Contemporary Mathematics
http://ojs.wiserpub.com/index.php/CM/

Research Article

Algorithm Optimizer in GA-LSTM for Stock Price Forecasting

YL Sukestiyarno[1*], Dian Tri Wiyanti[1], Lathifatul Azizah[1], Wahyu Widada[2], Khathibul Umam Zaid Nugroho[3]

[1]Universitas Negeri Semarang, Kelud Utara III Petompon Gajahmungkur Semarang 50229, Indonesia
[2]Bengkulu University, Jl. Kd. Limun, Bengkulu 38371, Indonesia
[3]Universitas PGRI Silampari, Jl. M. Toha, Lubuk Linggau 31625, Indonesia
E-mail: ylsukestiyarno@gmail.com

Received: 11 July 2023; **Revised:** 18 August 2023; **Accepted:** 14 September 2023

Abstract: The training and success of deep learning is strongly influenced by the selection of hyperparameters. This research uses a hybrid method between the genetic algorithm (GA) and long short-term memory (LSTM) to find a suitable model for predicting stock prices. GA is used to optimize the architecture, such as the number of epochs, window size, and LSTM units in the hidden layer. Tuning optimizer is also carried out using several optimizers to achieve the best value. The method that has been applied shows that the method has a good level of accuracy with mean absolute percentage error (MAPE) values below 10% in every optimizer used. A fairly stable and small value is generated by setting it using the Adam optimizer.

***Keywords*:** time series, forecasting, deep learning, GA, LSTM

MSC: 90C27, 90C31

Nomenclature

ANN	Artificial neural networks
ARIMA	Autoregressive integrated moving average
DL	Deep learning
GA	Genetic algorithm
LSTM	Long short-term memory
MAPE	Mean absolute percentage error
RMSE	Root mean square error
RNN	Recurrent neural network

1. Introduction

Public investment awareness is increasing and investment instruments that are currently quite attractive to the public are stocks. It is impossible to know for sure what the future will be like. The existence of supply and demand for

DOI: https://doi.org/10.37256/cm.5120243367

shares in the capital market makes stock prices fluctuate [1] which can make it difficult for investors to see investment opportunities in a company's shares. One tool that can help investors overcome this is represented by forecasting techniques [2]. Time series forecasting is applicable in many real-world problems, such as stock price forecasting, language processing, or weather forecasting that directly or indirectly affect human life [3].

Time series data that is available in large quantities can be converted into information that will be used for forecasting [4]. Forecasting is used to predict what might happen in the future [5]. The process of data forecasting can be simplified and accelerated with the help of the latest breakthroughs in computer technology. In artificial intelligence, time series data is just one area where machine learning has significantly improved. It can be said that deep learning uses artificial neural networks because it is a machine learning technique that mimics the neural network architecture of the human brain. According to [6], artificial neural networks (ANNs) are connected networks that study and transmit information from one artificial neuron to another, taking inspiration from biological neurons. According to [7], ANNs found to be a useful model as an information manager that has a similar function to the biological nervous system of the human brain that can be applied to problem-solving. Deep representation learning, often known as deep learning (DL), is the process of studying a hierarchy of representations or characteristics as inputs move between neurons. The DL approach learns the input to produce higher performance accuracy [8].

Nonlinear systems are ideal for the time series data method because they contain dynamic data with broad dimensions. The type of DL based on nonlinear predictions is a recurrent neural network [9]. One DL technique that can be used for time series is the recurrent neural network (RNN), which is designed to work with sequential data [10, 11]. The progress of RNN is growing quite rapidly in various fields, but RNN has a weakness in processing time series because the performance for prediction will have a negative effect if the sequence size is relevantly long and the other is that the RNN gradient will be lost, resulting in long-term memory failure [12]. Unlike the RNN, long short-term memory (LSTM) can manage memory for each input by using memory cells and gate units [13].

DL's training and success are strongly influenced by the selection of hyperparameters. Hyperparameters are the variables whose values are manually assigned to the model to assist in learning [14]. Research conducted in [15] succeeded in finding hyperparameters in the form of window size and hidden layer arrangement through the implementation of a genetic algorithm (GA) and LSTM in finding the best model. GA is a heuristic search and optimization technique that imitates the evaluation process [16]. Optimization itself is a process of solving certain problems in order to be in the most favorable condition [17]. GA is widely used to find optimal approximation solutions for optimization problems with large search space. Many studies have used a general approach with trial-and-error testing scenarios. By using GA, a systematic approach is applied to get the best solution and optimize predictions.

The model won't be able to capture the patterns of training instances if there aren't enough training epochs. Additionally, if the epoch number is too high, the model will overfit [18]. Finding a good epoch number is therefore essential to create a model with great performance. This research uses a hybrid method between the GA and LSTM to find a suitable model for predicting stock prices. GA is used in optimizing architectures such as window sizes, and the number of LSTM units in hidden layers and considering the number of epochs to find out the application of the GA-LSTM method in predicting stock prices. The selection of the right algorithm optimizers will produce the best value for each parameter model. Because optimizers are used to find the best value [19], therefore, an optimization setting will also be carried out using several optimizers to get the best value. GAs are used in optimizing architectures such as window sizes, and the number of LSTM units in hidden layers and considering the number of epochs to find out the application of the GA-LSTM method in predicting stock prices. The optimizer aims to find optimal weights, minimize errors and maximize accuracy [19]. In [20], the authors argued that the most widely used optimizer in DL is mini-batch. From the research that has been done, many use one optimizer subjectively, so research is also carried out to find the most optimal optimizer using a mini-batch optimizer.

In [21], the authors extract historical monthly financial time data from January 1985 to August 2018 from the Yahoo Finance website. In assessing the effectiveness of two techniques for forecasting time series data, namely LSTM and autoregressive integrated moving average (ARIMA), these two strategies were used together with a set of financial data, and the results showed that LSTM outperformed ARIMA. The LSTM-based algorithm specifically improves predictions by 85% on average more than ARIMA.

The authors of [22] compared the performance of the two techniques for predicting financial time series. After creating a stationary series, the authors apply the ARIMA model using different settings, the best model to maintain is random walk ARIMA (0, 1, 0). Then, the authors developed an LSTM architecture based on different parameter settings, the best configuration is two LSTM blocks contained in the hidden layer. To evaluate the model, root mean squared error (RMSE), mean absolute error (MAE), and mean squared error (MSE) calculations are used. Where LSTM shows small value results in all three calculations, thus confirming that LSTM has good performance in predicting more accurate values for the time series studied. [23] Implemented the LSTM-RNN method with historical stock data of AAPL (Apple Inc.), GOOG (Google) and TSLA (Tesla, Inc.) with Adam optimization. Regression, support vector machine (SVM), random forest (RF), feed forward neural network (FFNN), and backpropagation are some examples of traditional machine learning algorithms that have been used to compare models. The findings show that when compared to conventional machine learning methods, the RNN-LSTM model tends to give more accurate results. [24] Uses the LSTM approach to predict the time series Bank BRI shares, with the selection of 9 epochs resulting in an RMSE of 227.470333244533 which is considered quite good and visually shows a prediction graph that is almost identical to the original data.

Research on the use of two secondary data, namely stock index data and the USD to IDR exchange rate to make stock price forecasts in Indonesia using data from 09 June 2019 to 06 June 2019 with the LSTM method which produces testing under LSTM can predict stock prices from 2017 to 2019 well, shown through the results of error, so that conclusions can be drawn with accurate results, LSTM can estimate stock prices and can overcome long-term dependencies [13].

The research conducted in [25] on the implementation of LSTM on stock prices of three plantation companies in Indonesia resulted in the best LSTM for SSMS shares was 70, which resulted in RMSE 21,328 using hidden neurons and the RMSProp optimizer option. Then, the best LSTM model is the stock LSIP, which results in an RMSE score of 33.097 with Adam and hidden neurons set to a maximum of 80 in the optimizer. The best model is the SIMP stock, which when used with the Adamax optimizer setting and 100 hidden neurons, results in an RMSE score of 8.337.

Research conducted in [17] on the optimization of ANNs with GAs used to predict credit card approval by applying the neural network obtained an increase in results from 85.42% to 87.82%. Then, there is a study in predicting KOSPI stock prices by applying the use of GA-LSTM to produce the best LSTM model by setting a window size 10 and compiling 2 hidden layers with nodes of 15 and 7 respectively, the MSE and MAE were 181.99% and 10.21%, respectively [15].

[26] uses GA to choose the optimal time and number of hidden units with LSTM for prediction, showing that GA-LSTM works better than the RNN model which is not optimized in prediction accuracy.

Research conducted in [27] integrated GA with LSTM to find the optimal hyperparameter configuration for LSTM. By using GA, focus on optimizing architectural aspects to model optimal networks based on predictive accuracy. From the research that has been done, it shows that LSTM is a good forecasting method compared to traditional methods.

2. Materials and methods

In this study, the GA-LSTM method is applied to forecast the stock price of PT Bank Rakyat Indonesia (Persero) Tbk and PT Bank Mandiri (Persero) Tbk. The data collected and used in this study is the stock price dataset of Bank BRI [28] and Bank Mandiri [29] which was taken on February 3, 2022 in the form of CSV (Comma Separated Values) format obtained from the website www.finance.yahoo.com. The data used is close price as much as 2,487 price data which is converted into a format that can be used in Python using the Pandas module.

It is then checked whether there is missing or incomplete data, so that the missing data does not affect the overall data processing. Then, it is necessary to transform the data and normalize the data before using it. The data transformation is carried out to make the data stationary where the data to be used does not have a tendency to a certain trend. For the normalization in this study, we use the implementation of the transformation object from the -learn class scikit. Furthermore, the data is divided into training and the remaining 20% data for testing data. This study uses an LSTM design with 3 neural and a dropout module for each layer, uses a loss function MSE and uses several optimizers with the TensorFlow library (Figure 1). Figure 2 shows the GA-LSTM model flowchart.

LSTM has four components, namely input gate, forget gate, cell state, and output gate [30]. Input gate has two functions; to receive new information: *rt* and *dt*. *rt* prearranging hidden vectors h_{t-1} with new information x_t. That is, [h_{t-1}, x_t] , then multiplied by the weight matrix W_r , after that plus the noise vector b_r. D_t does the same. Then multiply r_t and

d_t by element-wise to get the cell state c_t.(2)

$$r_t = \sigma\left(W_r \cdot \left[h_{t-1}, x_t\right] + b_r\right), \tag{1}$$

$$d_t = \tanh(W_d \cdot [h_{t-1}, x_t] + b_d). \tag{2}$$

Forget gate *ft* looks like that similar to *rt* in input gate. This gate controls the limit until the value is stored in memory

$$f_t = \sigma\left(W_f \cdot \left[h_{t-1}, x_t\right] + b_f\right). \tag{3}$$

Cell state is used for multiplication calculation based on the element between the previous cell state C_{t-1}, and forget gate *ft* then added the result of input gate *rt* and *dt*.

$$C_t = f_t . C_{t-1} + r_t . d_t . \tag{4}$$

Here, O_t output gate in time t, and W_o and b_o are weights line and bias for gate output, respectively. h_t is the hidden layer that will go to in the next step, or up to output as applying y_t obtained by tanh to h_t. Note that the output o_t is not the output of y_t, but the gate used to control the output.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \tag{5}$$

$$h_t = o_t \tanh C_t . \tag{6}$$

In [20], the authors argued that the optimizer that is most widely used in DL is the mini-batch consisting of Adagrad, Adadelta, RMSprop, Nadam, and Adam.

In Adagrad deployments, the learning rate is normalized for each dimension on which the cost function depends. The learning rate in each iteration is the learning rate divided by the norm *l*2 of the gradient the previous to the current iteration for each dimension. The formula used in Adagrad is as follows:

$$S_t = S_{t-1} + \left[\frac{\partial L}{\delta wt}\right]^2 \tag{7}$$

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{S_t + \epsilon}} \cdot \frac{\partial L}{\delta w_t} . \tag{8}$$

Description:
t = for time step
w = weight/ parameter to be updated
α = learning rate (0.001)
$\frac{\partial L}{\delta w}$ = gradient L (loss function)
S = cumulative sum of squares of gradient current and

Previous is an Adagrad extension as an alternative to reduce Adagrad's aggressiveness, reduce the learning rate monotonically, also focus more on the learning rate. The formula used is as follows:

$$D_t = \beta D_{t-1} + (1-\beta)[\Delta w_t]^2 . \tag{9}$$

$$S_t = \beta S_{t-1} + (1-\beta)\left[\frac{\partial L}{\partial wt}\right]^2. \tag{10}$$

$$w_{t+1} = w_t - \frac{\sqrt{D_{t-1} + \epsilon}}{\sqrt{S_t + \epsilon}} \cdot \frac{\partial L}{\partial w_t}. \tag{11}$$

Description:
D = difference between the current weight and the updated weight
$\beta = 0.9$
ϵ = 1e-7
RMSprop is learning rate which is an improvement on Adagrad. The formula used is as follows:

$$S_t = \beta S_{t-1} + (1-\beta)\left[\frac{\partial L}{\partial wt}\right]^2, \tag{12}$$

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{S_t + \epsilon}} \cdot \frac{\partial L}{\partial w_t}. \tag{13}$$

RMSprop and AdaGrad are combined with momentum to form Adam. Adam measures learning rates via a quadratic gradient, similar to RMSprop, and uses dynamic gradient averaging to take advantage of momentum [30]. The formula used is as follows:

$$\hat{V}_t = \frac{V_t}{1-\beta_1^t}, \tag{14}$$

$$\hat{S}_t = \frac{S_t}{1-\beta_2^t}, \tag{15}$$

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{\hat{S}_t + \epsilon}} \cdot \hat{V}_t. \tag{16}$$

Description:
$\hat{V}$ = average gradient with momentum replacing gradient current
$\hat{S}$ = average cumulative sum of squares of gradients current and previous
$\beta_1 = 0.9$
$\beta_2 = 0.999$
ϵ = 1e-7
Nadam is used for noisy gradient or gradient with high curvature. The learning process is accelerated by adding up the decay of the moving average for the previous and current gradients, Nadam takes gradients one step further by using Nestrove to replace $\hat{V}$ in the previous equation with $\hat{V}$. The current formula used is as follows:

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{\hat{S}_t + \epsilon}}\left(\beta_1 \hat{V}_t + \frac{1-\beta_1}{1-\beta_1^t} \cdot \frac{\partial L}{\partial w_t}\right), \tag{17}$$

where

$$\hat{V}_t = \frac{V_t}{1-\beta_1^t};\ \hat{S}_t = \frac{S_t}{1-\beta_2^t}. \tag{14}$$

Description:
α = learning rate (0.002)
$\beta_1 = 0.9$
$\beta_2 = 0.999$
ϵ = 1e-7

```
model.compile (loss = 'mean_squared_error', optimizer = tf.keras.optimizers
.Adagrad()
```

```
model.compile (loss = 'mean_squared_error', optimizer = tf.keras.optimizers
.Adadelta()
```

```
model.compile (loss = 'mean_squared_error', optimizer = tf.keras.optimizers
.RMSprop()
```

```
model.compile (loss = 'mean_squared_error', optimizer = tf.keras.optimizers
.Adam()
```

```
model.compile (loss = 'mean_squared_error', optimizer = tf.keras.optimizers
.Adagrad()
```

Figure 1. Optimizers with the TensorFlow library

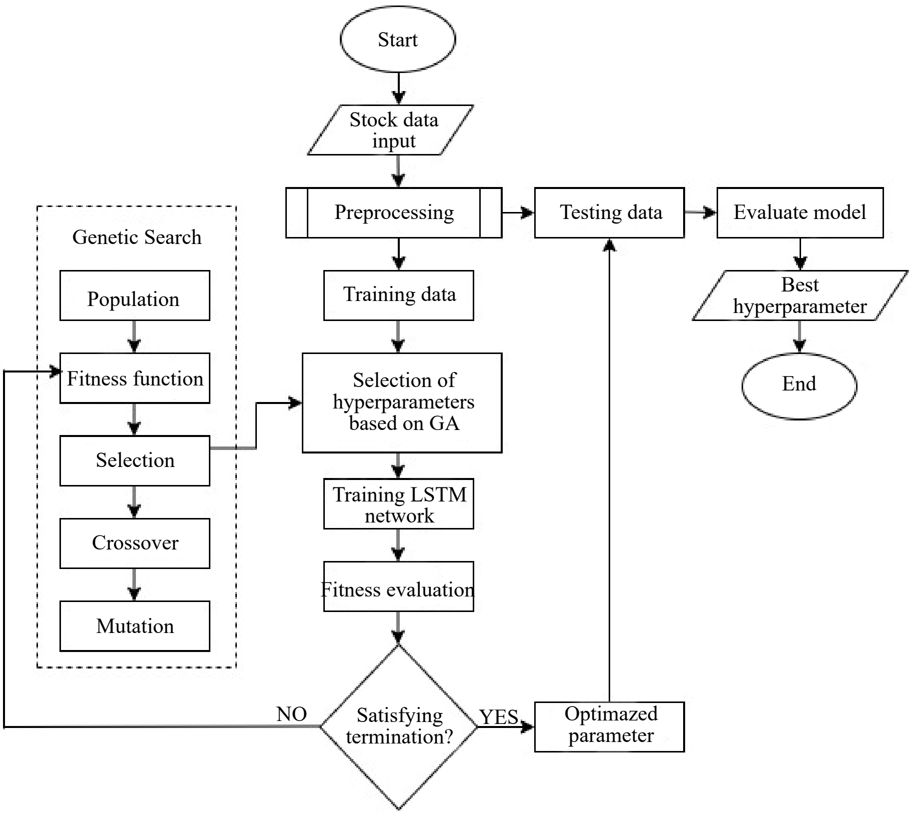

Figure 2. GA-LSTM model flowchart

While for the initialization design of the GA using DEAP library from Python, the initial stage is carried out by determining the initial population which is a collection of chromosomes containing solutions for the number of window sizes, epochs, and the number of units. The formation of chromosomes is done in binary using binary numbers. The basic structure of the GA consists of several steps [17], namely: 1) initialization of the population; 2) population evaluation; 3) selection of the population to be subjected to genetic operators; 4) the process of crossover of certain chromosome pairs; 5) certain chromosomal mutation processes; 6) evaluation of the new population; 7) repeat from step 3 if the stop condition is not met.

In this study, we used: population = 5, maximum generation = 10, crossover rate = 0.4, mutation rate = 0.1, operator initialization = initRepeat, operator crossover = cxTwoPoint, operator mutation = mutUniformInt, and operator selection = selRoulette. Figure 3 shows the operators of GA. Each design is carried out to evaluate the suitability of the GA. The GA process is repeated more than once by setting different values for the number of window sizes, epochs, and number of units.

```
toolbox.register ('population', tools.initRepeat, list , toolbox.individual)
toolbox.register ('mate', tools.cxTwoPoint)
toolbox.register ('mutate', tools.mutUniformInt, low = [5, 1, 1, 1, 1], up = [50, 5, 5, 5, 6],
indpb = 0.6)
toolbox.register ('select', tools.selRoulette)
```

Figure 3. Operators of GA

3. Results

The research uses the GA-LSTM method as a calculation process by applying several different optimizations to find hyperparameters of the number of epochs, window sizes, and the number of LSTM units in the hidden layer. This study is limited to using close price for the prediction. The results of this study are the model with the hyperparameters obtained from training and testing data with the lowest MAE and mean absolute percentage error (MAPE) values, then the obtained model can be used to predict the stock's price.

RMSE is used to measure the difference between the estimated target and the actual target by calculating the square root value of the MSE. The higher the value produced by the RMSE, the lower the level of accuracy, and vice versa, if the value of the resulting RMSE is lower, the level of accuracy is higher [31]. The RMSE formula is shown in the following equation.

$$\text{RMSE} = \sqrt{\sum_{i=1}^{n} \frac{\left(\hat{y}_i - y_i\right)^2}{n}} \tag{19}$$

Description:

y_i = value of the i

$\hat{y}_i$ = result forecast

n = amount of data

MAPE is used to measure error by calculating the average method the average absolute error divided by the true value, which results show the absolute percentage error value of the predicted model results. The prediction model is getting better if the MAPE value is lower [32]. Table 1 shows the interpretation of typical MAPE values. The MAPE formula is shown in the following equation.

$$\text{MAPE} = \frac{100\%}{n} \sum \left| \frac{y - \hat{y}}{y} \right| \tag{20}$$

Description:

$\hat{y}$ = value of forecast results
y = value of observation to i
n = amount of data

Table 1. Range MAPE value [32]

Range of MAPE	Meaning
< 10%	The accuracy rate is very good
10-20%	The accuracy rate is good
20-50%	The accuracy rate is decent
> 50%	The accuracy rate is bad

After doing all stages of the research, the output is in the form of RMSE and MAPE values, as well as a graph of the comparison of the original price with the predicted data, the results are shown.

3.1 *Case 1: shares of bank BRI*

Table 2. Forecasting results for Case 1

Optimizers	Epochs	Neurons	Window size	Data	RMSE	MAPE(%)
Adagrad	33	[4, 4, 3]	1	Training	100.58	1.49
				Testing	94.43	1.86
	38	[4, 4, 3]	1	Training	108.31	1.53
				Testing	95.39	1.88
Adadelta	35	[1, 3, 2]	1	Training	106.43	1.51
				Testing	94.47	1.86
	18	[2, 1, 4]	2	Training	99.69	1.58
				Testing	98.09	1.91
RMSprop	41	[4, 5, 5]	1	Training	102.91	1.52
				Testing	94.40	1.88
	43	[3, 1, 4]	5	Training	102.83	1.53
				Testing	95.49	1.91
Adam	38	[4, 5, 2]	4	Training	94.28	1.51
				Testing	95.49	1.87
	24	[4, 5, 2]	2	Training	95.93	1.51
				Testing	95.06	1.87
Nadam	15	[3, 3, 2]	6	Training	93.03	1.51
				Testing	95.62	1.87
	28	[4, 5, 4]	2	Training	95.99	1.50
				Testing	95.17	1.87

The forecasting results for Case 1 is shown in Table 2. In the model using Adagrad optimizer with hyperparameter, the number of epochs is 33, window size is1, and the number of LSTM units in the hidden layer is [4, 4, 3], the RMSE and MAPE values during training are 100.58 and 1.49%, while in testing are 94.43 and 1.86%. Then, the model with hyperparameter number of epochs is 38, window size is 1, and the number of LSTM units in the hidden layer is [4, 4, 3], the RMSE and MAPE values during training are 108.30 and 1.53%, while in testing namely 95.39 and 1.88%.

In the model using Adadelta optimizer with hyperparameter, the number of epochs is 35, window size is 1, and the number of LSTM units in the hidden layer is [1, 3, 2], RMSE and MAPE values during training namely 106.43 and 1.51%, while in testing it is 94.47 and 1.86%. Then, the model with hyperparameter number of epochs is 18, window size is 2, and the number of LSTM units in the hidden layer is [2, 1, 4], the RMSE and MAPE values during training are 99.69 and 1.58%, while in testing namely 98.09 and 1.91%.

In the model that uses RMSprop optimizer with hyperparameter, the number of epochs is 41, window size is 1, and the number of LSTM units in the hidden layer is [4, 5, 5], the RMSE and MAPE values during training are 102.91 and 1.52%, while in testing are 94.40 and 1.88%. Then, the model with hyperparameter number of epochs is 43, window size is 5, and the number of LSTM units in the hidden layer is [3, 1, 4], the RMSE and MAPE values during training are 102.83 and 1.53%, while in testing namely 95.49 and 1.91%.

In the model using Adam optimizer with hyperparameter, the number of epochs is 38, window size is 4, and the number of LSTM units in the hidden layer is [4, 5, 2], the RMSE and MAPE values during training are 94.28 and 1.51%, while in testing are 95.49 and 1.87%. Then, the model with hyperparameter number of epochs is 24, window size is 2, and the number of LSTM units in the hidden layer is [4, 5, 2], the RMSE and MAPE values during training are 95.93 and 1.51%, while in testing namely 95.06 and 1.87%.

In the model using Nadam optimizer with hyperparameters, the number of epochs is 15, window size is 6, and the number of LSTM units in the hidden layer is [3, 3, 2], the RMSE and MAPE values during training are 93.03 and 1.51%, while in testing are 95.62 and 1.87%. Then, the model with hyperparameter number of epochs is 28, window size is 2, and the number of LSTM units in the hidden layer is [4, 5, 4], the RMSE and MAPE values during training are 95.99 and 1.50%, while in testing namely 95.17 and 1.87%.

3.2 *Case 2: shares of bank mandiri*

The forecasting results for Case 2 is shown in Table 3. In the model using Adagrad optimizer with hyperparameter, the number of epochs is 36, window size is 4, and the number of LSTM units in the hidden layer is [1, 5, 1], the RMSE and MAPE values during training are 105.87 and 1.37%, while in testing are 153.16 and 1.89%. Then, the model with hyperparameter number of epochs is 14, window size is 5, and the number of LSTM units in the hidden layer is [4, 3, 2], the RMSE and MAPE values during training are 109.39 and 1.42%, while in testing namely 152.89 and 1.92%.

In the model that uses Adadelta optimizer with hyperparameter, the number of epochs is 35, window size is 3, and the number of LSTM units in the hidden layer is [3, 5, 4], the RMSE and MAPE values during training namely 108.21 and 1.43%, while in testing it is 152.84 and 1.94%. Then, the model with hyperparameter number epochs is 30, window size is 6, and the number of LSTM units in the hidden layer is [5, 2, 5], the RMSE and MAPE values during training are 110.35 and 1.45%, while in testing namely 153.51 and 1.93%.

In the model that uses RMSprop optimizer with hyperparameter, the number of epochs is 20, window size is 2, and the number of LSTM units in the hidden layers is [2, 2, 2], the RMSE and MAPE values during training are 104.99 and 1.35%, while in testing are 150.06 and 1.88%. Then, the model with hyperparameter number of epochs is 20, window size is 5, and the number of LSTM units in the hidden layer is [3, 3, 5], the RMSE and MAPE values during training are 105.00 and 1.35%, while in testing namely 151.44 and 1.89%.

In the model using Adam optimizer with hyperparameter, the number of epochs is 15, window size is 3 and the number of LSTM units in the hidden layer is [5, 1, 4], the RMSE and MAPE values during training are 105.15 and 1.35%, while in testing were 150.37 and 1.88%. Then, the model with hyperparameter number of epochs is 25, window size is 5, and the number of LSTM units in the hidden layer is [3, 3, 2], the RMSE and MAPE values during training are 105.10 and 1.35%, while in testing namely 150.41 and 1.88%.

In the model using Nadam optimizer with hyperparameters, the number of epochs is 44, window size is 5, and the number of LSTM units in the hidden layer is [3, 5, 3], the RMSE and MAPE values during training are 105.05 and 1.35%, while in testing are 151.59 and 1.89%. Then, the model with hyperparameter number of epochs is 21, window size is 6, and the number of LSTM units in the hidden layer is [4, 3, 5], the RMSE and MAPE values during training are 105.02

and 1.35%, while in testing namely 150.91 and 1.89%.

Table 3. Forecasting results for Case 2

Optimizers	Epochs	Neurons	Window size	Data	RMSE	MAPE(%)
Adagrad	36	[1, 5, 1]	4	Training	105.87	1.37
				Testing	153.16	1.89
	14	[4, 3, 2]	5	Training	109.39	1.42
				Testing	152.89	1.92
Adadelta	35	[3, 5, 4]	3	Training	108.21	1.43
				Testing	152.84	1.94
	30	[5, 2, 5]	6	Training	110.35	1.45
				Testing	153.51	1.93
RMSprop	20	[2, 2, 2]	2	Training	104.99	1.35
				Testing	150.06	1.88
	20	[3, 3, 5]	5	Training	105.00	1.35
				Testing	151.44	1.89
Adam	15	[5, 1, 4]	3	Training	105.15	1.35
				Testing	150.37	1.88
	25	[3, 3, 2]	5	Training	105.10	1.35
				Testing	150.41	1.88
Nadam	44	[3, 5, 3]	5	Training	105.05	1.35
				Testing	151.59	1.89
	21	[4, 3, 5]	6	Training	105.02	1.35
				Testing	150.91	1.89

4. Discussion

From the results of training and testing of each optimization with a different case, it can be seen that the resulting RMSE value shows a small value, which means that the model generated from this prediction has a small error rate. And the results of the MAPE value have a value below 10% which shows the prediction model has a very good level of accuracy. With the results of using several optimizations to produce small RMSE and MAPE values in this study, it can be said that GA-LSTM can improve performance and save time. Therefore, in one process, we can find hyperparameters to use [23-25]. Even so, if you look at Tables 2 and 3, it can be seen that the MAPE values generated by the optimizers Adam and Nadam have the same value. For the RMSE, in Adam's first model between training and testing difference error of 1.21 and in the second model it has an error 0.87. While the RMSE in the first model of Nadam between training and testing difference error of 2.59 and in the second model it has an error 0.82. Then, from Tables 2 and 3, it can be seen that the MAPE value generated by the two models by optimizers has the same value. Therefore, it seems that the value is quite stable and small using the Adam optimizer.

The plot of the close price forecasting results is shown in Figures 4 and 5.

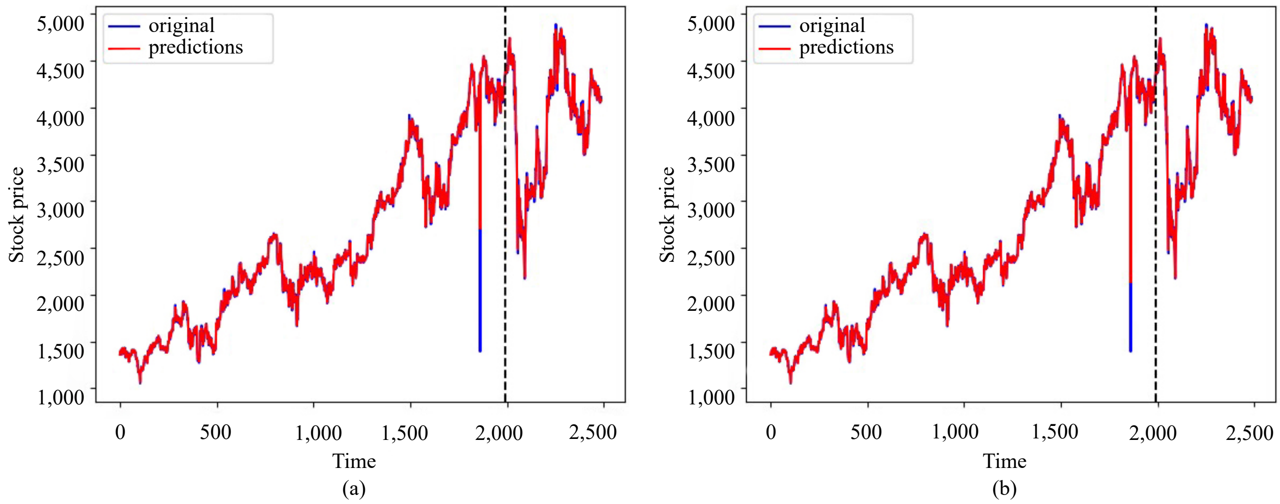

Figure 4. The plot of Bank BRI's stock forecasting results with the Adam optimizer

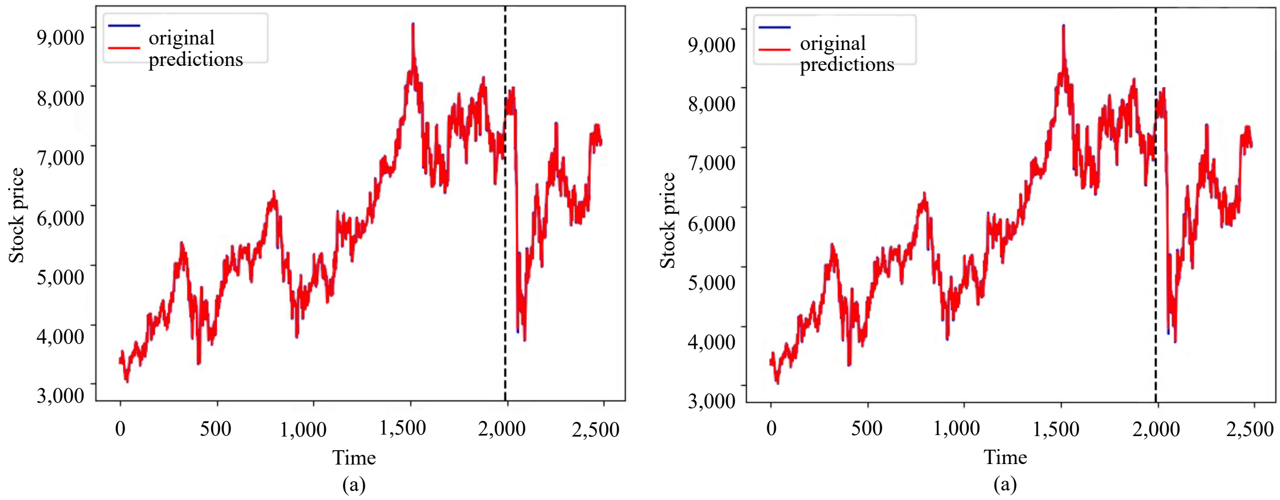

Figure 5. The plot of Bank Mandiri's stock forecasting results with the Adam optimizer

In the data forecasting results plot, the blue plot represents actual stock data, and the red plot represents forecasting data. From Figures 4 and 5, it can be seen that the predicted data and actual data have a very small difference or it can be said that the prediction using the GA-LSTM [23-25] is close to the actual data which is reinforced by the MAPE results below 10%. This is a good result, which shows that LSTM is meaningful in everyday life and culture [33].

5. Conclusions

The application of the GA-LSTM shows very good results, as evidenced by the RMSE and MAPE values generated in the training and testing of data which show a fairly low error rate and a fairly good level of accuracy with MAPE value below 10% in every optimizer used. In one process can find the best hyperparameter to use, so that GA-LSTM is able to find the optimal solution for the model effectively. The error rate generated is quite low, in Case1 with a minimum RMSE value of 93.03 and 94.40 and in Case 2 with an RMSE value of 104.99 and 150.06 during training and testing. A fairly stable and small value is generated by the setting using the Adam optimizer. The next research can be used to look for other hyperparameters or can apply a hybrid algorithm with other DL methods. This research is expected to be applied to the same data's characteristics using GA-LSTM to look for the hyperparameter.

Author contributions

YL Sukestiyarno: conceptualization, methodology, validation, data collection, data analysis, writing-preparation of original draft, writing-reviews and editing, project administration. Dian Tri Wiyanti: conceptualization, methodology, data collection, writing-preparation of original draft, writing-reviews and editing, fundraising. Lathifatul Azizah: validation, data analysis, writing-reviews and editing, project administration, fundraising. Wahyu Widada: methodology, validation, data collection, fundraising. Khathibul Umam Zaid Nugroho: validation, data collection, project administration, fundraising. All authors have read and approved the published version of the manuscript.

Funding

This research was funded by the researchers themselves.

Statement of informed consent

Informed consent was obtained from all subjects involved in this study.

Data availability statement

The data presented in this study is available upon request from the concerned authors. Data are not publicly available due to confidentiality and research ethics.

Conflict of interest

Authors declare there is no conflict of interest at any point with reference to research findings.

References

[1] Prasetya BD, Pamungkas FS, Kharisudin I. Modeling and forecasting stock data with time series analysis using Python. *PRISMA, Prosiding Seminar Nasional Matematika*. 2020; 3: 714-718.

[2] Jadmiko P. *Forecasting of Stock Prices on Indonesian Sharia Stock Index* (*ISSI*) *Using Fuzzy Time Series Markov Chain.* Undergraduate thesis. Universitas Islam Indonesia; 2018.

[3] Lubis JK, Kharisudin I. Long short term memory method and generalized autoregressive conditional heteroscedasticity for stock data modeling. *PRISMA, Prosiding Seminar Nasional Matematika*. 2021; 4: 652-658.

[4] Mishra N, Soni HK, Sharma S, Upadhyay AK. A comprehensive survey of data mining techniques on time series data for rainfall prediction. *Journal of ICT Research & Applications*. 2017; 11(2): 168-184.

[5] Ayu Rezaldi D. ARIMA method forecasting stock data PT. Indonesian Telecommunications. *PRISMA, Prosiding Seminar Nasional Matematika*. 2021; 4: 611-620.

[6] Bisong E. *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*. Berkeley, CA: Apress; 2019.

[7] Abiodun OI, Jantan A, Omolara AE, Dada KV, Mohamed NA, Arshad H. State-of-the-art in artificial neural network applications: A survey. *Heliyon*. 2018; 4(11): e00938.

[8] Varghese LR, Kandasamy V. Convolution and recurrent hybrid neural network for hevea yield prediction. *Journal of ICT Research & Applications*. 2021; 15(2): 188-203.

[9] Chen K, Zhou Y, Dai F. A LSTM-based method for stock returns prediction: A case study of China stock market. In: *2015 IEEE International Conference on Big Data* (*Big Data*). Santa Clara, CA, USA: IEEE; 2015. p.2823-2824.

[10] Lipton ZC, Berkowitz J, Elkan C. A critical review of recurrent neural networks for sequence learning. *ArXiv*. 2015; 4. Available from: doi:10.48550/arXiv.1506.00019.

[11] Hu Z, Zhao Y, Khushi M. A survey of forex and stock price prediction using deep learning. *Applied System Innovation*. 2021; 4(1): 9. Available from: doi:10.3390/asi4010009.
[12] Wu JM, Sun L, Srivastava G, Lin JCW. A novel synergetic LSTM-GA stock trading suggestion system in Internet of Things. *Mobile Information Systems*. 2021; 2021: 6706345. Available from: doi:10.1155/2021/6706345.
[13] Arfan A, Lussiana ETP. Prediction of stock prices in Indonesia using the long short-term memory algorithm. *Seminar Nasional Teknologi Informasi dan Komunikasi STI&K (SeNTIK)*. 2019; 1(3): 225-230.
[14] Peter G, Matskevichus M. Hyperparameters tuning for machine learning models for time series forecasting. In: *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*. Granada, Spain: IEEE; 2019. p.328-332.
[15] Chung H, Shin KS. Genetic algorithm-optimized long short-term memory network for stock market prediction. *Sustainability*. 2018; 10(10): 3765.
[16] Bouktif S, Fiaz A, Ouni A, Serhani MA. Optimal deep learning LSTM model for electric load forecasting using feature selection and genetic algorithm: Comparison with machine learning approaches. *Energies*. 2018; 11(7): 1636.
[17] Sugiyarto I, Faddillah, U. Optimization of artificial neural network with genetic algorithm on approval credit card prediction. *Jurnal Teknik Informatika STMIK Antar Bangsa*. 2017; 3(2): 151-156.
[18] Selvin S, Vinayakumar R, Gopalakrishnan EA, Menon VK, Soman KP. Stock price prediction using LSTM, RNN and CNN-sliding window model. In: *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. Udupi, India: IEEE; 2017. p.1643-1647.
[19] Manaswi NK. Basics of tensorFlow. In: *Deep Learning with Applications Using Python*. Berkeley, CA: Apress; 2018. Available from: doi:10.1007/978-1-4842-3516-4_1.
[20] El-Amir H, Hamdy M. *Deep Learning Pipeline: Building a Deep Learning Model with TensorFlow*. Berkeley, CA: Apress; 2020. Available from: doi:10.1007/978-1-4842-5349-6.
[21] Siami-Namini S, Tavakoli N, Namin AS. A comparison of ARIMA and LSTM in forecasting time series. In: *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. Orlando, FL, USA: IEEE; 2018. p.1394-1401.
[22] Rhanoui M, Yousfi S, Mikram M, Merizak H. Forecasting financial budget time series: ARIMA random walk vs LSTM neural network. *IAES International Journal of Artificial Intelligence*. 2019; 8(4): 317-327.
[23] Pawar K, Jalem RS, Tiwari V. Stock market price prediction using LSTM RNN. In: Rathore V, Worring M, Mishra D, Joshi A, Maheshwari S. (eds.) *Emerging Trends in Expert Applications and Security*. Advances in Intelligent Systems and Computing, vol 841. Singapore: Springer; 2019. p.493-503. Available from: doi:10.1007/978-981-13-2285-3_58.
[24] Karno ASB. Prediction of BRI Bank stock time series data with machine learning LSTM (Long Short-Term Memory). *Journal of Informatic and Information Security*. 2020; 1(1): 1-8.
[25] Yotenka R, El Huda F. Implementation of long short-term memory on stock prices of plantation companies in Indonesia. *UJMC (Unisda Journal of Mathematics and Computer Science)*. 2020; 6(1): 9-18.
[26] Wen X. Single-site passenger flow forecast based on ga-lstm. In: *Proceedings of the 6th International Conference on Control Engineering and Artificial Intelligence*. New York: ACM; 2022. p.16-20. Available from: doi:10.1145/3522749.3523073.
[27] Al Ali A, Khedr AM, El Bannany M, Kanakkayil S. GALSTM-FDP: A time-series modeling approach using hybrid GA and LSTM for financial distress prediction. *International Journal of Financial Studies*. 2023; 11(1): 38.
[28] PT Bank Rakyat Indonesia (Persero) Tbk (BBRI.JK) Stock Historical Prices & Data-Yahoo Finance. Available from: https://finance.yahoo.com/quote/BBRI.JK/history?p=BBRI.JK [Accessed 3rd February 2022].
[29] PT Bank Mandiri (Persero) Tbk (BMRI.JK) Stock Historical Prices & Data-Yahoo Finance. https://finance.yahoo.com/quote/BMRI.JK/history?p=BMRI.JK [Accessed 3rd February 2022].
[30] Goodfellow I, Bengio Y, Courville A. *Deep Learning*. MIT Press; 2016. Available from: doi:10.1016/j.ijpe.2018.03.022.
[31] Sautomo S, Pardede HF. Prediction of Indonesian government spending using long short-term memory (LSTM). *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*. 2021; 5(1): 99-106.
[32] Hadi MI, Abidin Z. Prediksi harga cryptocurrency menggunakan metode long short term memory dengan optimasi adaptive moment estimation. *Scientific Journal of Informatics*. 2019; 6(1): 1-11.
[33] Nugroho KU, Widada W, Herawaty D. The ability to solve mathematical problems through youtube based ethnomathematics learning. *International Journal of Scientific & Technology Research*. 2019; 8(10): 1232-1237.