568777	8.131705	13.33800444	5.0564175	8.4341123
GRU	Training	0.07720	0.04615	0.04347	0.06220	0.09549
	Testing	0.13508	0.12620	0.13382	0.17249	0.01685
	Denormalization	17.29084	16.02866	13.38213	8.99221	8.43778

TABLE 3. RMSE for Each District with 300 Epochs and Data Composition of 80/20

Method	RMSE	District				
		West Jakarta	South Jakarta	East Jakarta	North Jakarta	Central Jakarta
ENN	Training	0.12030	0.12817	0.14336	0.13711	0.12925
	Testing	0.26549	0.35603	0.48413	0.36957	0.24927
	Denormalization	21.77019	20.64969	15.49225	10.34791	10.71866
LSTM	Training	0.083018	0.053684	0.043906	0.063678	0.0834206
	Testing	0.157863	0.076464	0.156622	0.114218	0.146769
	Denormalization	20.20648	9.7109294	15.66229	6.053557	9.5400300
GRU	Training	0.07321	0.06115	0.04743	0.08234	0.09555
	Testing	0.18958	0.16032	0.16795	0.20974	0.10386
	Denormalization	24.26675	20.36085	16.79555	11.23732	6.75131

TABLE 4 RMSE for Each District with 300 Epochs and Data Composition of 90/10

Method	RMSE	District				
		West Jakarta	South Jakarta	East Jakarta	North Jakarta	Central Jakarta
ENN	Training	0.10214	0.15476	0.07021	0.11326	0.10670
	Testing	0.07399	0.07349	0.08288	0.10172	0.06983
	Denormalization	9.47020	9.33289	8.28839	5.39110	4.53894
LSTM	Training	0.11498	0.07383	0.06500	0.07437	0.118713
	Testing	0.07910	0.04427	0.09570	0.10073	0.07369
	Denormalization	10.12496	5.62297	9.57013	5.33869	4.78626
GRU	Training	0.07325	0.07700	0.07750	0.07791	0.11135
	Testing	0.09209	0.08614	0.07800	0.10374	0.07160
	Denormalization	11.78858	10.94026	7.65996	5.49867	4.64978

With 300 epochs, the RMSE on testing data with a composition of 90/10 for all three methods is less for every regency than that obtained using the 70/30 or 80/20 distributions of testing data. In other words, only two RMSEs were larger: those of North Jakarta using LSTM (0.10073) and of Central Jakarta using GRU (0.07160). The smallest RMSE obtained using both methods is for a 70/30 data composition. Therefore, in general, based on the data used and the RMSE values obtained, the simulation results of all three methods with 300 epochs and a 90/10 data composition yield the best prediction of DHF cases.

The results of the simulation using all three methods with 400 epochs with three data compositions are shown in Tables 5 to 7 below.

TABLE 5. RMSE for Each District with 400 Epochs and Data Composition of 70/30

Method	RMSE	District				
		West Jakarta	South Jakarta	East Jakarta	North Jakarta	Central Jakarta
ENN	Training	0.11985	0.12379	0.13739	0.13600	0.12795
	Testing	0.22586	0.29657	0.42508	0.31161	0.21604
	Denormalization	18.52033	17.20094	13.17738	8.72498	9.28979
LSTM	Training	0.0830180	0.05354437	0.042906	0.063513	0.0909285
	Testing	0.134069	0.0689192	0.143492	0.124974	0.131562
	Denormalization	17.4677865	13.57540255	14.34929	6.623649	8.5515720
GRU	Training	0.07720	0.04604	0.04370	0.06220	0.07905
	Testing	0.13452	0.12604	0.13463	0.17360	0.12839
	Denormalization	17.21947	16.00729	13.46384	9.20116	8.34573

TABLE 6. RMSE for Each District with 400 Epochs and Data Composition of 80/20

Method	RMSE	District				
		West Jakarta	South Jakarta	East Jakarta	North Jakarta	Central Jakarta
ENN	Training	0.11952	0.12628	0.14035	0.13516	0.13277
	Testing	0.26374	0.36189	0.49568	0.37687	0.24361
	Denormalization	21.62647	20.98975	15.86181	10.55234	10.47504
LSTM	Training	0.083096	0.053544	0.042930	0.063513	0.083450
	Testing	0.155666	0.116609	0.182761	0.144559	0.144300
	Denormalization	19.925341	14.80937	18.27261	7.661677	9.739536
GRU	Training	0.06204	0.06115	0.04722	0.08330	0.09659
	Testing	0.15551	0.16095	0.16820	0.20533	0.11440
	Denormalization	19.90546	20.44134	17.92285	10.88262	7.43662

TABLE 7. RMSE for Each District with 400 Epochs and Data Composition of 90/10

Method	RMSE	District				
		West Jakarta	South Jakarta	East Jakarta	North Jakarta	Central Jakarta
ENN	Training	0.10091	0.15069	0.06890	0.10714	0.10466
	Testing	0.07440	0.07059	0.08164	0.09939	0.07062
	Denormalization	9.52313	8.96552	8.16379	5.26742	4.59050
LSTM	Training	0.11462	0.07420	0.06523	0.07439	0.11930
	Testing	0.08288	0.05344	0.08830	0.12538	0.07061
	Denormalization	10.60690	6.78778	8.83084	6.64546	4.59004
GRU	Training	0,10014	0.00590	0.07681	0.09818	0.07943
	Testing	0.08464	0.06480	0.07996	0.10388	0.07160
	Denormalization	10.83509	8.23014	7.99608	5.73116	4.65422

With 400 epochs, the RMSEs for testing data with a composition of 90/10 for all three methods were also less than those obtained through data testing with a 70/30 or 80/20 in nearly every district. The only exception was that of testing data for North Jakarta using LSTM (0.12538) with a 70/30 data distribution.

Based on the data used and the RMSE of testing data from Tables 1 to 6, it can generally be stated that all three methods with a 90/10 data composition yield a better DHF case prediction than with a 70/30 or 80/20 data composition.

Through the comparison of the simulation results with two epochs, it can be said that changing the epoch from 300 to 400 has little effect on the smallest RMSE for ENN, which always has a 90/10 data composition. However, it affects the smallest RMSE of the LSTM and GRU methods in one or two regions. Moreover, nearly every RMSE on the denormalized data is smallest for a 90/10 data distribution with 300 or 400 epochs, except for that which was obtained using LSTM in North Jakarta (5.0564175), which is smallest for a 70/30 data distribution with 300 epochs. Generally, smaller RMSE values for a 90/10 data distribution for denormalized data are made possible by all three methods having learned enough from sufficiently fluctuating training data. This result shows that, in general, all three methods can predict DHF incidents data best with a 90/10 data distribution. Therefore, the simulation below only includes this particular distribution.

To explain the differences between the simulation results from all three methods, Figure 4 shows the simulation results from data and all three methods with the same data distribution on

denormalized DHF case data for all districts. Since no significant differences are expected between the results of RMSE testing with 300 and 400 epochs, the results with 300 epochs are chosen for the sake of efficiency in each region in Figure 4 as follows:

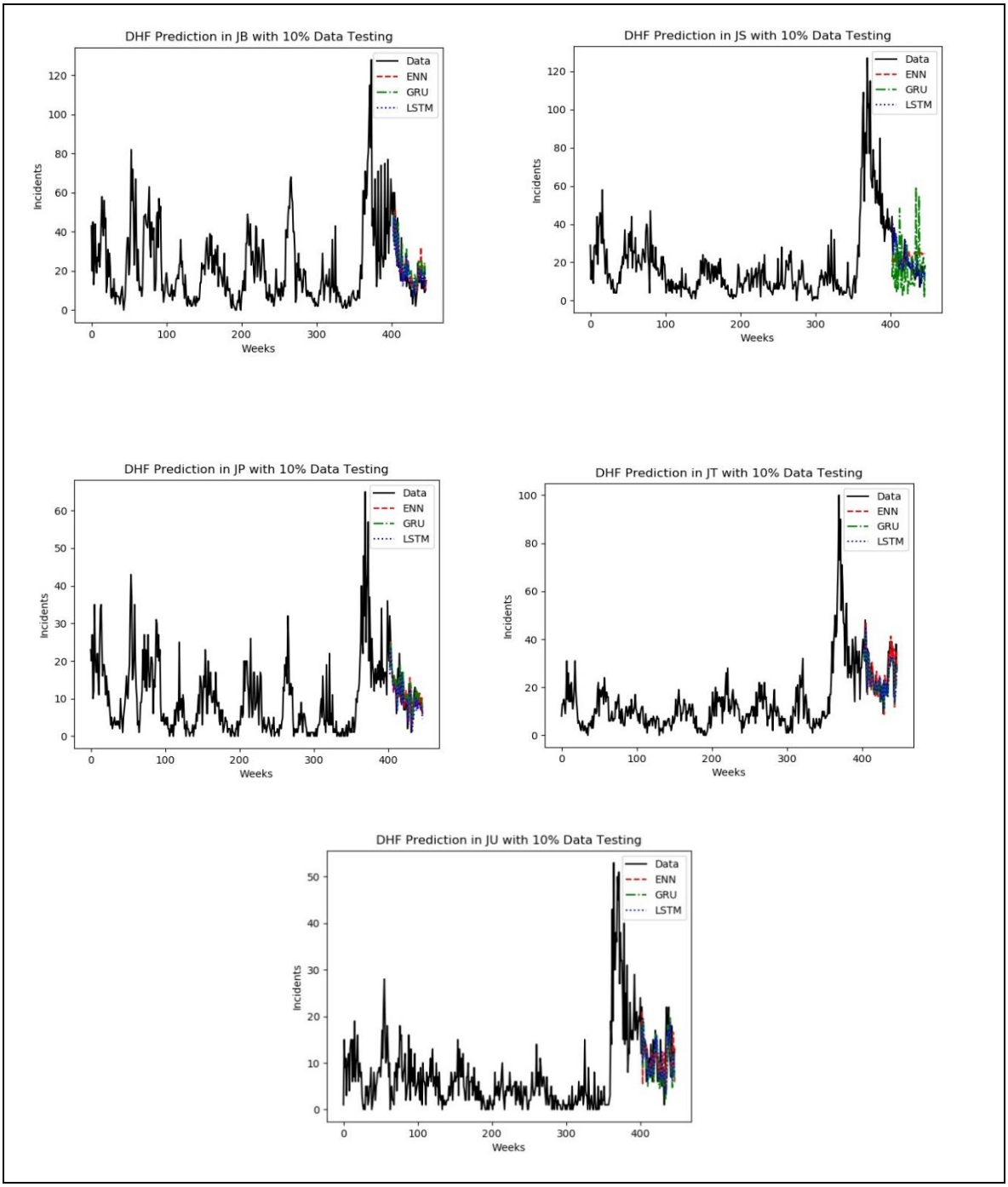

FIGURE 4. The Simulation Results from Data and all Three Methods with a 90/10 Data Distribution on Denormalized DHF Case Data for all Districts.

Figure 4 shows the simulation results from all three methods on testing data and DHF case data over the same period, where the DHF case data, ENN case data, GRU simulation results, and LSTM simulation results are marked in black, red, green, and blue respectively. The results of all these simulations are particularly good on training data, although the simulation data for GRU in South Jakarta strongly fluctuates on data testing. This is consistent with the results of the denormalized RMSE from GRU in South Jakarta, that has the highest value (10.94026) compared to denormalized RMSEs from ENN (9.33289) and LSTM (5.62297).

Based on the simulation results, all three methods with a 90/10 data distribution can produce a sufficiently accurate simulation that can follow actual data movements, except for GRU in South Jakarta. With regard to the actual DHF case data, the highest number of cases occurs between the 360^{th} and 380^{th} weeks; this sudden spike in cases is difficult for all three models to predict accurately. The information about this spike in cases can be included in training data with a 90/10 data distribution. For a more detailed view of the simulation results for all three methods on testing data, Figure 5. shows the simulation results for all three methods on testing data with 300 epochs and with a 90/10 data distribution across all regencies.

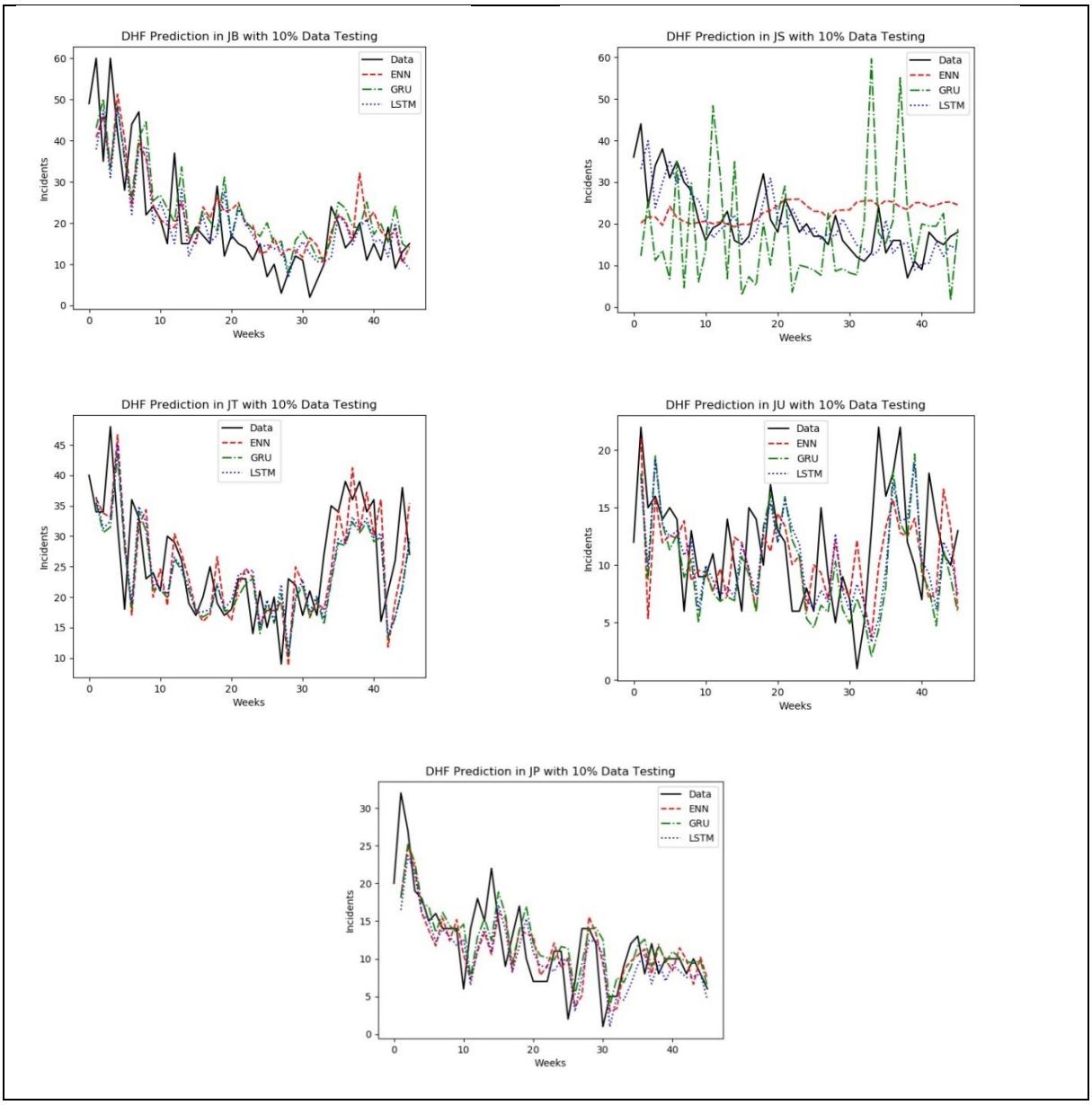

FIGURE 5. The Simulation Results for All Three Methods on Testing Data with 300 Epochs and with a 90/10 Data Distribution across All Districts

For all graphs in Figure 5, the DHF incidents data graph begins from time $t - 1$, while the graph for the simulation results for all three methods begins from time t on testing data. The DHF case data, ENN case data, GRU simulation results, and LSTM simulation results are marked in black, red, green, and blue respectively. An interesting phenomenon occurs in the graph for South Jakarta, where the LSTM method with RMSE of 0.04427 can predict DHF case data better than the other two methods. This is shown in the graph for the results of the simulation using LSTM on testing

data, which follows DHF case data more closely than the graphs for the results of the simulation using the other two methods, where the RMSEs on testing data from the ENN and GRU methods are 0.07349 and 0.08614 respectively. The results of the simulation on the testing data in Central and West Jakarta show that all three methods can predict the data reasonably well. However, for the simulation results on testing data in East Jakarta, all three methods struggled somewhat in predicting a sudden spike in cases near the end of the data. Testing data for East Jakarta yielded RMSEs for ENN, LSTM, and GRU of 0.08288, 0.09570, and 0.07800 respectively. Moreover, the results of the simulation for all three methods on testing data for North Jakarta fluctuate more than those for testing data in all other regions, and all three methods struggle to predict the strongly fluctuating case data in North Jakarta. This is borne out by the RMSEs for testing data in North Jakarta, where the RMSEs for ENN, LSTM, and GRU are 0.10172, 0.10073, and 0.10374 respectively. These values are greater than those obtained using the same methods for testing data in all other regions. Based on the results of RMSE values and the simulation on the testing data, the LSTM method can better predict DHF incident data in almost every district in DKI Jakarta.

4. CONCLUSION

This research used DHF incidents data in DKI Jakarta (excluding Kepulauan Seribu Regency) and weather data consisting of temperature, rainfall, and humidity for predicting DHF incidents using ENN, LSTM, and GRU. Based on the data used and the value of the RMSE, a data composition with 90% training data and 10% testing data yields a superior prediction than a data composition of either 80% training data and 20% testing data or 70% training data and 30% testing data. The RMSE value for all three methods is also smallest for a data composition with 90% training data and 10% testing data on denormalized data since all three methods have learned sufficiently from training data that has already included certain significant fluctuations. Based on the results of the DHF case simulation, all three methods with a data composition with 90% training data and 10% testing data tended to simulate the growth of DHF cases accurately, except for GRU in South Jakarta. Based on the results of the data testing simulation, it can be noted that

LSTM best predicts DHF case data when it is either slightly or strongly fluctuating compared to the other two methods. Hence, based on the DHF case and the weather data used, the simulation results show that LSTM is better suited to predicting DHF case data than ENN or GRU.

ACKNOWLEDGMENTS

This research was funded by Universitas Indonesia, PUTI grant scheme No.: NKB-1983/UN2.RST/HKP.05.00/2020. We thank Indonesia's Meteorology, Climatology, and Geophysical Agency and the Jakarta Health Department for their data sets. This research would not have been possible without their generous cooperation.

CONFLICT OF INTERESTS

The authors declare that there is no conflict of interests.

REFERENCES

[1] World Health Organization. Dengue: Guidelines for Diagnosis, Prevention, Treatment and Control, Geneva, Switzerland: World Health Organization. (2009). Retrieved from https://apps.who.int/iris/handle/10665/44188. Accessed March 19, 2020.

[2] CDC. Dengue. Retrieved from https://www.cdc.gov/dengue/index.html. Accessed February 14, 2020.

[3] The Indonesia Ministry of Health Department. Infodatin Situasi Penyakit Demam Berdarah di Indonesia tahun 2017. (2018). Retrieved from https://pusdatin.kemkes.go.id/resources/ download/pusdatin/infodatin/InfoDatin-Situasi-Demam-Berdarah-Dengue.pdf. Accessed March 19, 2020.

[4] Kompas.tv. Waspada! Beberapa wilayah masuk zona merah Demam Berdarah, termasuk Jakarta. (2020). Retrieved from https://www.kompas.tv/article/70622/waspada-beberapa-wilayah-masuk-zona-merah-demam-berdarah-termasuk-jakarta. Accessed June 22, 2020.

[5] J. Xu, K. Xu, Z. Li, F. Meng, T. Tu, L. Xu, Q. Liu. Forecast of dengue cases in 20 Chinese cities based on the deep learning method. Int. J. Environ. Res. Public Health, 17 (2) (2020), 453.

[6] A. L. Ramadona, L. Lazuardi, Y. L. Hii, Å. Holmner, H. Kusnanto, J. Rocklöv, Prediction of dengue outbreaks based on disease surveillance and meteorological data, PLoS ONE. 11 (2016), e0152688.

[7] R. L. Riswanto, Sutikno, Indriyati. Aplikasi prediksi jumlah penderita penyakit Demam Berdarah Dengue di Kota Semarang menggunakan Jaringan Syaraf Tiruan backpropagation. Jurnal Masyarakat Informatika, 5 (10) (2014), 19-27.

[8] A. S. Ichwani, H. A. Wibawa. Prediksi angka kejadian Demam Berdarah Dengue (DBD) berdasarkan faktor cuaca menggunakan metode Extreme Learning Machine (Studi Kasus Kecamatan Tembalang). Jurnal IPTEK, 23 (1) (2019), 31-24.

[9] K. P. Murphy. Machine learning: a probabilistic perspective. MIT Press, Cambridge, Mass. (2012).

[10] G. Petneházi, Recurrent neural networks for time series forecasting, ArXiv:1901.00069 [Cs, Stat]. (2018).

[11] C. C. Aggarwal, Neural networks and deep learning: a textbook, Springer, Cham, 2018.

[12] S. A. Abdulkarim. Time series prediction with simple recurrent neural networks. Bayero. J. Pure Appl. Sci. 9 (1) (2016), 19-24.

[13] M. Madhiarasan, S. N. Deepa. Elman neural network with modified grey wolf optimizer for enhanced wind speed forecasting. Circuits Syst. 7 (2016), 2975-2995.

[14] S. Ghosh, B. Tudu, N. Bhattacharyya, R. Bandyopadhyay. A recurrent Elman neural network in conjunction with an electronic nose for fast prediction of optimum fermentation time of black tea. Neural Comput. Appl. 31 (2017), 1165-1171.

[15] S. Hochreiter, J. Schmidhuber. Long short-term memory. Neural Comput. 9 (1997), 1735-1780.

[16] A. Graves, Generating sequences with recurrent neural networks, ArXiv:1308.0850 [Cs]. (2013).

[17] L. Liu, M. Han, Y. Zhou, Y. Wang, LSTM recurrent neural networks for influenza trends prediction, in: F. Zhang, Z. Cai, P. Skums, S. Zhang (Eds.), Bioinformatics Research and Applications, Springer International Publishing, Cham, 2018: pp. 259–264.

[18] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, ArXiv:1406.1078 [Cs, Stat]. (2014).

[19] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Gated feedback recurrent neural networks, ArXiv:1502.02367 [Cs, Stat]. (2015).

[20] Epidemiology Surveillance Section of the Health Department of DKI Jakarta. (2019). Retrieved from https://surveilansdinkesdki.net/chart.php. Accessed May 10, 2019.

[21] J. L. Elman. Finding structure in time. Cognitive Sci. 14 (1990), 179-211.

[22] Y. Zhang, X. Wang, H. Tang. An improved Elman neural network with piecewise weighted gradient for time series prediction. Neurocomputing. 359 (2019), 199-208.

[23] G. Zaccone, M. R. Karim. Deep learning with tensorflow. Second edition, Packt Publishing, Birmingham, (2018).

[24] A. Zhang, Z. C. Lipton, M. Li, A. J. Smola. Dive into deep learning. (2020). Retrieved from https://d2l.ai.

[25] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, ArXiv:1412.6980 [Cs]. (2017).

[26] E. Million. The Hadamard Product. (2007). Retrieved from http://buzzard.ups.edu/courses/2007spring/projects/million-paper.pdf. Accessed February 14, 2020.

[27] J. D. Seo. Only numpy: deriving forward feed and backpropagation in long short term memory. (2018). Retrieved from https://towardsdatascience.com/only-numpy-deriving-forward-feed-and-back-propagation-in-long-short-term-memory-lstm-part-1-4ee82c14a652. Accessed June 15, 2020.

[28] E. Stellwagen, S. Darin. Forecasting weekly and daily data: practical strategies for better results. Business Forecast System, Belmont. (2019).

[29] CDC. Dengue (2020). Retrieved from https://www.cdc.gov/dengue/index.html. Accessed February 14, 2020.

[30] V. Ughelli, Y. Lisnichuk, J. Paciello, J. Pane, Prediction of dengue cases in paraguay using artificial neural networks, in 3rd International Conference on Health Informatics Medical Systems (2017), pp. 130–136.

[31] A.F. Kohn, Autocorrelation and cross-correlation methods, in: M. Akay (Ed.), Wiley Encyclopedia of Biomedical Engineering, John Wiley & Sons, Inc., Hoboken, NJ, USA, 2006: p. ebs0094.

[32] Z. Mustaffa., Y. Yusof. A comparison of normalization techniques in predicting dengue outbreak. In: Proceeding of international conference on business and economics research, IACSIT Press, Kuala Lumpur, Malaysia. (2010).