RAGAM: Journal of Statistics and Its Application
Vol 05 No 01, Pages 44-53, February 2026
DOI: https://doi.org/10.20527/ragam.v5i1.18027
e-ISSN: 2962-8539

Forecasting the Rupiah Exchange Rate Against the Euro Utilising Long Short-Term Memory and Gated Recurrent Unit Techniques

Rahmat Hidayat
Kementerian Komunikasi dan Digital, Indonesia
Corresponding author: imjustrahmat2722@gmail.com

Abstract
The volatility of the Rupiah exchange rate against the Euro poses significant challenges for Indonesia's economic stability, particularly in periods of global financial uncertainty. Accurate forecasting of exchange rates is essential for supporting monetary policy decisions and guiding economic planning. This study aims to forecast the daily Rupiah-to-Euro exchange rate using deep learning approaches, specifically Long-Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) models, based on historical data from Bank Indonesia spanning January 2, 2018 to February 14, 2023. Both methods are variants of Recurrent Neural Networks (RNN) capable of learning long-term dependencies through memory cells, with GRU offering a simpler and more efficient architecture than LSTM. This study employs a univariate forecasting framework and evaluates model performance using Mean Squared Error (MSE) during training and Mean Absolute Percentage Error (MAPE) during testing. The results show that the optimal LSTM model (1 unit, batch size 32, timestep 50) achieved an MSE of 0.001451447 and a MAPE of 0.2321061%, outperforming the best GRU model with an MSE of 0.001475483 and a MAPE of 0.2412439%. These findings demonstrate that LSTM provides marginally superior forecasting accuracy for Rupiah-Euro exchange rate prediction, offering valuable insights for financial decision-making under economic volatility.

Keywords: *Exchange Rate, Forecasting, GRU, LSTM, Neural Network.*

Article History:
Received : 12 January 2026
Revised : 08 March 2026
Accepted : 09 March 2026

How to Cite:
R. Hidayat, "Forecasting the Rupiah Exchange Rate Against the Euro Utilising Long Short-Term Memory and Gated Recurrent Unit Techniques," RAGAM: *Journal of Statistics and Its Application*, vol. 5, no. 1, pp. 44–53, 2026, doi: 10.20527/ragam.v5i1.18207.

1. INTRODUCTION

The global economic recession predicted for 2023 posed significant risks to Indonesia, particularly regarding the weakening of the national currency against foreign currencies. Economic recession is defined as a significant and sustained decline in economic activity, typically characterized by GDP contraction. Its contributing factors include economic shocks, high inflation, rising benchmark interest rates, and declining global demand [1]. Among the key consequences of economic instability is increased volatility in foreign exchange markets, which directly affects Indonesia's trade balance and monetary policy. Consequently, forecasting the exchange rate serves as a critical approach to predict future currency fluctuations based on historical data. Forecasting is a systematic activity designed to predict or estimate future occurrences based on past observations [2]. The Rupiah-Euro exchange rate is particularly relevant given the importance of the Eurozone as one of Indonesia's major trading and investment partners.

Traditional econometric models such as ARIMA have been widely used for exchange rate forecasting; however, they often struggle to capture nonlinear patterns and complex temporal dependencies inherent in financial time series data. Deep learning methods, particularly Recurrent Neural Networks (RNN) and their variants, offer greater flexibility in modeling such complex dynamics. Long-Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) are specialized recurrent neural network architectures applicable to time series forecasting. Both LSTM and GRU are variants of RNN developed to better store and access information through memory cells, granting them stronger long-term memory than standard RNNs, resulting in models with superior accuracy [3]. GRU is a modification of the LSTM model featuring a simpler architecture, which leads to more efficient computation. Since LSTM and GRU models often possess comparable prediction accuracy, the determination of which model yields better accuracy for a specific prediction depends heavily on the dataset used [4].

Despite the growing literature on deep learning for financial forecasting, limited studies have directly compared LSTM and GRU performance specifically for the Rupiah-Euro exchange rate using daily data over a multi-year period. This research gap motivates the present study, which examines the forecasting of the Rupiah exchange rate against the Euro using LSTM and GRU methods. The research aims to identify the optimal forecasting model by comparing LSTM and GRU for predicting the Rupiah-Euro exchange rate. Furthermore, this research provides additional information for readers regarding currency forecasting in the context of economic instability.

2. LITERATURE REVIEW

2.1. Exchange Rate

The exchange rate is defined as the value of a domestic currency in terms of foreign currency, or the price of domestic currency required to purchase one unit of foreign currency [5]. Variations in exchange rates are observable in two distinct forms: appreciation and depreciation. Appreciation occurs when the value of the domestic currency increases against foreign currencies, while depreciation occurs when the value of the domestic currency decreases against foreign currencies [6].

2.2. Artificial Neural Network (ANN)

Artificial Neural Network (ANN) is an information processing system that exhibits performance characteristics similar to biological neural networks in the human brain [7]. The advantages of ANN include its ability to process non-linear and complex data, handle varied and unstructured data, and its adaptive nature in learning and adjusting based on changing inputs [7]. Generally, ANN consists of inputs, weights, summation functions, activation functions, and outputs. Weights are adaptive coefficients that determine the intensity of inputs and store the model's memory [8].

2.3. Recurrent Neural Network (RNN)

A Recurrent Neural Network (RNN) is a technique formulated to identify patterns in sequential data. RNNs employ a recurrent mechanism that enables the network to retain historical information [9]. Standard RNNs have effective short-term memory but are deficient in long-term memory, rendering them unable of retaining prior information for extended durations [10].

2.4. Long-Short Term Memory (LSTM)

Long-Short Term Memory (LSTM) networks were introduced by Hochreiter and Schmidhuber in 1997 to rectify the limitations inherent in Recurrent Neural Networks (RNNs). LSTM integrates a memory cell that facilitates the storage and retrieval of long-term information, thereby enabling the model to preserve critical data from the remote past [10]. The architectural design of LSTM consists of three distinct gates: the forget gate, the input gate, and the output gate. The function of the forget gate is to ascertain which information should be discarded, the input gate oversees the integration of novel information to be preserved, and the output gate delineates the information that is to be released [11]. The memory cell within an LSTM, commonly referred to as the cell state, is constituted of fixed self-connections and is regulated by the aforementioned three gates: the forget gate, the input gate, and the output gate. These gates possess the capability to capture essential information and make determinations regarding retention or elimination. The forget gate ascertains whether the information contained within the cell state will be maintained or altered, the input gate manages the influx of new information into the cell state, and the output gate directs the information residing in the cell state that is to be communicated [9]. The configuration of an LSTM cell is depicted in Figure 1.

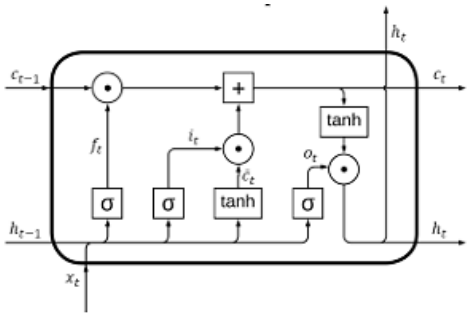

Figure 1. Architecture of Long-Short Term Memory Cell

The algorithm for the LSTM cell is as follows.

1. The forget gate ($\boldsymbol{f(t)}$) receives information from the current input $\boldsymbol{x(t)}$ and the concealed state from the previous timestep $\boldsymbol{h(t-1)}$. The forget gate ascertains which information from the prior cell state to discard, use the sigmoid activation function (σ).

$$\boldsymbol{f(t)} = \sigma(\boldsymbol{bf} + \boldsymbol{Ufx(t)} + \boldsymbol{Wfh(t-1)}) \quad (1)$$

2. The input gate ($\boldsymbol{i}(\boldsymbol{t})$) will determine the quantity of information to be incorporated into the cell state, utilising a sigmoid activation function (σ).

$$\boldsymbol{i}\,(\boldsymbol{t}) = \sigma(\boldsymbol{bi} + \boldsymbol{Uix}(\boldsymbol{t}) + \boldsymbol{Wih}(\boldsymbol{t}-\boldsymbol{1})) \quad (2)$$

3. The candidate cell state ($\tilde{\boldsymbol{c}}(\boldsymbol{t})$) is derived using the tanh activation function.

$$\tilde{\boldsymbol{c}}\,(\boldsymbol{t}) = \tanh(\boldsymbol{bc} + \boldsymbol{Ucx}(\boldsymbol{t}) + \boldsymbol{Wch}(\boldsymbol{t}-\boldsymbol{1})) \quad (3)$$

4. Cell state ($\boldsymbol{c}(\boldsymbol{t})$) will update the old cell state value to the new cell state value,

   by combining the forget gate information with the previous cell state and the gate input with the candidate cell state.

$$\boldsymbol{c}\,(\boldsymbol{t}) = \boldsymbol{f}(\boldsymbol{t}) \odot \boldsymbol{c}(\boldsymbol{t}-\boldsymbol{1}) + \boldsymbol{i}(\boldsymbol{t}) \odot \tilde{\boldsymbol{c}}(\boldsymbol{t}) \quad (4)$$

5. The output gate ($\boldsymbol{o}(\boldsymbol{t})$) will select and decide how much information is stored in the cell state to output and use as input to the hidden state, using the sigmoid activation function (σ).

$$\boldsymbol{o}\,(\boldsymbol{t}) = \sigma(\boldsymbol{bo} + \boldsymbol{Uox}(\boldsymbol{t}) + \boldsymbol{Woh}(\boldsymbol{t}-\boldsymbol{1})) \quad (5)$$

6. The new hidden state ($\boldsymbol{h}(\boldsymbol{t})$) is obtained from the combination of the output with the new cell state value, with the tanh activation function.

$$\boldsymbol{h}\,(\boldsymbol{t}) = \boldsymbol{o}(\boldsymbol{t}) \odot \tanh(\boldsymbol{C}(\boldsymbol{t})) \quad (6)$$

   where, $\odot$ is element-wise multiplication, $\boldsymbol{b}$ is bias, and $\boldsymbol{U}$ and $\boldsymbol{W}$ are weights.

2.5. Gated Recurrent Unit (GRU)

The Gated Recurrent Unit (GRU), presented by Cho et al. (2014), is a streamlined version of LSTM that omits an explicit cell state [9]. The GRU necessitates fewer gates for updating the hidden state, comprising solely a reset gate and an update gate. The reset gate regulates the retention of information from the prior hidden state, whereas the update gate governs the degree of preservation of the previous hidden state [3]. The update gate regulates and determines which information from the preceding concealed state will be preserved [6]. The construction of the GRU cell is illustrated in Figure 2.

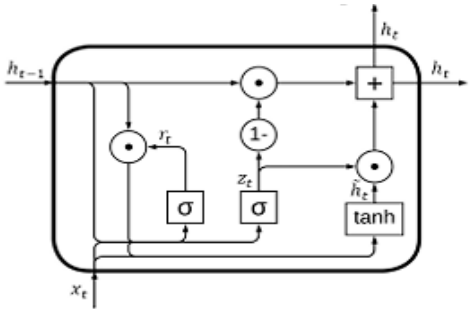

Figure 2. Gated Recurrent Unit Cell Architecture

The algorithm for the GRU cell is as follows .

1. The reset gate ($\boldsymbol{r}(\boldsymbol{t})$) acquires the current input $\boldsymbol{x}(\boldsymbol{t})$ and the hidden state from the preceding timestep h($\boldsymbol{t}-\boldsymbol{1}$). The reset gate utilises the sigmoid activation function

(σ) to choose and quantify the information retained from the preceding concealed state.

$$\mathrm{r}(\boldsymbol{t}) = \sigma(\boldsymbol{br} + \boldsymbol{Urx}(\boldsymbol{t}) + \boldsymbol{Wrh}(\boldsymbol{t-1})) \quad (7)$$

2. The update gate ($\boldsymbol{z}(\boldsymbol{t})$) regulates the retention of information from the preceding hidden state by the sigmoid activation function (σ).

$$\boldsymbol{z}\,(\boldsymbol{t}) = \sigma(\boldsymbol{bz} + \boldsymbol{Uzx}(\boldsymbol{t}) + \boldsymbol{Wzh}(\boldsymbol{t-1}))\,) \quad (8)$$

3. A candidate hidden state ($\tilde{\boldsymbol{h}}(\boldsymbol{t})$) is generated with the tanh activation function.

$$\tilde{\mathbf{h}}\,(\boldsymbol{t}) = \tanh(\boldsymbol{bh} + \boldsymbol{Uhx}(\boldsymbol{t}) + \boldsymbol{Wh}[\boldsymbol{r}(\boldsymbol{t}) \odot \boldsymbol{h}(\boldsymbol{t-1})]) \quad (9)$$

4. A new hidden state ($\boldsymbol{h}(\boldsymbol{t})$) is created from the combination of the reset gate and the hidden state candidate with the reset gate and the previous hidden state.

$$\mathbf{h}(\boldsymbol{t}) = (1-\boldsymbol{z}(\boldsymbol{t})) \odot \boldsymbol{h}(\boldsymbol{t-1}) + \boldsymbol{z}(\boldsymbol{t}) \odot \tilde{\boldsymbol{h}}(\boldsymbol{t}) \quad (10)$$

where $\odot$ is element-wise multiplication, $\boldsymbol{b}$ is the bias, and $\boldsymbol{U}$ and $\boldsymbol{W}$ are the weights.

In LSTM and GRU architectures, the hidden state value constitutes the output of the respective LSTM or GRU cell. The concealed state value is subsequently utilised as input for the output layer, which processes it to get the final predicted output.

2.6. Activation Functions

Non-linear activation functions are crucial for establishing intricate mappings in neural networks. LSTM and GRU generally employ Sigmoid and Tanh functions [10]. The Sigmoid function produces outputs ranging from 0 to 1, rendering it appropriate for gating systems that regulate information flow [11]. The Tanh (Hyperbolic Tangent) function produces values ranging from -1 to 1, enabling it to process both positive and negative information efficiently. The sigmoid activation function possesses a "S" shape and is frequently employed in artificial neural networks (ANNs) [12]. The sigmoid activation function yields a positive output value within the range of 0 to 1, making it suitable for determining which information should be retained or discarded in gating mechanisms [6]. The sigmoid activation function is articulated in equation (11).

$$\sigma{:}\mathbb{R} \rightarrow (0,1)$$

$$(x) = \frac{1}{1+e^{-x}} \quad (11)$$

The first derivative of the sigmoid activation function can be seen in equation (12).

$$\sigma'(x) = \sigma(x)(1-\sigma(x)) \quad (12)$$

The tanh activation function, or hyperbolic tangent, possesses a range of -1 to 1, yielding output values within this interval. The tanh activation function possesses a broader range, enabling it to effectively manage both positive and negative input [13]. The tanh activation function is articulated in equation (13).

2.7. Hyperparameters

Hyperparameters are external configurations that must be established prior to training [11]. Essential hyperparameters comprise the number of epochs, batch size, learning rate, and optimiser. The Adam (Adaptive Moment Estimation) optimiser is extensively utilised due to its integration of the benefits of AdaGrad and RMSProp, hence enhancing the training efficiency.

2.8. Data Pre-processing and Evaluation

Data normalisation, specifically Min-Max Normalisation, is performed to scale data within a certain range, such as [-1, 1], to improve model stability and expedite training pace [14]. This study employs Mean Squared Error (MSE) as the loss function during the training process to guide model optimization through minimization of prediction errors [15]. Mean Absolute Percentage Error (MAPE) is employed separately as the evaluation metric during the testing phase to assess the accuracy of the final model on unseen data. These two metrics serve distinct roles: MSE drives the training process, while MAPE provides an interpretable, scale-independent measure of forecasting accuracy for model comparison [16].

3. RESEARCH METHODS

This work utilises a quantitative methodology to predict the exchange rate of the Indonesian Rupiah relative to the Euro through deep learning techniques, specifically Long-Short Term Memory (LSTM) and Gated Recurrent Unit (GRU). The study utilised RStudio software for data processing and the development of forecasting models.

3.1. Data Source and Variables

This research use secondary data sourced from the official Bank Indonesia website (www.bi.go.id). The dataset comprises the daily exchange rate of the Rupiah relative to the Euro from January 2, 2018, to February 14, 2023, encompassing 1,252 data points. This study examines the daily closing price of the exchange rate.

3.2. Data Analysis

Procedure The analysis process involves several systematic steps, described as follows:

a. Data Input and Visualization: The data is imported into RStudio, followed by data visualization and descriptive statistical testing to understand the data characteristics.
b. Pre-processing: The dataset is first divided into training data (80%) and testing data (20%). Min-Max normalization is then applied by fitting the scaler exclusively on the training data, and subsequently transforming both training and testing sets using the training-derived parameters. This procedure ensures no data leakage from the testing set into the normalization process. The normalized data is subsequently transformed into a lagged dataset according to the specific timesteps used.
c. Data Splitting: The dataset is divided into two subsets prior to normalization: training data and testing data, with a proportion of 80% for training and 20% for testing. This ensures that normalization parameters are derived solely from the training set.

d. Model Configuration: This study adopts a univariate forecasting framework, meaning that only the historical closing exchange rate values are used as input features. Each model consists of a single recurrent layer (LSTM or GRU) followed by a dense output layer producing a single predicted value. The LSTM and GRU models are constructed with the following specific hyperparameters, determined prior to the learning process:
    i. Features: 1
    ii. Units: 1 and 50
    iii. Batch Sizes: 1, 32, and 64
    iv. Timesteps: 1 and 50
    v. Learning Rate: 0.001
    vi. Optimizer: Adam
    vii. Epochs: 1,000
    viii. Loss Function: Mean Squared Error (MSE) — used as the optimization objective during model training
    ix. Early Stopping: Triggered when the difference in loss value between the current epoch and the previous epoch is less than 0.000001.
    x. Training: The models are trained using the training dataset based on the defined architecture to learn the temporal patterns of the exchange rate.
    xi. Calculate the MSE to evaluate the model using the training data.
    xii. Test the LSTM and GRU models using the testing data.
    xiii. Denormalize the predicted data using the testing data.
    xiv. Calculate the MAPE to evaluate the predicted results using the testing data.
    xv. Select the best LSTM and GRU models based on the MSE and MAPE values.
    xvi. Draw conclusions.

4. RESULTS AND DISCUSSION

In the data preparation stage, the dataset was first split into training and testing sets (80%/20% ratio), and Min-Max Normalization was then applied using parameters derived solely from the training data to prevent data leakage. The normalization used the [-1,1] interval because LSTM and GRU modeling uses the tanh activation function, which compresses data to this range. The normalized data was subsequently converted into a lagged dataset prior to model construction. The time series data was transformed into a dataset with two variables: the independent variable (X) as the lagged variable and the dependent variable (Y) as the target. The number of X variables was determined based on the number of timesteps (1 and 50). For timestep 1, the training set contained 1,000 data points and the testing set contained 251 data points (total: 1,251; the one-observation discrepancy from the original 1,252 reflects the creation of one lagged observation). For timestep 50, the training set comprised 961 data points and the testing set comprised 241 data points, with the reduction accounting for the 50-period lag window.

The subsequent phase involved the development of a Long-Short-Term Memory (LSTM) model and a Gated Recurrent Unit (GRU) model. The optimal LSTM or GRU model was

identified by hyperparameter tuning, namely by developing many models utilising various combinations of the unit hyperparameter, batch size, and timestep. The LSTM and GRU models were subsequently trained utilising the previously established training data and hyperparameters. The model underwent training for 1,000 epochs, signifying it was trained 1,000 times until convergence was attained. The model training procedure was halted if the MSE value of the loss function from the preceding epoch to the present epoch was less than or equal to 0.000001. Subsequent to model training, the model underwent evaluation utilising the testing data. Table 1 presents the training and testing outcomes for all LSTM and GRU models, including assessment metrics such as the MSE from training and the MAPE from testing.

Table 1. Results of Training and Testing for LSTM and GRU Models

Model Name	Model Type	Unit	Batch Size	Timestep	Epoch	Training MSE	Testing MAPE
Model1.1.1	LSTM	1	1	1	85	2,705,733	3,272,611
Model1.1.50	LSTM	1	1	50	37	1,813,173	3,598,405
Model1.32.1	LSTM	1	32	1	522	1,738,949	2,953,895
Model1.32.50	LSTM	1	32	50	314	1,451,447	0.2321061%*
Model1.64.1	LSTM	1	64	1	771	1,847,914	3,198,559
Model1.64.50	LSTM	1	64	50	493	1,482,736	2,314,096
Model50.1.1	LSTM	50	1	1	294	2,020,418	4,451,178
Model50.1.50	LSTM	50	1	50	-	Failed to converge	-
Model50.32.1	LSTM	50	32	1	66	1,459,359	2,325,521
Model50.32.50	LSTM	50	32	50	107	1,344,096	3,223,758
Model50.64.1	LSTM	50	64	1	114	1,466,461	2,331,531
Model50.64.50	LSTM	50	64	50	123	1,445,421	3,462,715
Model1_1_1	GRU	1	1	1	69	2,403,372	3,240,596
Model1_1_50	GRU	1	1	50	39	2,016,613	3,289,181
Model1_32_1	GRU	1	32	1	469	1,503,593	2,321,311
Model1_32_50	GRU	1	32	50	211	1,518,126	2,628,067
Model1_64_1	GRU	1	64	1	597	1,671,959	2,720,061
Model1_64_50	GRU	1	64	50	317	157,408	2,712,309
Model50_1_1	GRU	50	1	1	340	2,254,626	4,802,171
Model50_1_50	GRU	50	1	50	-	Failed to converge	-
Model50_32_1	GRU	50	32	1	22	1,475,483	0.2412439%*
Model50_32_50	GRU	50	32	50	145	2,299,691	6,658,798
Model50_64_1	GRU	50	64	1	38	1,476,238	2,424,479
Model50_64_50	GRU	50	64	50	219	1,239,672	3,696,074

Table 1 highlights that variations in hyperparameters directly affect predictive accuracy and model convergence. The models configured with 50 units, a batch size of 1, and a timestep of 50 failed to converge within 1,000 epochs. This non-convergence can be attributed to multiple factors: the combination of a large timestep and high unit count increases model complexity and the number of trainable parameters, while an extremely small batch size of 1 introduces high gradient variance during parameter updates. This instability is consistent with known issues of gradient instability and poor generalization

in highly parameterized RNN architectures trained with small batches on limited data. It is important to note that the Testing MAPE values in Table 1 for models other than those marked with an asterisk (*) are reported as raw decimal fractions (e.g., 3,272,611 represents approximately 327.26%). All models except the two best-performing ones yielded MAPE values substantially higher than 100%, indicating poor forecasting performance. Only the two marked models achieved practically acceptable MAPE values below 1%. Therefore, determining the optimal model requires balancing hyperparameters and minimizing both MSE and MAPE values simultaneously. The analysis identified the top-performing LSTM architecture as having 1 unit, a timestep of 50, a batch size of 32, and 314 epochs, resulting in an MSE of 0.001451447 and a MAPE of 0.2321061%. In contrast, the superior GRU model achieved an MSE of 0.001475483 and a MAPE of 0.2412439% using 50 units, a timestep of 1, a batch size of 32, and 22 epochs.

Table 2. Comparison of Training and Testing Results of the Best Model

Model	Epoch	Training MSE	Testing MAPE
Model1.32.50	314	1,451,447	2,321,061
Model50_32_1	22	1,475,483	2,412,439

The LSTM model configured with 1 unit, a batch size of 32, a time step of 50, and 314 epochs proved to be the best-performing model for forecasting the Rupiah-Euro exchange rate. As shown in Table 2, the performance difference between the best LSTM and GRU models is marginal (MSE difference: 0.000024036; MAPE difference: 0.000091378), which suggests that both architectures are well-suited for this task. However, since the LSTM consistently recorded lower error values across both metrics, it is selected as the recommended model. It should be noted that while the models are compared based on their MSE and MAPE values, formal statistical significance testing (e.g., Diebold-Mariano test) was not conducted in this study, which represents a limitation. The practical implications of this finding suggest that deep learning models, particularly LSTM with appropriate hyperparameter tuning, can achieve highly accurate Rupiah-Euro exchange rate forecasts (MAPE < 0.25%), and thus can serve as a reliable reference tool for financial practitioners and policymakers monitoring currency fluctuations.

5. CONCLUSION

This study compared Long-Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) models for forecasting the daily Rupiah-Euro exchange rate using historical data from Bank Indonesia (January 2018 – February 2023). The optimal LSTM configuration (1 unit, timestep 50, batch size 32, 314 epochs) achieved an MSE of 0.001451447 and a MAPE of 0.2321061%, marginally outperforming the best GRU model (50 units, timestep 1, batch size 32, 22 epochs) with an MSE of 0.001475483 and a MAPE of 0.2412439%. Both models demonstrated excellent forecasting accuracy (MAPE < 0.25%), confirming the suitability of deep learning approaches for this task. The results indicate that a simpler LSTM architecture (1 unit) combined with a longer timestep (50) captures temporal

dependencies more effectively for this dataset than higher-complexity configurations. A key limitation of this study is the univariate framework, which does not incorporate macroeconomic variables such as interest rate differentials or trade balance data that may influence exchange rate movements. Additionally, formal statistical significance tests for model comparison were not conducted. Future research should explore multivariate models and extend the comparison to include traditional econometric benchmarks such as ARIMA. From a policy perspective, the high forecasting accuracy achieved by the LSTM model suggests it can serve as a practical tool for Bank Indonesia and financial institutions in monitoring and anticipating Rupiah-Euro exchange rate fluctuations, thereby supporting more informed monetary and trade policy decisions.

REFERENCES

[1] T. A. Surya, "Mewaspadai Ancaman Resesi Ekonomi Global," *Info Singkat*, vol. XIV, no. 19, 2022.

[2] I. Moridu *et al.*, *Manajemen Keuangan Internasional*. Bandung: Widina Bhakti Persada, 2021.

[3] A. Graves, "Generating Sequences with Recurrent Neural Networks," *arXiv*, 2013.

[4] K. Cho *et al.*, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," *arXiv*, 2014

[5] I. Simorangkir and Suseno, *Sistem dan Kebijakan Nilai Tukar*. Jakarta: Pusat Pendidikan dan Studi Kebanksentralan (PPSK) BI, 2004.

[6] C. C. Aggarwal, *Neural Networks and Deep Learning: A Textbook*. New York: Springer, 2018.

[7] S. Samarasinghe, *Neural Networks for Applied Sciences and Engineering*. Boca Raton: Auerbach Publications, 2006.

[8] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, 1997.

[9] A. A. Hamoud, A. Hoenig, and K. Roy, "Sentence subjectivity analysis of a political and ideological debate dataset using LSTM and BiLSTM with attention and GRU models," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, pp. 7974-7987, 2022.

[10] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," *arXiv*, 2014.

[11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016.

[12] S. Haykin, *Neural Networks and Learning Machines*, 3rd ed. Upper Saddle River, NJ: Pearson Prentice Hall, 2009.

[13] D. P. Kingma and J. L. Ba, "Adam: A Method for Stochastic Optimization," *arXiv*, 2015.

[14] D. Singh and B. Singh, "Investigating The Impact of Data Normalization on Classification Performance," *Applied Soft Computing Journal*, vol. 97b, 2020.

[15] S. G. Makridakis, S. C. Wheelwright, and R. J. Hyndman, *Forecasting: Methods and Applications*, 3rd ed. New York: John Wiley & Sons, 1997.

[16] R. F. Byrne, "Beyond Traditional Time-Series: Using Demand Sensing to Improve Forecasts in Volatile Times," *Journal of Business Forecasting*, vol. 31, p. 13, 2012.