Long Short-Term Memory Neural Network Model for Time Series Forecasting: Case Study of Forecasting IHSG during Covid-19 Outbreak

Raden Aurelius Andhika Viadinugroho, Dedi Rosadi*
Department of Mathematics, Gadjah Mada University, Indonesia

*E-mail: dedirosadi@gadjahmada.

Abstract. Long Short-Term Memory (LSTM) is one of the developments from Recurrent Neural Network (RNN) architecture. In this paper, we use LSTM architecture for modeling and forecasting the Indonesian Composite Stock Price Index (IHSG) closing value data. We also compare the performance of the LSTM method with the ARIMA and the Radial Basis Function (RBF) Neural Network method. In the implementation, we use both R and Python open source software. For empirical study we use the data from January until August 2020 to see the performance of the considered methods during Covid-19 pandemic periods of time. From the analysis, we found that LSTM performs better than ARIMA, but outperformed by RBF for this data.

1. Introduction
In time series modeling, we already know some classic method like Autoregressive (AR), Moving Average (MA), Autoregressive Moving Average (ARMA), and Autoregressive Integrated Moving Average (ARIMA). Besides that, also known model for heteroscedastic data such as Autoregressive Conditional Heteroscedasticity (ARCH) and Generalized Autoregressive Conditional Heteroscedasticity (GARCH) [36].

In order to assist researchers to implementing statistical method and analysis, machine learning algorithms are introduced, where machine learning learns how computers are learning (or improving model performance) based on data [18]. The advent of machine learning makes it easier for researchers to implement and develop these classical methods in the form of algorithms, so that its use in various fields is becoming increasingly widespread. However, the development of machine learning is also followed by the increase of complexity and volume data, where the data used in the analysis becomes increasingly large and the pattern of the data that we obtained becomes increasingly irregular.

Furthermore, these issues become a problem that occurs on time series modeling using classic methods like ARIMA. The greater the size of the data and the more irregular data patterns that formed, then the ARIMA model equation that formed will be more complicated and this will result in overfitting of the model, so the resulting performance of the model decreases. Therefore, deep learning are introduced to overcome those complexity data problem.

In deep learning, there is a neural network architecture that known as Recurrent Neural Network (RNN). RNN is a family of neural networks for processing data that have a form or nature of sequential data, such as text data and time series data. Then, an improvement of RNN – called Long Short-Term Memory (LSTM) [19] – are developed to overcome RNN problems, such as long memory problem and vanishing gradient problem.

The discovery of LSTM becomes a breakthrough for researchers to modeling sequential data, especially time series data. Apart from that, with the emergence of deep learning algorithms, gives motivation for researchers to develop methods for modeling time series using neural networks, start with using regular MLP to using Deep LSTM model (See [3], [11], [15], [29], [32], [35], and [37]).
In this paper, we build a LSTM model that specified for time series modelling by using IHSG data from January to August 2020 for the case study. The difference of this research from the previous researches are we compare the LSTM model performance with classic method like ARIMA and RBF neural network, and conduct the forecast beyond dataset using the LSTM, ARIMA, and RBF model, respectively.

2. Neural network architecture
In this section, LSTM architecture will be explained by giving the brief concept of RNN and LSTM, then the explanation of the hyperparameter that we used for the analysis.

One of the problem that the researchers encountered when using Multilayer Perceptron (MLP) architecture is the inability of MLP to read the sequence dependency on the sequence data. This become the motivation for the development of RNN [28]. Unlike MLP, the RNN architecture can read the sequence dependency on the sequence data, where the information flow on the previous state of hidden vector flows to the next state of hidden vector, which is illustrated on Figure 1.

![Figure 1 Recurrent Neural Network (RNN) architecture](image1)

![Figure 2 LSTM memory cell](image2)
The difference between LSTM and original RNN are located on the hidden state. While RNN using regular hidden state, LSTM using specific memory cell that illustrated on Figure 2 that contains more parameter and gating system unit that controls information flow. This gating system makes the LSTM able to overcome RNN weakness, such as long memory problem and vanishing gradient problem that suffered by original RNN, thus makes the LSTM performance better than RNN. The equation that LSTM memory cell uses can be written as recursive equation

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i)$$ \hspace{1cm} (3)

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f)$$ \hspace{1cm} (4)

$$c_t = f_t c_{t-1} + i_t \sigma(W_{xc}x_t + W_{hc}h_{t-1} + W_{cc}c_{t-1} + b_c)$$ \hspace{1cm} (5)

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o)$$ \hspace{1cm} (6)

$$h_t = o_t \tanh(c_t)$$ \hspace{1cm} (7)

where $\sigma$ denotes sigmoid function, $W$ denotes weight matrices, and $i_t, f_t, c_t, o_t, h_t$ are input gate, forget gate, state cell, output gate, and hidden vector on state $t$, respectively.

3. Model training

In this section, we will explain the hyperparameter that we used for training the model, with the loss function, optimizer, and the evaluation metrics as well.

The following are the hyperparameter that we use on the neural network model (LSTM and RBF) for this analysis.

| Parameters                        | LSTM  | RBF   |
|-----------------------------------|-------|-------|
| Input shape on data train and data test | (127, 1, 1) and (35, 1, 1) | (127, 1) and (35, 1) |
| Output shape on data train and data test | (127, 1) and (35, 1) | (127, 1) and (35, 1) |
| Hidden layers                     | 3     | 2     |
| LSTM Neurons                      | 50    | 50    |
| Epochs                            | 150   | 150   |
| Dropout usage                     | 0.1   | -     |
| Activation function               | ReLu  | Radial Basis Function |
| Learning rate                     | 0.01  | 0.01  |

Neural network model that we used in this paper will be trained using Adam method [23] for the optimizer, and the loss function that we use here are Mean Square Error (MSE), since this paper is about time series modeling problem. The evaluation metrics that we use to compare the model performance are Mean Absolute Percentage Error (MAPE), Mean Absolute Error (MAE), MSE, and Root Mean Square Error (RMSE), with the evaluation metric equations are follows.

$$MAPE = \frac{1}{N} \sum_{k=1}^{N} \left| \frac{y_k - \hat{y}_k}{y_k} \right| \times 100\%$$ \hspace{1cm} (8)

$$MAE = \frac{1}{N} \sum_{k=1}^{N} |y_k - \hat{y}_k|$$ \hspace{1cm} (9)

$$MSE = \frac{1}{N} \sum_{k=1}^{N} (y_k - \hat{y}_k)^2$$ \hspace{1cm} (10)

$$RMSE = \sqrt{\frac{1}{N} \sum_{k=1}^{N} (y_k - \hat{y}_k)^2}$$ \hspace{1cm} (11)
4. Case study: jakarta composite index (ihsg) from january to august 2020

In this section, we will explain the case study for this paper by constructing ARIMA, RBF, and LSTM model for Jakarta Composite Index (Indeks Harga Saham Gabungan, IHSG) data.

The data that we used on this paper are the IHSG closing price from January 2nd to August 19th, 2020 on business day [13]. Before we conduct the analysis, we conduct some data preprocessing by impute the missing value on the data by the data before the missing value occurs. This method are used because the nature of the stock price – it only changes on the trading time, so it assumes that the stock value when the missing value happen are same with the value prior to the missing value periods. The preprocessed data plot can be seen on Figure 3.

![IHSG stock value plot](image)

Figure 3 IHSG stock value plot

From the plot above, we can see that the data have an decreasing trend, then followed by increasing trend, and the pattern that generated on the data are irregular (i.e. the data pattern are nonlinear). By using ARCH-LM test (see on [1] and [2]) for nonlinear data, we can conclude that the IHSG data are nonlinear. Then, we split the data into training and testing data, where training data consists of the data from January 2nd to June 30th, 2020 and testing data consists of the data from July 1st to August 19th, 2020.

For the ARIMA model, here we test the stationarity of data with the Augmented Dickey-Fuller test (see on [26]), and based from the test we found that the data are stationer on second-order differencing. Then, from the correlogram of differenced data (see on Figure 4), we find out that the base model for ARIMA model creation are ARIMA(4,2,1) with the models underfitting, whether it uses mean constant or not. The ARIMA model underfitting can be seen on Table 2.

| Table 2. ARIMA model underfitting |
|-----------------------------------|
| ARIMA (4,2,1) with constant        | ARIMA (4,2,1) without constant  |
| ARIMA (4,2,0) with constant        | ARIMA (4,2,0) without constant  |
| ARIMA (3,2,1) with constant        | ARIMA (3,2,1) without constant  |
| ARIMA (3,2,0) with constant        | ARIMA (3,2,0) without constant  |
| ARIMA (2,2,1) with constant        | ARIMA (2,2,1) without constant  |
| ARIMA (2,2,0) with constant        | ARIMA (2,2,0) without constant  |
| ARIMA (1,2,1) with constant        | ARIMA (1,2,1) without constant  |
| ARIMA (1,2,0) with constant        | ARIMA (1,2,0) without constant  |
| ARIMA (0,2,1) with constant        | ARIMA (0,2,1) without constant  |
From the ARIMA model, we found that the best model are ARIMA(1,2,1) without constant, and if we see from the model’s forecast on test data (see on Figure 5 and 6), we found that the forecast result are very different compared with the actual data. In other words, the ARIMA model suffers overfitting.

Next, we create the neural network model using LSTM and RBF architecture. Before we conduct the model creation using neural network, first we transform the data using min-max transformation using the following equation

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

(12)
Where $z$ denotes the transformed data, $x$, $\min(x)$, and $\max(x)$ denotes the original data, minimum value of original data, and maximum value of original data, respectively. After we obtain the predicted value from model, then we conduct the inverse transforming to revert it to the original value using the following equation.

$$x = z(\max(x) - \min(x)) + \min(x)$$  \hspace{1cm} (13)

Next, we change the data format into supervised learning format – consists of input value and output value – by observing the correlogram of the data. Based on the correlogram on Figure 7, we find out that the PACF lag are significant on first lag. Therefore, the input value that we use for neural network model are the IHSG data from $(t - 1)$-th period, and the output value are from $t$-th period.

![Figure 7 The correlogram of IHSG original data](image)

For the LSTM model, we reshape the input data into [sample, timestamps, features] format, since the LSTM on Tensorflow library only accepts 3-dimensional inputs. The following are the training result, comparison of actual and forecast in data train, and comparison of actual and forecast in data train from LSTM, respectively.

![Figure 8 Train and test loss from LSTM model training](image)
From the training result, we can see that the LSTM model generate the decreasing loss values – both on training and validation data – over the epochs. This means that the LSTM model performs well in the training process using the data that we use in this analysis.

![Forecast Plot in Training Data (LSTM)](image1)

**Figure 9** Comparison of actual and forecast data on training data (LSTM)

![Forecast Plot in Testing Data (LSTM)](image2)

**Figure 10** Comparison of actual and forecast data on testing data (LSTM)

From the actual and forecast data comparison on Figure 9 and 10, by exploratory we can see that the LSTM model forecast pattern are pretty close with the actual data (both on training and testing data). Therefore, by exploratory means, we can conclude that LSTM performance are better than ARIMA.

The following are the comparison of actual and forecast in data train and comparison of actual and forecast in data train from RBF, respectively.
From the actual and forecast data comparison on Figure 11 and 12, by exploratory we can see that the RBF model forecast pattern are closer with the actual data (both on training and testing data) than LSTM. Therefore, by exploratory means, we can conclude that RBF performance are better than ARIMA and LSTM. To prove it, we compare the evaluation metrics on the testing dataset from each model as follows.

**Table 3.** Evaluation metrics of each model on testing data

| Model  | MAPE      | MAE       | MSE        | RMSE     |
|--------|-----------|-----------|------------|----------|
| ARIMA  | 3.806265% | 194.6707  | 44004.7    | 209.7730 |
| LSTM   | 2.901452% | 146.9627  | 26271.87   | 162.086  |
| RBF    | 0.7582347%| 38.61302  | 2762.742   | 52.56179 |
From the evaluation metrics comparison on Table 3, we can see that the LSTM model generates lower evaluation metrics score – all from MAPE, MAE, MSE, and RMSE than ARIMA model, but higher than RBF model. This means that LSTM model performs better than ARIMA, but still outperformed by RBF. This is happen because of the lack of the observation in the data and the data pattern in training data and testing data that are very different – especially on the data trend, thus makes the LSTM overfits the training data.

The difference between the previous researches with this research, here we conduct forecast beyond dataset for every model in 30-day period, to find out the models performance in forecast process. The result of forecast beyond dataset are as follows.

**Table 4.** Forecast beyond dataset in 30-day period for IHSG data

| Date       | ARIMA  | LSTM   | RBF    |
|------------|--------|--------|--------|
| 20-Aug-20  | 4877.644 | 5298.538 | 5324.871 |
| 21-Aug-20  | 4876.754 | 5305.227 | 5379.951 |
| 24-Aug-20  | 4875.865 | 5306.97  | 5437.991 |
| 25-Aug-20  | 4874.975 | 5307.433 | 5498.607 |
| 26-Aug-20  | 4874.085 | 5307.543 | 5561.502 |
| 27-Aug-20  | 4873.196 | 5307.574 | 5626.367 |
| 28-Aug-20  | 4872.306 | 5307.585 | 5692.736 |
| 31-Aug-20  | 4871.417 | 5307.585 | 5760.048 |
| 01-Sep-20  | 4870.527 | 5307.585 | 5827.777 |
| 02-Sep-20  | 4869.637 | 5307.585 | 5895.356 |
| 03-Sep-20  | 4868.748 | 5307.585 | 5962.159 |
| 04-Sep-20  | 4867.858 | 5307.585 | 6027.643 |
| 07-Sep-20  | 4866.968 | 5307.585 | 6091.174 |
| 08-Sep-20  | 4866.079 | 5307.585 | 6152.359 |
| 09-Sep-20  | 4865.189 | 5307.585 | 6210.768 |
| 10-Sep-20  | 4864.299 | 5307.585 | 6266.057 |
| 11-Sep-20  | 4863.411 | 5307.585 | 6317.899 |
| 14-Sep-20  | 4862.522 | 5307.585 | 6366.166 |
| 15-Sep-20  | 4861.631 | 5307.585 | 6410.696 |
| 16-Sep-20  | 4860.741 | 5307.585 | 6451.634 |
| 17-Sep-20  | 4859.851 | 5307.585 | 6488.934 |
| 18-Sep-20  | 4858.962 | 5307.585 | 6522.757 |
| 21-Sep-20  | 4858.072 | 5307.585 | 6553.182 |
| 22-Sep-20  | 4857.182 | 5307.585 | 6580.474 |
| 23-Sep-20  | 4856.293 | 5307.585 | 6604.846 |
| 24-Sep-20  | 4855.403 | 5307.585 | 6626.475 |
| 25-Sep-20  | 4854.513 | 5307.585 | 6645.614 |
| 28-Sep-20  | 4853.624 | 5307.585 | 6662.494 |
| 29-Sep-20  | 4852.734 | 5307.585 | 6677.328 |
| 30-Sep-20  | 4851.845 | 5307.585 | 6690.369 |
5. Conclusions
From the analysis that we conduct on the previous section, we find out that LSTM model generates lower evaluation metrics score – all from MAPE, MAE, MSE, and RMSE than ARIMA model, but higher than RBF model. This means that LSTM model performs better than ARIMA, but still outperformed by RBF. This is happen because of the lack of the observation in the data and the data pattern in training data and testing data that are very different – especially on the data trend, thus makes the LSTM overfits the training data – although, it still can prevent overfitting that suffered by ARIMA model. This means that the LSTM model performs better in greater dataset with the same trend on training data and testing data.

The suggestion that we can give from this analysis for future research are we would like to create RNN model using complex number input value since many of RNN implementation are using real number input value.

References
[1] Aitchison, J. and Silvey, S. 1958. Maximum-likelihood Estimation of Parameters Subject to Restraints, *Annals of Mathematical Statistics*, vol. 29, pp. 813-828.
[2] Aitchison, J. and Silvey, S. 1960. Maximum-likelihood Estimation Procedures and Associated Tests of Significance, *Journal of the Royal Statistical Society*, vol. 22, pp. 154-171.
[3] Aizenberg, I., Sheremetov, L., Villa-Vargas, L., and Martinez-Muñoz, J. (2016). Multilayer Neural Network with Multi-Valued Neurons in time series forecasting of oil production, *Neurocomputing*, vol. 175, pp. 980–989.
[4] Aggarwal, C. C. 2018. Neural Networks and Deep Learning. Springer.
[5] Anton, H. and Rorres, C. 2014. Elementary Linear Algebra. 11th ed. Wiley.
[6] Bain, L. J. and Engelhardt, M. 1992. Introduction to Probability and Mathematical Statistics. 2nd ed. Duxbury Press.
[7] Bengio, Y., Simard, P., and Frasconi, P. 1994. Learning long-term dependencies with gradient descent is difficult, *IEEE Transactions on Neural Network*, vol. 5, no. 2, pp. 157-166.
[8] Bishop, C. M. 2006. Pattern Recognition and Machine Learning. Springer.
[9] Brownlee, J. 2017. Long Short-Term Memory Networks With Python. Jason Brownlee: Melbourne.
[10] Cleveland, R. B., Cleveland, W. S., McRae, J. E., and Terpenning, I. 1990. STL: A Seasonal-Trend Decomposition Procedure Based on Loess, *Journal of Official Statistics*, vol. 6, no. 1, pp. 3-73.
[11] Davey, N., Hunt, S., and Frank, R. 1999. Time Series Prediction and Neural Networks, *Procs 5th Int Conf on Engineering Applications of Neural Networks (EANN’99)*, pp. 93-98.
[12] Duchi, J., Hazan, E., and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization, *The Journal of Machine Learning Research*, vol. 12, pp. 2121-2159.
[13] Finance.yahoo.com. 2020. Yahoo! Finance. [online] Available at: https://finance.yahoo.com/quote/%5EJKSE?p=%5EJKSE [Accessed 5 Feb. 2020].
[14] Finance.yahoo.com. 2020. Yahoo! Finance. [online] Available at: https://finance.yahoo.com/quote/USDIDR%3DX/history?p=USDIDR%3DX [Accessed 3 Mar. 2020].
[15] Firmino, P. R. A., de Mattos Neto, P. S. G., and Ferreira, T. A. E. 2014. Correcting and combining time series forecasters, *Neural Networks*, vol. 50, pp. 1-11.
[16] Goodfellow, I., Bengio, Y., and Courville, A. 2016. Deep Learning. Cambridge: MIT Press.
[17] Graves, A., Jaitly, N., and Mohamed, A. 2013. Hybrid speech recognition with Deep Bidirectional LSTM, *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pp. 273-278.
[18] Han, J., Kamber, M., and Pei, J. 2012. Data Mining Concepts and Techniques. 3rd ed. Boston: Elsevier.
[19] Hochreiter, S. and Schmidhuber, J. 1997. Long short-term memory, *Neural Computation*, pp. 1735-1780.
[20] Hogg, R. V., McKean, J., and Craig, A. T. 2014. Introduction to Mathematical Statistics. 7th ed. New York: Pearson Education.
[21] Hosseini, MP., Lu, S., Kamaraj, K., Slowikowski, A., and Venkatesh, H. C. 2020. "Deep Learning Architectures” in Pedrycz, W. and Chen, SM. (eds) Deep Learning: Concepts and Architectures. Studies in Computational Intelligence. vol. 866, pp. 1-23. Springer.
[22] Kelleher, J. D. 2019. Deep Learning. Cambridge: MIT Press.
[23] Kingma, D. P. and Ba, J. 2015. A method for stochastic optimization, *Proceedings of 3rd International Conference on Learning Representations*.
[24] McCulloch, W. S. and Pitts, W. H. 1943. A logical calculus of the ideas immanent in nervous activity.
Bulletin of Mathematical Biophysics, vol. 5, pp. 115-133.

[25] Picard, R. R. and Berk, K. N. 1990. Data Splitting, The American Statistician, vol. 44, no. 2, pp. 140-147.

[26] Rosadi, D. 2011. Analisis Ekonometrika dan Runut Waktu Terapan dengan R. Yogyakarta: Andi Offset.

[27] Rosadi, D. 2014. Analisis Runut Waktu dan Aplikasinya dengan R. Yogyakarta: UGM Press.

[28] Rumelhart, D., Hinton, G., and Willams, R. 1986. Learning representations by back-propagating errors, Nature, vol. 323, pp. 533–536.

[29] Sagheer, A. and Kotb, M. 2019. Time series forecasting of petroleum production using deep LSTM recurrent networks, Neurocomputing, vol. 323, pp. 203-213.

[30] Schuster, M. and Paliwal, K. K. 1992. Bidirectional Recurrent Neural Networks, IEEE Transactions On Signal Processing, vol. 45, no. 11, pp. 2673-2681.

[31] Schmidhuber, J. 1992. Learning complex, extended sequences using the principle of history compression, Neural Computation, vol. 4, no. 2, pp. 234-242.

[32] Sedki, A., Ouazar, D., and El Mazoudi, E. (2009). Evolving neural network using real coded genetic algorithm for daily rainfall–runoff forecasting, Expert Systems with Applications, vol. 36, pp. 4523–4527.

[33] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting, Journal of Machine Learning Research, vol. 15, pp. 1929-1958.

[34] Subanar. 2013. Statistika Matematika: Probabilitas, Distribusi, dan Asimtotis dalam Statistika. Yogyakarta: Graha Ilmu.

[35] Theodosiou, M. 2011. Disaggregation & aggregation of time series components: A hybrid forecasting approach using generalized regression neural networks and the theta method, Neurocomputing, vol. 74, pp. 896–905.

[36] Wei, W. W. S. 2006. Time Series Analysis Univariate and Multivariate Methods. Boston: Pearson Education.

[37] Xiangxue, W., Lunhui, X., and Kaixun, X. 2019. Data-Driven Short-Term Forecasting for Urban Road Network Traffic Based on Data Processing and LSTMRNN, Arabian Journal for Science and Engineering, vol. 44, pp. 3043-3060.

[38] Zhu, W., Lan, C., Xing, J., Zeng, W., Li, Y., Shen, L., and Xie, X. 2016. Co-Occurrence Feature Learning for Skeleton Based Action Recognition Using Regularized Deep LSTM Networks, Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16), pp. 3697-3703.