Department of Digital Business
Journal of Artificial Intelligence and Digital Business (RIGGS)
Homepage: https://journal.ilmudata.co.id/index.php/RIGGS
Vol. 4 No. 2 (2025) pp: 274-284
P-ISSN: 2963-9298, e-ISSN: 2963-914X

Enhanced Rainfall Forecasting Through Deep Learning Optimization Using Long Short-Term Memory Networks

Ade May Luky Harefa[1], Robin Antoni[2], Andri Ismail Sitepu[3], Yohannes France Limbong[4], Muhammad Syahputra Novelan[5]
[1,2,3,4,5]Master of Information Technology, Pembangunan Panca Budi University
[1]ademayluky@gmail.com*, robinantonisianipar1979@gmail.com[2], adrismstp@gmail.com[3], yohannesfr4anc3@gmail.com[4], putranovelan@dosen.pancabudi.ac.id[5]

Abstract

This study aims to develop a rainfall prediction system using Deep Learning with the Long Short-Term Memory (LSTM) method to improve prediction accuracy and efficiency. The model was built using rainfall data from Gunung Sitoli, covering the period from October 16 to December 14, 2004. The dataset was divided into 90% for training and 10% for testing. The LSTM model was configured with 1 hidden layer and trained for 50 epochs. To evaluate its performance, the Mean Squared Error (MSE) metric was applied. The model achieved an MSE of 0.03 on the test data, indicating a low prediction error and good accuracy. This result shows that LSTM is capable of learning rainfall patterns over time and producing reliable forecasts. Furthermore, the model was integrated into a system to streamline the forecasting and evaluation process. This integration provides an efficient alternative to manual calculations, offering users faster and more accessible predictions. The implementation of this system is especially beneficial for early warning and decision-making processes in regions like Gunung Sitoli, where rainfall patterns can significantly impact on daily activities and disaster preparedness.

Keywords: Rainfall; Deep Learning; LSTM; Prediction; MSE.

1. Introduction

In today's digital era, data has become a strategic asset in supporting fast and precise decision-making [1]. The concept of Data Driven Decision Making (DDDM) emerged in response to this need, where data is not only a consideration, but also the main basis for determining policies or actions [2]. The role of technology is vital in supporting the processing and analysis of large amounts of data, including in the meteorological sector which requires accurate and sustainable modelling of rainfall predictions. Unfortunately, conventional methods still face limitations, such as the large number of parameters required, complex mathematical assumptions, and inflexible equation structures.

As a solution to the limitations of conventional prediction models, Artificial Intelligence-based approaches are beginning to be widely applied [3]. AI has the ability to learn from historical data and infer important patterns through machine learning and artificial neural network approaches [4]. One of the fastest-growing AI approaches is Deep Learning, which allows learning from multi-layered data, resembling how the human brain works in understanding complex information [5].

The use of Deep Learning in the form of Recurrent Neural Network (RNN) has shown promising results in handling sequential data or time series [6]. Nevertheless, RNNs have limitations in remembering long-term information due to vanishing gradient issues. To overcome these weaknesses, a Long Short-Term Memory (LSTM) architecture was developed that is able to store important information for longer with the help of memory cells and gate mechanisms, making it more effective in modeling long temporal dependencies on time series data such as rainfall [7].

Previous studies have proven the advantages of LSTMs in a variety of applications, including market price prediction, weather analysis, and medical data processing. In the context of meteorology, LSTM has been proven to be able to predict climate parameters such as air temperature, humidity, pressure, and precipitation more accurately than other methods. The ability of LSTMs to handle sequential inputs and generalize historical data patterns makes them an ideal choice for processing time-series climatological data.

Departing from these conditions, this research was conducted with the aim of optimizing the Deep Learning method using LSTM architecture to project rainfall in the Mount Sitoli area. Given the importance of weather prediction information in various sectors such as agriculture, transportation, and disaster mitigation, a reliable LSTM-based prediction system is expected to be an alternative in supporting better decision-making and responsiveness to weather dynamics.

Using historical rainfall data from the Gunung Sitoli Climatology Station from 1985 to 2021, this study not only focuses on the prediction process alone, but also on the development of predictive systems that can be implemented directly. This research is expected to be able to produce predictive models that have high accuracy and can be used in real scenarios to reduce the risk of increasingly complex climate uncertainty.

2. Research Methods

Process analysis is carried out to find out how to solve the problem so that it can produce solutions using the right methods, through a cooperative and interactive process starting from analyzing the problem, identifying the problem, the final result of observations in various representation formats, to checking the Interpolate NaN for the accuracy of the understanding obtained. The following is a flowchart of the Process Analysis steps

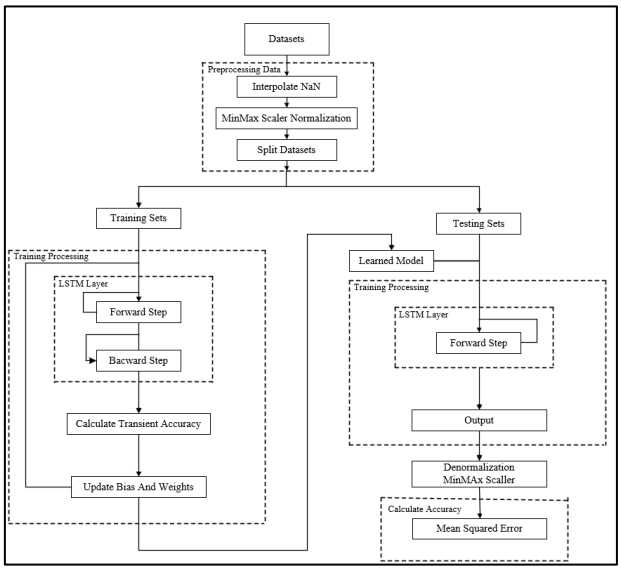

Figure 1. Flowchart of Process Analysis Steps

LSTM memiliki tiga di antaranya gerbang, untuk melindungi dan mengontrol cell state, Struktur gerbangnya mencakup forget gate, input gate. dan output gate. Gate terdiri dari layer jaring saraf sigmoid dan operasi perkalian pointwise [7].

Forget Gate

On the forget gate, the information on each input data will be processed and which data will be stored or discarded on memory cells. The activation function used in this forget gate is the sigmoid activation function. Where the output is between 0 and 1. If the output is 1 then all data will be stored and vice versa if the output is 0 then all data will be discarded [8].

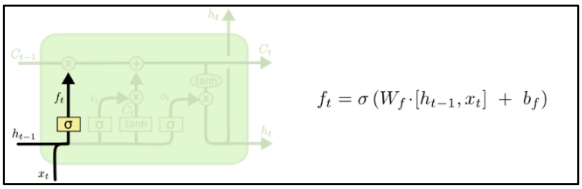

Figure 2. Forget Gate Equation in LSTM

DOI: https://doi.org/10.31004/riggs.v4i2.487

Input Gate

At the input gate there are two gates that will be implemented, first it will be decided which value will be updated using the sigmoid activation function. Next, the tanh activation function will create a new value vector that will be stored in the memory cell [9].

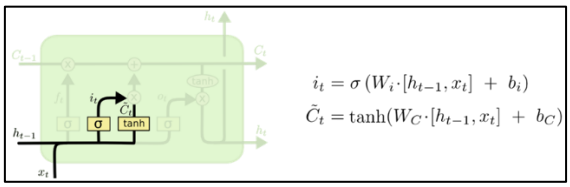

Figure 3. Equation passing through the Input Gate in LSTM

In the next step, the LSTM decides what information will be stored from the cell state. First, a sigmoid layer called an "input gate layer" (i_t) decides which values to update. After that, the tanh layer creates a new candidate value vector, C_t, which can be added to the state.

Cell state / Memory State

In the cell state gates, the value of the previous memory cell will be replaced with the new memory cell value. Where this value is obtained by combining the values found in the forget gate and the input gate [10].

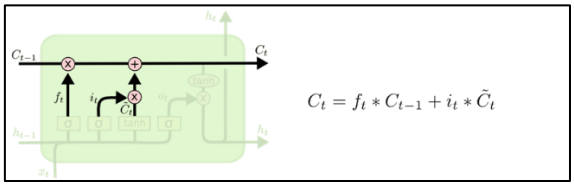

Figure 4. Equation for updating cell state in LSTM

In the next step, the two merged layers are combined to make an update to the cell state. In this step, the value of the old cell state (C_(t-1)), will be updated to the value of the new cell state (C_t) where the LSTM will multiply the old cell state by (f_t) and then add it to (i_t*C̃_t). This is the new candidate value, which is scaled based on how much it decides to update each cell state value.

Output Gate

At the output gate there are two gates that will be implemented, the first will be decided which value on the memory cell will be ejected using the sigmoid activation function. Next, the value will be placed on the memory cell using the tanh activation function. Finally, the two *gates* are multiplied so that the value to be issued is produced.

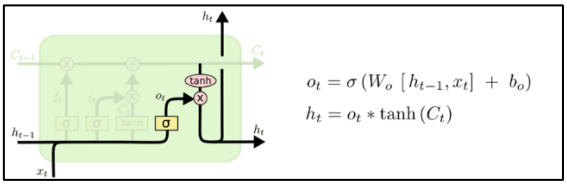

Figure 5. Equation passing through the Output Gate on LSTM

The last is the stage where it is necessary to decide what to produce. The output will be based on the cell state, but it will be a filtered version. First, the researcher runs a sigmoid layer that decides which part of the cell state the researcher will produce. Then, the cell state is placed through the tanh (to push the value to between -1 and 1) and multiplied by the output of the sigmoid gate layer, so that it will only display the broken part.

3. Results and Discussions

Proses Training Model

At the training stage, the LSTM model is carried out by backpropagation according to the data used. Here are some of the stages in the LSTM model training process with backpropagation as follows:

DOI: https://doi.org/10.31004/riggs.v4i2.487

1. Initialize the values of epoch, batch, timestep/sequence, features, dropout, units, and Learning rate which will later be the required hyperparameter values such as the initial weight value, hidden layer (hidden layer) are automatically determined by the system.
2. The training data input is in the form of a 3D tensor.
3. Conducting LSTM model training on each input, starting with forget gates, input gates, cell states and finally output gates.
4. MSE calculation to obtain the value of the difference between the value of the LSTM network output and the actual output.
5. Gradient calculation to determine the value of weight and bias so that the loss/error result is close to 0 using the backpropagation algorithm.
6. After obtaining the gradient value, it is continued with the equation of the Stochastic Gradient Descent (SGD) optimization function and updating the weight and bias values.
7. If the entire batch of data has been completed in the iteration, go back to Step 2 up to the predetermined number of epochs.

Based on the above training steps, the formation of the LSTM model begins with the model formed with data trained by passing the Gates mechanism on the LSTM. The data is continuously trained until it reaches the error limit according to the desired number of epochs by determining and changing the hyperparameters used.

Proses Testing Model

When the learning has reached the target according to the input epoch value, the iteration process stops and then the model is tested with test data. In this process, it reloads the learned model that has been produced in the previous training process and calculates the output results based on the bias and weight values given during the training process that are already in the learned model. Then it is compared with the data generated by the LSTM method at the specified time range with the accuracy method used using MSE.

LSTM Manual Calculation

Before the system design was carried out, the researcher conducted a descriptive analysis for the design of the system from the LSTM Method with manual/numerical calculations on the data Some of the data obtained directly from the Binaka Gunung Sitoli Climatology Station. The following is an example of a manual calculation of an LSTM network.

Table 1. Hyperparameter Initialization Value

Hyperparameter	Value
Number of Epochs	50
Batch Size (**n**)	1
Panjang Timestep/Sequence	2
Features	rr
Number of LSTM Units	1
Number of Hidden Layers	1
Learning rate ($\boldsymbol{\gamma}$)	0.1
Probabilitas Dropout	0

Tabel 2. Data Yang di Gunakan Dalam Perhitungan Manual LSTM

No	Tanggal	rr	(rr) MinMaxScale	Input		y (label)
				x_1	x_2	
1	16 Oktober 2004	0.5	0.0052	0.0052	0.0206	1.0000
2	17 Oktober 2004	2	0.0206	0.0206	1.0000	0.5258
3	18 Oktober 2004	97	1.0000	1.0000	0.5258	0.6206
4	19 Oktober 2004	51	0.5258	0.5258	0.6206	0.2371
5	20 Oktober 2004	60.2	0.6206	0.6206	0.2371	0.1031
6	21 Oktober 2004	23	0.2371	0.2371	0.1031	0.1753
7	22 Oktober 2004	10	0.1031	0.1031	0.1753	0.0206
8	23 Oktober 2004	17	0.1753	0.1753	0.0206	0.0010
9	24 Oktober 2004	2	0.0206	0.0206	0.0010	0.1546
10	25 Oktober 2004	0.1	0.0010	0.0010	0.1546	0.0897
…	…	…	...			
60	14 Desember 2004	1.2	0.0124			

DOI: https://doi.org/10.31004/riggs.v4i2.487

As seen in table 2 of the 60 data rows used, the data is divided into a 9:1 scale for training data and data testing, where data rows 1-54 are used for training data and data rows 55-60 are used for data testing. The value of the input (x) and label (y) is based on the form of a sliding windows data pattern where the input length is based on the timestep length of the initialization of the hyperprameter value as seen in table 1 of the value used, for this manual calculation the researcher only uses 2 timestep values which means that there are only 2 inputs for the model that is designed, namely the x_1 and x_2 inputs, Where the input x_1 is the value of the feature line to n, and the input value x_2 is the value of the line n+1, and for the value of the label y is the value of the feature line to n+2 and so on until the limit of the data used from each training and testing data.

Initialization of the dimensions of each parameter is indispensable for the construction of the basic blocks of the LSTM network including other types of artificial neural networks. The dimensions of all parameters of the LSTM depend on the dimensions of the hidden unit. For a more complete explanation, you can see the following table 3.

Table 3. Parameter Dimensions on LSTM Network

Parameter	Information	Dimension
Units	Number of unit neurons in the hidden layer	(n_h)
feature	Jumlah Feature	(f)
n_x	Input size (timestep length)	(n_x)
$\mathbf{C}^{<t-1>}$	Previous cell state dimensions	(n_h, f)
$\mathbf{h}^{<t-1>}$	Previous output dimensions	(n_h, f)
$\mathbf{x}^{<t>}$	Current input	(n_x, f)
$[\,]\mathbf{x}^{<t>}\mathbf{h}^{<t-1>}$	Combination of previous output and current input	$(n\_x + n\_h, f)$
$\mathbf{W_f}, \mathbf{W_i}, \mathbf{W_c}, \mathbf{W_o}$	Weight for all gates	$(n\_h, n\_x + n\_h)$
$\mathbf{b_f}, \mathbf{b_i}, \mathbf{b_c}, \mathbf{b_o}$	Bias for all operations	$(n\_h, 1)$

As seen in table 3. In the Dimensions column, Almost every dimension of the parameter in the LSTM has a direct or indirect relationship with the hidden unit (n_h) in the LSTM layer, and it is important to understand that the multiplication of the matrix of 2 measures (a,b)*(b,c) results in a measured output (a,c). After analyzing the dimensions of the components, the next step is to analyze the output dimensions of each component. For a more complete explanation, you can see the following table 4.

Table 4. Output Dimensions on LSTM Network

Output	Operation	Dimension
$\mathbf{f_t}$	$\sigma(W_f * [x^{<t>} \quad h^{<t-1>}] + b_f)$	$(n\_h, n\_x + n\_h) * (n\_x + n\_h, m) + (n\_h, 1) = (n\_h, m)$
$\mathbf{i_t}$	$\sigma(W_i * [x^{<t>} \quad h^{<t-1>}] + b_i)$	$(n\_h, n\_x + n\_h) * (n\_x + n\_h, m) + (n\_h, 1) = (n\_h, m)$
$\tilde{\mathbf{C}}_\mathbf{t}$	$\tanh(W_c * [x^{<t>} \quad h^{<t-1>}] + b_c)$	$(n\_h, n\_x + n\_h) * (n\_x + n\_h, m) + (n\_h, 1) = (n\_h, m)$
$\mathbf{o_t}$	$\sigma(W_o * [x^{<t>} \quad h^{<t-1>}] + b_o)$	$(n\_h, n\_x + n\_h) * (n\_x + n\_h, m) + (n\_h, 1) = (n\_h, m)$
$\mathbf{C_t}$	$f_t * C_{t-1} + i_t * \tilde{C}_t$	$(n\_h, m) * (n\_h, m) + (n\_h, m) * (n\_h, m) = (n\_h, m)$
$\mathbf{h_t}$	$o_t * \tanh(C_t)$	$(n\_h, m) * (n\_h, m) = (n\_h, m)$

Training Model LSTM

After knowing the initial weight value, then the training process begins. In the algorithm training process in the LSTM model, forward step and backward step are used. The following is an example of a manual calculation of an LSTM network. Calculations starting from epoch 1 and batch 1 for one iteration are calculated numerically or

DOI: https://doi.org/10.31004/riggs.v4i2.487

manually as follows. After determining the hyperparameters, initial parameters/values, and the next dimension is to calculate forward step timestep 1 as follows.

$$
\begin{aligned}
f_t &= \sigma\left(W_f * [x^{<t>} \quad h^{<t-1>}] + b_f\right) \\
&= \sigma((n\_h, n\_x + n\_h) * (n\_x + n\_h, f) + (n\_h, 1)) \\
&= \sigma([0.5774\ 0.5774\ 0.5774] * [0.0052 \quad 0.0206 \quad 0] + [0]) \\
&= \sigma(0.0149) \\
&= \frac{1}{1 + e^{-0.0149}} \\
&= 0.5037 \\
i_t &= \sigma(W_i * [x^{<t>} \quad h^{<t-1>}] + b_i) \\
&= \sigma((n\_h, n\_x + n\_h) * (n\_x + n\_h, f) + (n\_h, 1)) \\
&= \sigma([0.5774 \quad 0.5774 \quad -0.5774] * [0.0052 \quad 0.0206 \quad 0] + [0]) \\
&= \sigma(0.0149) \\
&= \frac{1}{1 + e^{-0.0149}} \\
&= 0.5037 \\
\tilde{C}_t &= tanh(W_c * [x^{<t>} \quad h^{<t-1>}] + b_c) \\
&= tanh((n\_h, n\_x + n\_h) * (n\_x + n\_h, f) + (n\_h, 1)) \\
&= tanh([0.5774 \quad 0.5774 \quad 0.5774] * [0.0052 \quad 0.0206 \quad 0] + [0]) \\
&= tanh(0.0149) \\
&= 2 * \left(\frac{1}{1 + e^{-2*0.0149}}\right) - 1 \\
&= 0.0149 \\
o_t &= \sigma(W_o * [x^{<t>} \quad h^{<t-1>}] + b_o) \\
&= \sigma((n\_h, n\_x + n\_h) * (n\_x + n\_h, f) + (n\_h, 1)) \\
&= \sigma([0.5774 \quad 0.5774 \quad 0.5774] * [0.0052 \quad 0.0206 \quad 0] + [0]) \\
&= \sigma(0.0149) \\
&= \frac{1}{1 + e^{-0.0149}} \\
&= 0.5037 \\
C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\
&= 0.5037 * 0 + 0.5037 * 0.0149 \\
&= 0.0075 \\
h_t &= o_t * tanh(C_t) \\
&= 0.5037 * tanh(0.0075) \\
&= 0.5037 * \left(2 * \left(\frac{1}{1 + e^{-2*0.0075}}\right) - 1\right) \\
&= 0.0037
\end{aligned}
$$

Because the batch size is no more than 1, each forward step is completed in 1 timestep directly to the backward step.

$$
\begin{aligned}
\delta h_t &= h_t - y^{<t>} + \Delta h_{t+1} \\
&= 0.0037 - 1.0 + 0 \\
&= -0.9962 \\
\delta C_t &= \delta h_t * o_t * (1 - tanh(C_t)^2) + \delta C_{t+1} * f_{t+1} \\
&= -0.9962 * 0.5037 * (1 - tanh(0.0075)^2) + 0 * 0 \\
&= -0.5018 \\
\delta o_t &= \delta h_t * tanh(C_t) * o_t * (1 - o_t) \\
&= -0.9962 * tanh(0.0075) * 0.5037 * (1 - 0.5037) \\
&= -0.0019 \\
\widetilde{\delta C}_t &= \delta C_t * i_t * \left(1 - \tilde{C}_t^{\,2}\right) \\
&= -0.5018 * 0.5037 * (1 - 0.0149^2) \\
&= -0.2527 \\
\delta i_t &= \delta C_t * \tilde{C}_t * i_t * (1 - i_t)
\end{aligned}
$$

DOI: https://doi.org/10.31004/riggs.v4i2.487

$$= -0.5018 * 0.0149 * 0.5037 * (1 - 0.5037)$$
$$= -0.0019$$
$$\delta f_t = \delta C_t * C_{t-1} * f_t * (1 - f_t)$$
$$= -0.5018 * 0 * 0.5037 * (1 - 0.5037)$$
$$= 0.0000$$
$$U_f = weight\ h\ dari\ W_f\ [x_1 \quad x_2 \quad h]\ [0.5774 \quad 0.5774 \quad 0.5774]$$
$$= 0.5774$$
$$U_i = weight\ h\ dari\ W_i\ [x_1 \quad x_2 \quad h]\ [0.5774 \quad 0.5774 \quad 0.5774]$$
$$= 0.5774$$
$$U_c = weight\ h\ dari\ W_c\ [x_1 \quad x_2 \quad h]\ [0.5774 \quad 0.5774 \quad 0.5774]$$
$$= 0.5774$$
$$U_o = weight\ h\ dari\ W_o\ [x_1 \quad x_2 \quad h]\ [0.5774 \quad 0.5774 \quad 0.5774]$$
$$= 0.5774$$
$$\Delta h_t = [U_f \quad U_i \quad U_c \quad U_o] * [\delta f_t \quad \delta i_t \quad \widetilde{\delta C_t} \quad \delta o_t]$$
$$= \begin{bmatrix} 0.5774 \\ 0.5774 \\ 0.5774 \\ 0.5774 \end{bmatrix} * \begin{bmatrix} 0.0000 \\ -0.0019 \\ -0.2527 \\ -0.0019 \end{bmatrix}$$
$$= -0.1481$$

Based on the optimizer used is Stochastic Gradient Descent (SDG), then bias and weight updates are carried out at each timestep.

$$W_f{}^{new} = W_f{}^{old} - \gamma * \sum_{t=1}^{n} [\delta f_t] * [x^{<t>} \quad h^{<t-1>}]$$
$$= W_f{}^{old} - 0.1 * ([0.0000] * [0.0052 \quad 0.0206 \quad 0.0000])$$
$$= [0.5774 \quad 0.5774 \quad 0.5774] - 0.1 * [0.0000 \quad 0.0000 \quad 0.0000]$$
$$= [0.5774 \quad 0.5774 \quad 0.5774]$$
$$b_f{}^{new} = b_f{}^{old} - \gamma * \sum_{t=1}^{n} [\delta f_t]$$
$$= 0 - 0.1 * ([0.0000])$$
$$= 0.0000$$
$$W_i{}^{new} = W_i{}^{old} - \gamma * \sum_{t=1}^{n} [\delta i_t] * [x^{<t>} \quad h^{<t-1>}]$$
$$= W_i{}^{old} - 0.1 * ([-0.0019] * [0.0052 \quad 0.0206 \quad 0.0000])$$
$$= [0.5774 \quad 0.5774 \quad 0.5774] - 0.1 * [-0.00001 \quad -0.00004 \quad 0.000]$$
$$= [0.577401 \quad 0.577404 \quad 0.5774]$$
$$b_i{}^{new} = b_i{}^{old} - \gamma * \sum_{t=1}^{n} [\delta i_t]$$
$$= 0 - 0.1 * ([-0.0019])$$
$$= 0.0002$$
$$W_c{}^{new} = W_c{}^{old} - \gamma * \sum_{t=1}^{n} [\widetilde{\delta C_t}] * [x^{<t>} \quad h^{<t-1>}]$$
$$= W_c{}^{old} - 0.1 * ([-0.2527] * [0.0052 \quad 0.0206 \quad 0.0000])$$
$$= [0.5774 \quad 0.5774 \quad 0.5774] - 0.1 * [-0.0013 \quad -0.0052 \quad 0.0000]$$
$$= [0.57753 \quad 0.57792 \quad 0.5774]$$
$$b_c{}^{new} = b_c{}^{old} - \gamma * \sum_{t=1}^{n} [\widetilde{\delta C_t}]$$
$$= 0 - 0.1 * ([-0.2527])$$
$$= 0.0253$$

DOI: https://doi.org/10.31004/riggs.v4i2.487

$$W_o{}^{new} = W_o{}^{old} - \gamma * \sum_{t=1}^{n} [\delta o_t] * [x^{<t>} \quad h^{<t-1>}]$$
$$= W_o{}^{old} - 0.1 * ([-0.0019] * [0.0052 \quad 0.0206 \quad 0.0000])$$
$$= [0.5774 \quad 0.5774 \quad 0.5774] - 0.1 * [-0.00001 \quad -0.00004 \quad 0.000]$$
$$= [0.577401 \quad 0.577404 \quad 0.5774]$$
$$b_o{}^{new} = b_o{}^{old} - \gamma * \sum_{t=1}^{n} [\delta o_t]$$
$$= 0 - 0.1 * ([-0.0019])$$
$$= 0.0002$$

The Accuracy value is calculated using the MSE formula from the calculation of timeteps decay in batch 1 as follows.

$$MSE = \frac{\sum_{t=1}^{n}(y_t - h_t)^2}{n}$$
$$= \frac{((1.0000 - 0.0037)^2)}{1}$$
$$= 0.992463094$$

Based on the results of the error value from batch 1 data, the error value was 0.992463094. Where the error value of 0.992463094 means that the prediction feature (RR) has an inaccuracy level of approximately 0.992463094 in predicting rainfall from rainfall data normalized by Minmaxscaller.

The calculation of batches 2-52 is the same as before, only some parameters such as bias and weight are updated after calculating the Stochastic Gradient Descent (SGD) from the previous batch calculation. Forward batch 2-52 is the result of the forward calculation in batch 2-52 as seen in the following table 5.

Tabel 5. Hasil Forward Epoch 1 Batch 2-52

batch	time step	Forward					
		Gate				C_t	h_t
		f_t	i_t	$\tilde{C}_t$	o_t		
2	1	0.6432	0.6432	0.5477	0.6432	0.3523	0.2177
3	1	0.7070	0.7078	0.7254	0.7078	0.5134	0.3345
4	1	0.6597	0.6615	0.6108	0.6618	0.4041	0.2537
5	1	0.6213	0.6230	0.4924	0.6232	0.3068	0.1854
6	1	0.5490	0.5502	0.2301	0.5503	0.1266	0.0693
7	1	0.5401	0.5414	0.1992	0.5416	0.1079	0.0582
8	1	0.5282	0.5294	0.1508	0.5296	0.0798	0.0422
9	1	0.5031	0.5042	0.0492	0.5043	0.0248	0.0125
10	1	0.5225	0.5237	0.1310	0.5238	0.0686	0.0359
…	...	…	…	…	…	…	…
52	1	0.5711	0.5749	0.4223	0.5751	0.2428	0.1369

Table 5 shows the results of the LSTM forward step at epoch 1 of each timestep in batches 2-52 with a batch size of no more than 1 timestep.

Backward batch 2-52 is the result of the backward calculation in batch 2-52 as seen in the following table 6**.**

Tabel 6. Hasil Backward Epoch 1 Batch 2-52

batch	Backward						
	δh_t	δC_t	Gate				Δh_t
			δo_t	$\widetilde{\delta C}_t$	δi_t	δf_t	
2	-0.3081	-0.1755	-0.0239	-0.0790	-0.0221	0.0000	-0.0722
3	-0.2861	-0.1573	-0.0280	-0.0528	-0.0236	0.0000	-0.0602
4	0.0166	0.0094	0.0014	0.0039	0.0013	0.0000	0.0038
5	0.0823	0.0467	0.0057	0.0221	0.0054	0.0000	0.0192
6	-0.1060	-0.0574	-0.0033	-0.0299	-0.0033	0.0000	-0.0211
7	0.0376	0.0201	0.0010	0.0105	0.0010	0.0000	0.0072

DOI: https://doi.org/10.31004/riggs.v4i2.487

8	0.0411	0.0217	0.0008	0.0112	0.0008	0.0000	0.0074
9	-0.1421	-0.0716	-0.0009	-0.0360	-0.0009	0.0000	-0.0218
10	-0.0538	-0.0281	-0.0009	-0.0144	-0.0009	0.0000	-0.0094
…	…	…	…	…	…	…	…
52	0.1369	0.0743	0.0080	0.0351	0.0077	0.0000	0.0293

In table 6. shows the LSTM backward step results at epoch 1 of each timestep in batches 2-52 with a batch size of no more than 1 timestep. The Error value is calculated using the MSE formula from the calculation of timeteps decay in batches 2-52 as shown in the following table 7.

Tabel 7. Nilai Error Epoch 1 Batch 2-52

Batch	y_t	h_t	MSE
2	0.525773	0.217691	0.094914
3	0.620619	0.334509	0.081859
4	0.237113	0.253746	0.000277
5	0.103093	0.185386	0.006772
6	0.175258	0.069303	0.011226
7	0.020619	0.058186	0.001411
8	0.001031	0.042180	0.001693
9	0.154639	0.012518	0.020198
10	0.089691	0.035888	0.002895
…	…	…	…
52	0.000000	0.136923	0.018748

Based on table 7, the results of the MSE error value from epoch 1 in the error value as seen in the MSE column means that the prediction feature (rr) of each batch has a level of inaccuracy more or less as seen in the column in predicting rainfall from rainfall data that is normalized by the minmaxscaller. To find out the total error of 1 epoch, it is necessary to find the average value of the total of all error values of each batch.

$$\sum MSE = \frac{\begin{pmatrix} 0.992463 + 0.094914 + 0.081859 + 0.000277 + 0.006772 + \\ 0.011226 + 0.001411 + 0.001693 + 0.020198 + 0.002895 + \\ 0.050616 + 0.005471 + 0.121181 + 0.000061 + 0.003715 + \\ 0.015693 + 0.017365 + 0.031130 + 0.037291 + 0.016880 + \\ 0.013968 + 0.009037 + 0.021404 + 0.002102 + 0.409258 + \\ 0.012966 + 0.016350 + 0.307339 + 0.000029 + 0.113373 + \\ 0.025934 + 0.019826 + 0.001033 + 0.000019 + 0.009310 + \\ 0.638287 + 0.054834 + 0.008875 + 0.006308 + 0.008717 + \\ 0.000151 + 0.241178 + 0.010134 + 0.022709 + 0.002557 + \\ 0.551693 + 0.043619 + 0.006199 + 0.003975 + 0.172749 + \\ 0.012568 + 0.018748 + \end{pmatrix}}{52}$$

$$= 0.082276130$$

Based on the results of the total MSE error value from the epoch 1 data, the error value was 0.082276130. Where the error value of 0.082276130 means that the prediction feature (rr) has an inaccuracy level of approximately 0.082276130 in predicting rainfall from rainfall data that is normalized by minmaxscaller.

In the calculation of Epoch 2-50, the error value is calculated using the total of the MSE formula as shown in the following table 8.

Tabel 8. Nilai *Error Epoch* 2-50

Epoch	$\sum$ MSE
2	0.075683
3	0.072499
4	0.070754
5	0.069697
6	0.069002
7	0.068510

DOI: https://doi.org/10.31004/riggs.v4i2.487

8	0.068140
9	0.067844
10	0.067598
…	…
50	0.064291

Based on table 8 the results of the MSE error value from epoch 2-52 in the error value as seen in the column, where the error value always decreases along with the epoch repetition for 50 times. This means that the prediction feature (RR) of each epoch iteration that is carried out has a level of inaccuracy that continues to decrease as seen in the column in predicting rainfall from rainfall data that is normalized by Minmaxscaller.$\sum$ MSE

Testing Model LSTM
After conducting the LSTM Training Model, the bias value and optimal weight of 50 epochs are known as in the following table 9.

Tabel 9. Bias dan Weight dari Learned Model

Parameters	Forget Gate	Input Gate	Memory Gate	Output Gate
bias	[0.0000]	[0.0257]	[0.8835]	[−0.0236]
weight	[0.5774	[0.4437	[0.3817	[0.4342
	0.5774	0.3524	0.3405	0.3135
	0.5774]	0.5774]	0.5774	0.5574]

The optimal results of the bias and weight of the learned model were used to test the testing data. In the algorithm testing process in the LSTM model that is used, only a forward step as much as the data is tested as seen in the following table 10**.**

Tabel 10. Hasil Testing Model

t	y_t	h_t	MSE
1	0.0082	0.1878	0.032252586
2	0.0000	0.1716	0.029432851
3	0.0175	0.1709	0.023530073
4	0.0124	0.1716	0.025350684

In table 10 for the same error value as in the training stage, the researcher only uses the MSE formula for the calculation of the error value.

$$\sum MSE = \frac{0.032252586 + 0.029432851 + 0.023530073 + 0.025350684}{4}$$
$$= 0.027641549$$

Based on the results of the MSE error value from the testing process, the error value was 0.027641549. Where the error value of 0.027641549 means that the prediction feature (rr) at the time of testing has an inaccuracy level of approximately 0.027641549 in predicting the feature/variable (rr) of rainfall. It can be said that the projection or manual calculation of LSTM carried out by the researcher using only 1 feature/variable, namely (rr) rainfall, with several other hyperparameter values, and using deep learning with the Long Short-Term Memory method with a backpropagation algorithm is still far from the expected accuracy value.

4. Conclusion

Based on analysis using the Deep Learning method based on Long Short-Term Memory (LSTM), this study successfully shows that the LSTM approach is able to identify historical rainfall data patterns and provide relevant predictions for the coming period. By leveraging the power of LSTM in handling sequential data, this research contributes to the development of predictive technologies that are more accurate and adaptive to climate dynamics. The application of this model in an integrated prediction system shows the potential for real application in the context of decision-making, particularly in the agriculture, water resource management, and data-driven disaster mitigation sectors. This research not only strengthens the understanding of the effectiveness of LSTM methods in

DOI: https://doi.org/10.31004/riggs.v4i2.487

time-series prediction, but also opens up opportunities for further development in other climate data processing, the complexity of which demands more sophisticated artificial intelligence-based predictive approaches.

Reference

[1] A. Wahyudi, M. B. T. Assyamiri, W. Al Aluf, M. R. Fadhillah, S. Yolanda, and M. I. Anshori, "Dampak transformasi era digital terhadap manajemen sumber daya manusia," *J. Bintang Manaj.*, vol. 1, no. 4, pp. 99–111, 2023.

[2] L. Ahmad Gunawan, *Digital Leadership for Industry 5.0: Integrasi Manusia, Teknologi dan Industri*. Takaza Innovatix Labs, 2025.

[3] M. Hasan, S. Kom, M. Kom, S. Serwin, and M. Kom, *Penerapan Sistem Informasi Berbasis AI untuk Analisis Data Real-time*. Takaza Innovatix Labs, 2024.

[4] S. Rifky *et al.*, *Artificial Intelligence: Teori dan Penerapan AI di Berbagai Bidang*. PT. Sonpedia Publishing Indonesia, 2024.

[5] M. S. Novelan, S. Efendi, P. Sihombing, and H. Mawengkang, "VEHICLE ROUTING PROBLEM OPTIMIZATION WITH MACHINE LEARNING IN IMBALANCED CLASSIFICATION VEHICLE ROUTE DATA.," *Eastern-European J. Enterp. Technol.*, vol. 125, no. 3, 2023.

[6] D. Sawitri, "PERAN DEEP LEARNING DAN BIG DATA DALAM MENDEKTEKSI MASALAH KEUANGAN," *Djtechno J. Teknol. Inf.*, vol. 6, no. 1, pp. 193–207, 2025.

[7] H. Alizadegan, B. Rashidi Malki, A. Radmehr, H. Karimi, and M. A. Ilani, "Comparative study of long short-term memory (LSTM), bidirectional LSTM, and traditional machine learning approaches for energy consumption prediction," *Energy Explor. Exploit.*, vol. 43, no. 1, pp. 281–301, 2025.

[8] Y. A. Susetyo, H. A. Parhusip, S. Trihandaru, and B. Susanto, "LSTM-IOT (LSTM-based IoT) untuk Mengatasi Kehilangan Data Akibat Kegagalan Koneksi," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 12, no. 1, pp. 175–186, 2025.

[9] V. Shatravin, D. Shashev, and S. Shidlovskiy, "Sigmoid activation implementation for neural networks hardware accelerators based on reconfigurable computing environments for low-power intelligent systems," *Appl. Sci.*, vol. 12, no. 10, p. 5216, 2022.

[10] A. Sepas-Moghaddam, A. Etemad, F. Pereira, and P. L. Correia, "Long short-term memory with gate and state level fusion for light field-based face recognition," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 1365–1379, 2020.

DOI: https://doi.org/10.31004/riggs.v4i2.487