Generative Adversarial Network in Predicting LQ45 Stock Price Index

Nafisya Alya Aurelitha[1], Mila Novita[2], dan Gianinna Andaneswari[3]

Department of Mathematics, Faculty of Mathematics and Natural Sciences, University of Indonesia, UI Depok Campus, 16424, Indonesia

Email: nafisya.alya@ui.ac.id

Abstract

The LQ45 Stock Price Index is an index that measures the price performance of 45 stocks that have high liquidity and large market capitalization listed on the Indonesia Stock Exchange. The LQ45 Stock Price Index prediction can be used to measure the performance of a stock portfolio in the future so that investors can evaluate the shares they own. Predicting the LQ45 Stock Price Index is a difficult task because this stock price index data tends to have quite high fluctuations. For this reason, an appropriate technique is needed to predict the LQ45 Stock Price Index. The LQ45 Stock Price Index is a type of time series data. Several models have been developed to predict time series data, one of which is machine learning. Generative Adversarial Network (GAN) is a special approach to machine learning through generative modeling. The GAN method can produce predictions that have high accuracy, because GAN uses two networks, namely generator and discriminator. Long Short-Term Memory (LSTM) is used as generator to study data and make predictions and Convolutional Neural Network (CNN) is used as discriminator to classify data. Therefore, in this thesis, the author applies the GAN method in predicting the LQ45 Stock Price Index. The application of this method aims to increase accuracy in predictions so that investors can measure the performance of their stock portfolio in the future properly. The data used in this thesis is the daily closing price of the LQ45 Stock Price Index from the period 2 January 2019 to 30 December 2022. The prediction results of the LQ45 Stock Price Index can be shown by the MAPE value. For training data, the MAPE value is 20,9340% and for testing data, the MAPE value is 2,3740%. These results use a comparison of 80% training data and 20% testing data.

Keywords: Convolutional Neural Network (CNN), deep learning, discriminator, Long Short-Term Memory (LSTM), neural network, time series

Introduction

Currently, more and more people are interested in investing, especially shares. To find out whether share prices are rising or falling, predictions are needed that predict future share prices. The Indonesian Stock Exchange has 11 types of stock price indexes, one of which is the LQ45 Stock Price Index. In this research, the sample that will be used as observation material is the LQ45 Stock Price Index. According to the Indonesian Stock Exchange (2022), the LQ45 Stock Price Index is an index that measures the price performance of 45 shares that have high liquidity and large market capitalization and are supported by good company fundamentals.

LQ45 Stock Price Index movement data is a type of time series data. According to Idrees et al. (2019), time series data refers to ordered data or a collection of data taken at the same time interval. Time series data prediction can use various methods, one of which is machine learning.

The machine learning method aims to enable computers to learn automatically without human intervention in adjusting appropriate actions in analyzing time series data and tracking objects detected due to noise and so on (Husnaini, 2023).

Machine learning also has various approaches to predicting time series data. In this research, the author explains one specific approach to machine learning through generative modeling called Generative Adversarial Network (GAN). Among other machine learning methods, GAN-based approaches with generators and discriminators are found to be more suitable for handling time series-based stock price index data. GANs can learn quickly from the internal representation of the data and the density distribution of the data. This is because GANs use trained generators whose aim is to learn data and make predictions. The generator attempts to produce fake samples that are very similar to the real samples. GANs also use a trained discriminator as a classifier. The discriminator attempts to prevent fake samples from the generator from being classified into real samples. These two networks will continue to iterate until the generator produces fake samples that are very similar to the real samples and the discriminator cannot distinguish the fake samples from the real samples. As a result, GANs can produce data that is very similar to the original data and of course accurate predictions (Subba et al., 2022). Therefore, the author proposes this method for predicting the LQ45 Stock Price Index, so that the resulting prediction data has high accuracy.

Generative Adversarial Network (GAN) is an artificial neural network architecture that uses two opposing networks to form or generate completely new data. GAN is divided into two constituent artificial networks, namely generator and discriminator. The generator and discriminator are trained simultaneously and compete with each other during training. The generator will generate samples from a random vector. The discriminator will receive sample results from the generator and determine whether the samples are genuine or fake (generated from the generator). The generator will try to minimize its loss function so that the resulting sample will be determined as the original sample by the discriminator. On the other hand, the discriminator attempts to prevent the fake samples received from being classified as genuine samples. (Goodfellow et al., 2020)

Generative Adversarial Network was first introduced by Goodfellow et al. (2014). This research still uses input data in the form of image data and aims to predict input. The use of Generative Adversarial Network to predict stock prices was first carried out by Zhang et al. (2019).

In the study, Zhang et al. proposed a new Generative Adversarial Network (GAN) architecture with Multi-Layer Perceptron (MLP) as the discriminator and Long Short-Term Memory (LSTM) as the stock closing price forecasting generator. Experimental results show that the new GAN can obtain promising performance in stock price prediction compared with other models in machine learning and deep learning.

Subsequently, GANs began to be used several times to predict stock prices. Xu et al. (2020) proposed a stock prediction model using a Generative Adversarial Network (GAN) with Gated Recurrent Units (GRU) used as a generator and a Convolutional Neural Network (CNN) as a discriminator. Furthermore, in the research of Subba et al. (2022), a GAN-based deep learning framework called Stock-GAN is implemented with a generator and discriminator. The generator uses Long Short-Term Memory (LSTM), a variant of Recurrent Neural Network (RNN), while the discriminator uses Convolutional Neural Network (CNN). Apart from that, in Stock-GAN, there are several processes before entering the GAN process that differentiate it from ordinary GANs, namely starting from Variational Autoencoder (VAE) to extract input data features followed by feature selection and dimension reduction with Principal Component Analysis (PCA) and XGBoost , then proceed to the GAN process to produce output.

This research aims to construct a Generative Adversarial Network (GAN) and analyze the performance of GAN to predict the LQ45 Stock Price Index. The type of prediction used is ex post prediction, namely predictions made after the event occurs. The generator used is the Long Short-Term Memory (LSTM) method and the discriminator used is the Convolutional Neural Network (CNN) method. Evaluation of the performance of the Generative Adversarial Network (GAN) in predicting the LQ45 Stock Price Index using the Mean Absolute Percentage Error (MAPE) value.

Theoretical Review

Generative Adversarial Network

Ian J. Goodfellow introduced the Generative Adversarial Network (GAN) in 2014. GAN is an artificial neural network architecture that uses two opposing networks to form or generate completely new data. Furthermore, the concept of Generative Adversarial Network (GAN) can be explained from three parts as the name suggests, namely generative, adversarial, and network.

Generative means that this network involves learning input data automatically in such a way that the model can be used to produce new data that is similar to the original data. GAN learns the data distribution of the training set so that it can generate new data points that are similar to the training dataset. This network uses artificial intelligence (AI), statistics and probability in its applications. So, GANs are trained to be able to generate new data based on a collection of data that has been seen previously during the training process.

Adversarial means that the model training is done in an adversarial setting, i.e. both networks in the GAN try to beat each other and by doing that, they both get better and better. The competition between them makes both networks continually better with respect to their respective goals. This adversarial process produces synthetic data that becomes increasingly realistic over time, that is, highly convincing data that is difficult to distinguish from the real thing.

Network means network, which in this case uses a neural network as an artificial intelligence (AI) algorithm for training purposes. Neural networks are a method in AI that teaches computers to process data in a way that is inspired by the human brain, namely receiving input, processing that input, then producing output.

Based on the explanation of these three parts, the combination of the three can be defined as GAN, which is a neural network structure that learns generative models and is adversarial or competing with each other to ultimately produce synthetic data that is difficult to distinguish from real data. In the GAN architecture, there is a network of generators and discriminators. The main idea behind GANs is to learn two such networks, where the generator network is responsible for making predictions on data that is very similar to real data followed by a discriminator network that is responsible for distinguishing between fake data and real data.

The generator in GAN is a neural network that makes forecasting data to be trained on the discriminator. The generator is responsible for creating fake data that is very similar to the real data from the dataset. The output from the generator is then fed to the discriminator. The generator is trained to trick the discriminator, that is, to make forecasts of data, which when seen by the discriminator, the discriminator cannot differentiate between real and fake data.

The discriminator in a GAN is a neural network that classifies input data as real data from a dataset or fake data created by a generator. The discriminator input data comes from two different sources, namely real data which is a dataset and fake data which is prediction data made by the generator. The discriminator is trained to be able to recognize details from real data so that when compared with fake data from the generator it will be able to detect the slightest differences. The discriminator is also trained to learn fake data from the generator to be more accurate in detecting characteristics of fake data. The discriminator is expected to output 0 when the input is fake data and is expected to output 1 when the input is real data. An output value closer to 0 indicates that the data is considered fake, while an output value closer to 1 indicates that the data is considered genuine. So, the discriminator is used to train the generator so that the generator continues to improve its performance so that it can produce accurate predicted values.

The process from generator to discriminator is repeated until it reaches equilibrium conditions, namely conditions where the generator and discriminator reach a balance point, and the GAN has been successfully trained. The generator and discriminator get better at their respective jobs after each iteration. If the discriminator considers the fake data from the generator to be real data or produces a value close to 1, then the generator will be considered successful in creating forecast data that is very similar to the real data. However, if the discriminator can differentiate between fake data and real data or produces a value close to 0, then the generator must try to improve the quality of the forecasting data it produces. In this process, the generator's performance will continue to be improved until the resulting forecast data is very difficult to distinguish from the original data.

The generator and discriminator are trained together with opposing goals. The generator's goal is to fool the discriminator, so the generator is trained to maximize the final classification error (between the original data and the generator-generated data). Meanwhile, the goal of the discriminator is to detect fake data generated by the generator, so that the discriminator is trained to minimize the final classification error. The opposing goals of these two networks explain the name "adversarial network". The GAN process is complete when the generator creates fake data that closely resembles real data, so that the discriminator is unable to differentiate between fake data and real data. Therefore, after the GAN process is complete, a generator model will be produced which can produce accurate prediction values because it has been trained by the discriminator.

Long Short-Term Memory (LSTM) was found to be good at forecasting time series data. LSTM is a development algorithm from Recurrent Neural Network (RNN), which has the advantage of being able to overcome the vanishing gradient problem, namely changes in the range of values from one layer to the next layer, which cannot be overcome by RNN. LSTM has also proven to be adept at making predictions over a long period of time compared to other algorithms. Therefore, LSTM is chosen to be the generator. Meanwhile, the discriminator uses a Convolutional

Neural Network (CNN). CNN is one of the machine learning methods from the development of Multi-Layer Perceptron (MLP). The way CNN works is similar to MLP, but in CNN each neuron is presented in 2 dimensions, unlike MLP where each neuron is only 1 dimension. CNN is also proven to be superior in terms of classification compared to other algorithms.

The overall framework of GANs starts from data preprocessing. Next, training data and testing data are divided. Training data is used to train the model, while testing data is used to evaluate the model's performance. Next, in the model building step, the generator and discriminator are initialized randomly. The generator is then trained to produce fake data that is similar to the real data and the discriminator is trained to distinguish the fake data from the real data. Next, the best GAN hyperparameters are fit to the model to obtain optimal model performance.

Next, in the model training step, the training data is passed to the generator and then the discriminator repeatedly. The generator and discriminator will learn from each other and improve their performance to produce the final model. Next, the final model from the training process is used on testing data to evaluate model performance. Next, in the performance evaluation step, model performance is measured using MAPE.

In more detail, the model training algorithm starts from data predictions made using a generator (LSTM). The prediction results from the generator (fake data) and stock price data (real data) are given to the discriminator (CNN). The discriminator classifies the data. The iterative process uses a backpropagation algorithm to adjust the weight and bias values. The process continues until convergence, which is the state where the GAN approaches the optimal solution. In the process, data predictions are updated. The step ends with the algorithm returning a final prediction.

Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) is a nonlinear model for ordered data developed from RNN. Unlike RNNs which only have short-term memory units, LSTMs have memory units that are capable of retaining information for longer periods. LSTM uses 4 gates, namely forget gate, input gate, input modulation gate, and output. The equation for the hidden layer of LSTM can be written as follows (Houdt et al., 2020).

$$F_s = \sigma(W_F X_s + U_F H_{s-1} + B_F)$$

$$I_s = \sigma(W_I X_s + U_I H_{s-1} + B_I)$$

$$M_s = \tanh(W_M X_s + U_M H_{s-1} + B_M)$$
$$O_s = \sigma(W_O X_s + U_O H_{s-1} + B_O)$$
$$C_s = F_s \times C_{s-1} + I_s \times M_s$$
$$H_s = \tanh(C_s) \times O_s$$

with

- F_s, I_s, M_s, O_s is the forget gate, input gate, input modulation gate, and output gate matrix at the s-th time step
- $\sigma()$ and $tanh()$ is the sigmoid and tanh activation function of the hidden layer
- W_F, W_I, W_M, W_O is a weight matrix that maps input from the input layer to each gate
- X_s is the input layer matrix at the s-th time step
- U_F, U_I, U_M, U_O is a weight matrix that maps the output of the previous hidden state to each gate
- B_F, B_I, B_M, B_O is the bias matrix of each gate
- C_s is the cell state matrix at the s-th time step
- H_s is the hidden state matrix at the s-th time step
- s denotes the s-th time step with $s = 1,2,\ldots,S$

The equation for the output layer of LSTM can be written as follows (Houdt et al., 2020).

$$Y_t = VH_S + B_Y + \epsilon_t$$

with

- Y_t is the output layer matrix at time t
- V is the weight matrix of the output layer
- H_S is the hidden state matrix at the last time step
- B_Y is the bias matrix of the output layer
- ϵ_t is the error matrix at time t
- S denotes the last time step

The following is the activation function used in the equations above.

1. Sigmoid Activation Function (Logistic Function)

   The sigmoid function is a function that has a value between 0 and 1. In LSTM, the sigmoid function is used as an activation function to find the value of the forget gate, input gate, and output gate. A value of 0 means the information will be forgotten or will not be included, while a value of more than 0 to 1 means the information will be retained or included. This is in accordance with the function of the three gates.

The sigmoid function is shown by the following equation.

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

(Nwankpa *et al.*, 2018)

2. Hyperbolic Tangent Activation Function (tanh)

   The hyperbolic tangent function is a function that has a value between -1 to 1. In LSTM the hyperbolic tangent function is used in calculating modulation gate and hidden state inputs. A value of -1 means the information will be deleted, while a value of more than -1 to 1 means the information will be kept. This is in accordance with the function of the two units.

   The hyperbolic tangent function is shown by the following equation.

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

(Nwankpa *et al.*, 2018)

Next, create input data with the selected time step. This input data is used in the input layer. The input data is data from the time before t-th, namely $t-1, t-2, \dots, t-S$. This input data depends on the number of time steps (S) selected. If the output data is e_t and the selected time step is S then the input data is $e_{t-1}, e_{t-2}, \dots, e_{t-S}$.

Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) is a type of neural network whose way of working is similar to a regular neural network, but there is an additional layer after the input called a convolutional layer. In CNN, a mathematical operation is carried out called convolution, where convolution itself is a linear operator. CNN has three main layers, namely convolutional layers, flattening layers, and fully connected layers.

1. *Convolutional Layer*

   The most important thing in this layer is the convolution operator which is the dot product of two real-valued matrices (Goodfellow et al., 2016). The two matrices are the input data matrix and the matrix that contains the values for extracting the input matrix, namely the filter or kernel. The elements in the filter or kernel matrix are called weights. The convolution process is carried out by shifting the kernel across the input data and calculating the product of the values in the input data with the weight values in the kernel. The weights determine how the input is processed by the CNN to produce the output. During the learning process, weights will be updated to improve model accuracy. The convolution operation at this layer is responsible for changing the input into a set of features through filters or kernels (Yang, 2019).

The following is the function resulting from the convolution operation or what is called a feature map (Goodfellow et al., 2016).

$$S[i,j] = \sum_{m}\sum_{n} I[i-m, j-n] * K[m,n]$$

with

- $K(m,n)$ is the filter matrix or kernel
- $\mathrm{I}(i,j)$ is the input matrix

In the CNN method, the input is numerical data so that the convolution operation is carried out between the input matrix containing numerical data and the kernel matrix. The size of the input matrix depends on the size of the numeric data, for example $n_h \times n_w$, while the size of the kernel matrix can be set independently, for example $k_h \times k_w$. So, the output size can be calculated by $(n_h - k_h + 1) \times (n_w - k_w + 1)$. Throughout the training process, the convolution layer will identify the weight values in the kernel that work best in detecting patterns based on the given training data.

The CNN in the convolutional layer has hyperparameters that need to be considered because adjusting these hyperparameters can help the model perform better in classification tasks. Here are the hyperparameters to pay attention to is padding. Padding is the process of adding elements to each side of the matrix. Padding aims to change the dimensions of the feature map so that when extracted it does not lose dimensions drastically, so that more features can be successfully extracted (Zhang et al., 2019). Padding keeps the output size of the convolution process the same as the input size. The input convolution operation $(n_h \times n_w)$ after zero padding with the kernel $(k_h \times k_w)$ produces an output of size $(n_h - k_h + p_h + 1) \times (n_w - k_w + p_w + 1)$, where p_h is the number of increases in the size of the top and bottom sides of the matrix and p_w is the number of increases in the size of the right and left sides of the matrix.

Before entering the flattening layer, the results of the convolution operation are summed with the bias parameters. Bias is a parameter used to adjust the output offset of each neuron or add a constant value to the output of each neuron. The bias is initialized randomly. During the learning process, biases will be updated to improve model accuracy.

2. Flattening Layer

   The output from the convolutional layer is still in the form of a matrix measuring two dimensions or more, so it is necessary to form a one-dimensional vector through the flattening layer before going to the fully connected layer (Yang, 2019). The flattening layer changes the output from the convolutional layer into input that can be processed by the fully connected layer, because the fully connected layer can only process input in the form of a 1-dimensional vector. Therefore, there needs to be a flattening process to convert the convolutional layer output into a 1-dimensional vector. Flattening layer is the initial stage of forming a fully connected layer.

3. Fully Connected Layer

   A fully connected layer is a layer where every neuron in the previous layer is connected to every neuron in the next layer. This layer is the last layer in the CNN which makes each layer connected, so that you can get the final result in the form of the desired class. The fully connected layer is tasked with classifying the output from the flattening layer in the form of vectors to obtain the final result in the form of 2 classes, namely the real data class and the fake data class.

   The task of the fully connected layer in this research is to classify input data into real data classes or fake data classes. The output from the fully connected layer is a probability value calculated using the sigmoid activation function. The sigmoid function is a function that has a value between 0 and 1. A value close to 0 indicates that the input is more likely to enter the fake data class, while a value close to 1 indicates that the input is more likely to enter the real data class. By using the sigmoid activation function in the last layer of the discriminator, the discriminator output can be interpreted as the probability that the input data belongs to the real data class or the fake data class.

Loss Function Generative Adversarial Network

1. Loss Function Discriminator Network (D_{loss})

The idea of a loss function discriminator network is to make the discriminator more accurate in distinguishing real data from fake data. The loss function of the discriminator network is calculated using the discriminator predictions for real data and fake data. The D_{loss} equation is as follows (Lin et al., 2020).

$$D_{loss} = -\frac{1}{m}\sum_{i=1}^{m} \log D(y) - \frac{1}{m}\sum_{i=1}^{m} \log\left(1 - \mathrm{D}\big(G(x)\big)\right)$$

with

- (y) is the discriminator output probability for real data
- $D\big(G(x)\big)$ is the discriminator output probability for fake data
- $G(x)$ is the output produced by the generator (fake data)
- y is the data that the generator wants to produce
- x is the input data for the generator

This equation calculates the average of the discriminator prediction entropy for real data and fake data. The smaller the discriminator loss value, the better the discriminator can distinguish between real data and fake data.

2. Loss Function Generator Network (G_{loss})

The idea of a loss function generator network is to make the generator produce fake data that is more similar to real data. The loss function of the generator network is calculated using the discriminator predictions for fake data. The equation is as follows (Lin et al., 2020).

$$G_{loss} = -\frac{1}{m}\sum_{i=1}^{m} \left(\log D\big(G(x)\big)\right)$$

with

- $D\big(G(x)\big)$ adalah is the discriminator output probability for fake data
- $G(x)$ is the data generated by the generator (fake data)
- x is the input data for the generator

This equation calculates the entropy of the discriminator predictions for fake data. The greater the generator loss value, the better the generator can produce fake data that looks genuine.

Generative Adversarial Network Hyperparameters

The hyperparameter values that need to be determined are the learning rate, number of epochs, and batch size. Learning rate is used to control how quickly the model adjusts the weight and bias parameters. Epoch is an iteration of all input data and output data used to train the model. Batch size is the number of samples processed before the weights and bias are updated. (Yu & Zhu, 2020)

Model Accuracy Level

Mean Absolute Percentage Error (MAPE) is an error calculation method that can be calculated by finding the percentage of the Mean Absolute Error (MAE). MAPE is defined by the following equation (Temür et al., 2019).

$$MAPE = \frac{100}{n}\sum_{t=1}^{n}\left|\frac{y_t - \hat{y}_t}{y_t}\right|$$

with

- MAPE is Mean Absolute Percentage Error
- y_t is the actual value at time t
- $\hat{y}_t$ is the forecast value at time t
- n is the number of samples

This model's level of accuracy is measured to measure the model's performance in forecasting. The model with the smallest MAPE value criteria will be selected as the best model. MAPE is suitable for use in cases where the predicted value and the actual value have different units. MAPE is also suitable for use in cases where relative error is more important than absolute error. Apart from that, MAPE is also easier to understand because the results are in the form of percentages. (Chicco et al., 2021)

Research Methods

The research methods used in this research are literature study, construction of a Generative Adversarial Network (GAN) using the Python programming language, and analysis of GAN performance to predict the LQ45 Stock Price Index.

Research result

LQ45 Stock Price Index Data

The data used is the closing LQ45 Stock Price Index. The data used is daily data for the period starting from January 2 2019 to December 30 2022. Based on this period, there are 980 data samples. The data used comes from *Investing.com*. The LQ45 closing index data can be accessed and downloaded via the site https://id.investing.com/indices/jakarta-lq45-historical-data.

Establishment of a Generator Network Model: Long Short-Term Memory (LSTM)

The LSTM model created has a total of 6 layers, namely 5 hidden layers and 1 output layer. The time step used is 3 (Lin et al., 2020). The formation of the LSTM model is implemented in the Python program with the help of the Tensorflow package. The model used is sourced from the journal owned by Lin et al., (2020), but there are several changes made to increase the accuracy of the model.

1. Input

   The input from the generator is the daily LQ45 Stock Price Index closing price data.

2. Hidden Layer

   There are six hidden layers used as follows.

   a. The first hidden layer uses 1024 units, return_sequences=True, and dropout 0.3.

   b. The second hidden layer uses 512 units, return_sequences=True, and dropout 0.3.

   c. The third hidden layer uses 256 units, return_sequences=True, and dropout 0.3.

   d. The fourth hidden layer uses 128 units, return_sequences=True, and dropout 0.3.

   e. The fifth hidden layer uses 64 units, return_sequences=True, and dropout 0.3.

   f. The sixth hidden layer becomes the output layer.

3. Output

   Hidden layer that uses 1 unit and return_sequences=False. In the final results, the predicted value will be obtained from the closing price data of the daily LQ45 Stock Price Index or what is called fake data.

Establishment of a Discriminator Network Model: Convolutional Neural Network (CNN)

The CNN model formation was implemented in the Python program with the help of the Tensorflow package. The model used comes from the journal owned by Lin et al., (2020).

1. Input

   The input of the discriminator is fake data from the generator output and real data from the LQ45 Stock Price Index closing price data with an input size of 4x1.

2. Convolutional Layer

There are three convolutional layers used as follows.

a. The first convolutional layer uses 32 filters measuring 3 x 1 and padding=Same.
b. The second convolutional layer uses 64 filters measuring 5 x 1 and padding=Same.
c. The third convolutional layer uses 128 filters measuring 5 x 1 and padding=Same.

The output from the convolutional layer is called a feature map.

3. Fully connected Layer

   There are three fully connected layers used, namely as follows.

   a. The first fully connected layer uses 220 neurons and use_bias=True.
   b. The second fully connected layer uses 220 neurons and use_bias=True.
   c. The third fully connected layer becomes the output layer.

4. Output

   Fully connected layer that uses 1 neurons and a sigmoid activation function. In the final results, the values of the original data and fake data will be obtained, so that the data can be classified as closer to the real data class or the fake data class.

Determining Generative Adversarial Network Hyperparameter Values

Some of the hyperparameter values that will be determined in this final project are the number of learning rates, number of epochs, and batch size. The initial hyperparameter values were sourced from the journal belonging to Lin et al., (2020). Apart from that, hyperparameter values were also taken from the journal belonging to Xu et al., (2022). After experiments using several hyperparameter values around the values in the two journals, the optimal learning rate, number of epochs, and batch size were obtained in Table 4.1.

Table 1. Optimal GAN Hyperparameter Values

Hyperparameter	Optimal Value
Learning rate	0,00007
Number of epochs	160
Batch size	128

Generative Adversarial Network Accuracy Level based on Data Sharing

This research uses the Adam optimizer. The level of accuracy used is the Mean Absolute Percentage Error (MAPE) value from the testing data. The decimal rounding used is 4 digits. Below are the GAN results for each data division.

Table 2. GAN Accuracy Level based on the Proportion of Training Data and Testing Data

Data Training:Data Testing	MAPE Value
70:30	4,1032%
75:25	2,7037%
80:20	2,3740%
85:15	3,0746%
90:10	6,8045%
95:5	4,0989%

Based on Table 2, it can be concluded that GAN produces the best MAPE value of 2.3740% using a comparison of 80% training data and 20% testing data.

Implementation of Generative Adversarial Network

Figure 1 is the prediction result of training data and testing data from the closing price of the daily LQ45 Stock Price Index using Generative Adversarial Network (GAN) in graphic form. The blue line is the actual daily closing price data for the LQ45 Stock Price Index, while the red line is the prediction result using GAN. The level of GAN accuracy can be seen from the MAPE value for training data and testing data. GAN has a MAPE value for training data of 20.9340% and for testing data of 2.3740%.

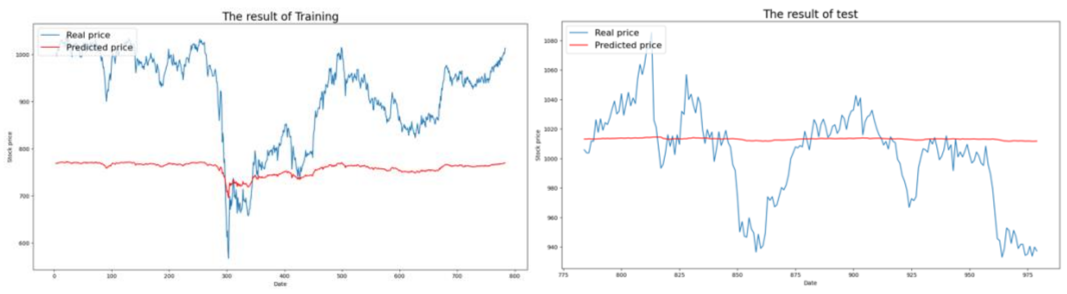

Figure 1. Prediction Results from Training Data and Testing Data on Closing Prices for the LQ45 Stock Price Index

Discussion

Model Performance Analysis

From Figure 1 it can be seen that the predicted results of the training data can quite follow the trend of the actual data. However, the predicted value is still far from the value of the actual data. Meanwhile, the predicted results of the testing data are less able to follow the trend of the actual data, but the predicted results are not far from the values of the actual data. So, the predicted results of the training data can follow the trend of the actual data, but the predicted values of the testing data are closer to the values of the actual data. This probably happens because the amount of testing data is small so the model cannot learn trends and because the actual data, namely the closing price data for the LQ45 Stock Price Index, tends to rise and fall quickly coupled with a drastic decline in March 2020 due to the Covid-19 pandemic, so difficult to predict with time series models.

The smaller the MAPE value of a model, the better the model's accuracy level. GAN has a MAPE value for training data of 20.9340% and for testing data of 2.3740%. It can be seen that the MAPE value of the training data is greater than the testing data. This probably happens because the training data has greater fluctuations than the testing data. This was exacerbated by the drastic decline in March 2020 due to the Covid-19 pandemic which made training data more difficult to predict, resulting in a MAPE value that was greater than testing data.

Conclusion

Forecasting the LQ45 Stock Price Index with Generative Adversarial Network (GAN) can be done by modeling a generator network and a discriminator network. The generator network is modeled using the Long Short-Term Memory (LSTM) method, while the discriminator network is modeled using the Convolutional Neural Network (CNN) method. The process of the generator network followed by the discriminator network is repeated until it reaches an equilibrium condition, namely the condition where the generator and discriminator reach a balance point, and the GAN has been successfully trained.

The LQ45 Stock Price Index prediction results can be shown by the MAPE value. For training data it has a MAPE value of 20.9340% and for testing data it has a MAPE value of 2.3740%. These results use a comparison of 80% training data and 20% testing data.

Suggestion

Based on the research results, the author suggests several strategies for future research related to Generative Adversarial Networks. First, predictions can be made using time series data other than the LQ45 Stock Price Index data which is predicted using the Generative Adversarial Network (GAN), for example gold prices, crude oil prices, etc. Second, the process can be carried out by adding other variables that influence the LQ45 Stock Price Index data apart from the historical closing index data for the LQ45 Stock Price Index, for example the opening index, total trading volume, etc. Third, the Generative Adversarial Network (GAN) used in making predictions can use a combination of a generator network and another discriminator network, such as a generator network that can use a Gated Recurrent Unit (GRU) and a discriminator network that can use a Multi Layer Perceptron (MLP). It is hoped that by implementing these suggestions, future research will produce deeper insights and more accurate results.

Reference list

- Bursa Efek Indonesia. (2022). Indeks. *https://www.idx.co.id/id/produk/indeks*
- Chicco, D. *et al*. (2021). The Coefficient of Determination R-Squared is More Informative than SMAPE, MAE, MAPE, MSE and RMSE in Regression Analysis Evaluation. *PeerJ Computer Science, vol. 7, no. 5*.
- Goodfellow, I. *et al.* (2014). Generative Adversarial Nets. *Annual Conference on Neural Information Processing Systems 2014.*
- Goodfellow, I. *et al.* (2016). *Deep Learning*. MIT Press.
- Goodfellow, I. *et al*. (2020). Generative Adversarial Networks. *Association for Computing Machinery Digital Library.*
- Houdt, G. V. *et al*. (2020). A Review on the Long Short-Term Memory Model. *Artificial Intelligence Review, vol. 53, no. 8.*

- Husnaini, A. (2023). Implementasi Machine Learning pada Prediksi Model Data Ketinggian Muka Air Laut dengan Metode Fbprophet dan Pendeteksian Anomali dengan Metode Klasifikasi. *Universitas Lampung.*
- Idrees, S. M. *et al*. (2019). A Prediction Approach for Stock Market Volatility Based on Time Series Data. *IEEE Access*, 1–1.
- Lin, H. *et al*. (2020). Stock Price Prediction using Generative Adversarial Networks. *Journal of Computer Science*.
- Nwankpa, C. *et al*. (2018). Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378.*
- Subba, R. P. *et al*. (2022). Multi-Model Generative Adversarial Network Hybrid Prediction Algorithm (MMGAN-HPA) for stock market prices prediction. *Elsevier*.
- Temür, A. S. *et al.* (2019). Predicting Housing Sales in Turkey Using ARIMA, LSTM and Hybrid Models. *Journal of Business Economics and Management, vol. 20, no. 5.*
- Xu, H. *et al*. (2022). A Self-Regulated Generative Adversarial Network for Stock Price Movement Prediction Based on The Historical Price and Tweets. *Elsevier.*
- Yang, X. S. (2019). *Introduction to Algorithms for Data Mining and Machine Learning*. Academic Press.
- Yu, T. & Zhu, H. (2020). Hyper-Parameter Optimization: A Review of Algorithms and Applications. *arXiv:2003.05689 [cs.LG]*
- Zhang, A. *et al.* (2019). *Dive into Deep Learning*. Unpublished Draft. Retrieved, 19.
- Zhang, K. *et al*. (2019). Stock Market Prediction based on Generative Adversarial Network. *Procedia Comput. Sci.*