Deep Learning Approaches for Predicting Intraday Price Movements: An Evaluation of RNN Variants on High-Frequency Stock Data

M Ridwan[1,2]**, K Sadik**[2,*]**, FM Afendi**[2]

[1]BPS-Statistics of Bekasi Regency
[2]Departement Statistics IPB University

*Corresponding author's e-mail: kusmans@apps.ipb.ac.id

Abstract. This study discusses the comparison of four recurrent neural networks (RNN) models: Simple RNN, Gated Recurrent Unit (GRU), Long Short-Term Memory (LSTM), and Bidirectional RNN (BiRNN), in forecasting minute-level stock price time series data. The performance of these four models is evaluated using the Mean Absolute Percentage Error (MAPE) on a stock dataset from Bank Central Asia (BBCA.JK). The experimental results reveal that the GRU model exhibits the best performance with an average MAPE of 0.0255%, followed by the LSTM model with an average MAPE of 0.0377%. The BiRNN model also demonstrates good performance with an average MAPE of 0.0668%, while the Simple RNN has the highest average MAPE at 0.5118%. This suggests that more complex recurrent architectures like GRU and LSTM have better capabilities in capturing patterns in high-frequency time series data. This study can be expanded by exploring other models such as CNN, conducting tests on diverse datasets, and experimenting with a wider range of hyperparameter variations. Additional variables such as economic indicators, global market data, and social data can also offer a more comprehensive understanding of factors influencing stock prices.

1. Introduction

1.1. Background

Predicting intraday stock price changes has become increasingly important in the setting of the global financial market, which is continually changing and growing with complicated dynamics. Understanding and responding to price fluctuations in extremely short timescales is a critical component of investment decision-making and risk management. Higher-level data, rather as daily, weekly, or monthly data, is necessary to overcome these difficulties. Stock data with minute-level granularity is typically referred to as high-frequency data in the financial sector. High-frequency data is information that is updated at extremely short intervals, sometimes in seconds, minutes, or hours [1]. High-frequency data, which includes price information, trading volume, and market depth, has become a vital source for more in-depth analysis of market dynamics. High-frequency data allows analysts and researchers to investigate patterns in price changes that occur at very short time periods, allowing them to more effectively spot trading opportunities and trends.

High-frequency data is critical in delivering real-time insights into dynamic market activity in intraday trading, when price fluctuations can occur very quickly. Observing price fluctuations in real time helps market participants to respond swiftly to prospective trading opportunities and efficiently manage risks. As a result, high-frequency data analysis is crucial because it gives a more accurate picture of price fluctuations in a short period of time, allowing market players to take more precise and timely decisions.

However, the major problem emerges when high-frequency data must be processed quickly and used to make rapid and precise investment decisions. We must figure out how to handle this fast changing and complicated data while reliably identifying patterns and trading possibilities in extremely short intervals. Classical techniques to time series analysis can have difficulties when dealing with the complexities of high-frequency data that changes quickly. Classical statistical approaches frequently rely on linear and independent assumptions, which may not be appropriate for high-frequency data with strong time dependencies and non-linear patterns. The intricacy of price swings that are difficult to recognise in intraday trading might also provide a challenge to traditional statistical methodologies.

Deep learning is another common method in time series analysis. Deep learning provides a robust way for interpreting and modelling time series data, with better data complexity handling capabilities. Recurrent Neural Networks (RNN) are a type of deep learning architecture that is commonly used in time series analysis. RNN is a sort of artificial neural network architecture created primarily to solve problems requiring sequential data, such as time series [2]. They have internal "memory" that helps them to understand temporal connections in data, recognise sophisticated patterns, and adapt to quick changes. Furthermore, RNN has grown to include variants such as Bidirectional RNN, LSTM (Long Short-Term Memory), and GRU (Gated Recurrent Unit), each with increased capabilities in managing strong temporal dependencies and quick changes in time series data.

The aim of this research is to examine and compare the performance of various RNN variants, such as Simple RNN, GRU, LSTM, and Bidirectional RNN, in predicting intraday stock price movements at a minute-level frequency for Bank BCA's stock data. The research also seeks to understand to what extent each model can handle the complexity and time dependencies in fast-paced trading activities.

This research is limited to the stock price fluctuations of Bank BCA in the year 2022. The purposeful selection of this data attempts to offer an accurate portrayal of the real and dynamic trade features throughout that period. Focusing on the year 2022 also aids in developing a better grasp of how deep learning algorithms might address the issues of forecasting intraday prices in the setting of high-frequency data.

This research is expected to provide better insights into the effectiveness and potential of deep learning approaches in addressing the complexity of high-frequency data and supporting improved decision-making in intraday trading by involving different RNN variants and focusing on high-frequency data.

1.2. Related Works

Zhao et al. in 2021 used RNN, LSTM, and GRU on daily data of the SSE 180 Index. The research results indicated that in the most basic comparison, GRU and LSTM significantly outperformed the RNN model, and the GRU model performed slightly better than the LSTM model [3]. In 2020, Li et al. compared ARIMA, SVM, and LSTM. ARIMA showed a significant deviation in forecasting high-frequency financial time series. Combining ARIMA with deep learning, this study successfully improved prediction accuracy without increasing computational complexity. The enhanced ARIMA model with deep learning provided an effective model for high-frequency trading strategy design [4]. Taroon et al. in 2020 reviewed deep learning models and compared LSTM and its variants in predicting minute-level movements of the SPDR S&P 500 index. The results showed that LSTM outperformed other models [5]. In 2021, Jebli et al. utilized Deep Learning techniques for solar energy prediction, specifically using Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), and Gated Recurrent Units (GRU). RNN and LSTM slightly outperformed GRU due to their ability to maintain long-term dependencies in time series data [6]. In 2020, Qiu et al. conducted a comparison of LSTM and GRU on S&P 500, DJIA, and HIS datasets. The experiments on the S&P 500 and DJIA datasets indicated that the coefficient of determination for the LSTM model was higher than 0.94, and the average squared error of the model was lower than 0.05 [7]. In 2022, Saravanan et al. used deep learning algorithms for air quality prediction. The proposed model was the Bidirectional Recurrent Neural Networks (Bi-RNN). The research results showed that Bi-RNN outperformed conventional methods in predicting air quality and severity [8].

According to research, LSTM and GRU models outperform RNN models in forecasting stock price fluctuations. All models (RNN, LSTM, and GRU) are capable of preserving long-term dependencies in time series data, which is critical for comprehending stock price movements impacted by long-term variables. However, inconsistent results regarding model depth imply that increasing depth does not necessarily result in higher stock price prediction ability. RNN, LSTM, and GRU models show tremendous potential for forecasting stock price changes in high-frequency data, as they address patterns that occur at short time intervals.

Overall, deep learning approaches with RNN, LSTM, GRU, and Bidirectional RNN demonstrate significant potential in predicting stock prices, especially with high-frequency data. However, the choice of the appropriate model should be tailored to the data characteristics and specific prediction objectives.

2. Methodology

2.1. High-Frequency Data

Gençay et al. in 2001 stated that high-frequency data means a very large amount of data, where the number of observations in a single day of the stock market can be equivalent to the number of daily data in 30 years [9]. Statistically, the higher the number of independently measured observations, the higher the degree of freedom, which results in better estimator values. High-frequency data paves the way for studying the financial market on different time scales, from minutes to years, and this can represent an aggregation factor of four to five times [9]. In 2012, Fabozzi explained that there are two main interests in high-frequency data. First, high-frequency data allows capturing interesting events or phenomena in the stock market with a high level of precision. Measuring intraday trading risk and seeking short-term trading profit opportunities have become of interest to many financial institutions. Second, high-frequency data can be exploited to enhance forecasting accuracy [10]. By having direct access to more detailed market transaction information, forecasting models can be built while considering very rapid price changes. In this context, analysis using high-frequency data can help generate more accurate predictions of stock price movements and other financial assets.

High-frequency data has become the primary focus of research for those who want to understand the financial market in more detail. This type of data has advantages in explaining the microstructure of the market, which involves trading rules that influence stock prices in the capital market [11].

2.2. RNN

RNN (Recurrent Neural Networks) is a type of neural network that can process sequential data of varying lengths [12]. Naturally, RNNs can be used to solve tasks related to sequential data. Examples of such tasks include language translation, speech recognition, predicting the next element in a time series, and more. The equation for RNN can be expressed as follows:

$$h_t = f(Ux_t + Wh_{t-1}) \tag{1}$$

h_t (hidden state) is the output at the time step t, x_t is the input at the time step t dan h_{t-1} is the output in the previous time step. U and W are weight matrices that convert previous inputs and outputs into current outputs. The activation function f (usually sigmoid or tanh) is used to combine the previous input and output into the current output.

The recursion relationship defines how state develops step by step through a sequence through a feedback loop over the previous state, as illustrated in the diagram in Figure 1 below. The three parameters in RNN are U (change the input x_t becomes s_t), W (change the previous state s_{t-1} becomes s_t), and V (mapping the newly calculated internal state, s_t to the output y_t).

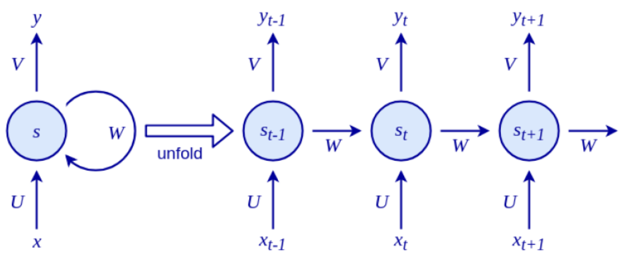

Figure 1. RNN Flow [12]

2.3. Bidirectional RNN
Bidirectional RNN is a variant of RNN that allows the processing of information from both directions: from beginning to end (forward) and from end to beginning (backward) in sequence data. Bidirectional RNNs are often used in tasks that require understanding context from both the beginning and end of sequences, such as in natural language processing and handwriting recognition. Bidirectional RNNs can better capture dependencies in sequence data by processing data in two directions.

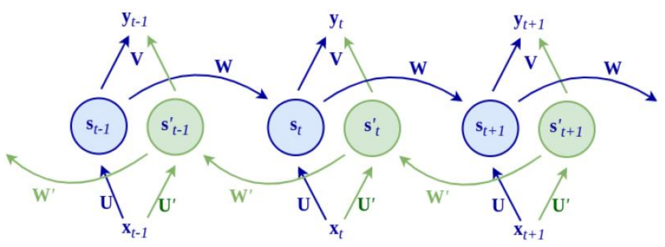

Figure 2. Bidirectional RNN Flow [12]

In Figure 2 it can be seen that the Bidirectional RNN has two sets of parameters, one to process sequence data from beginning to end (forward pass) and the other to process sequence data from end to beginning (backward pass). Here is the equation to calculate the output at each time step *t* in a Bidirectional RNN:
Forward Pass:

$$\overrightarrow{h_t} = \sigma(W_{\vec{h}}\vec{x}_t + U_{\vec{h}}\vec{h}_{t-1} + b_{\vec{h}}) \tag{2}$$

Backward Pass:

$$\overleftarrow{h_t} = \sigma(W_{\overleftarrow{h}}\overleftarrow{x}_t + U_{\overleftarrow{h}}\overleftarrow{h}_{t+1} + b_{\overleftarrow{h}}) \tag{3}$$

Output in step *t*:

$$h_t = [\overrightarrow{h_t}, \overleftarrow{h_t}] \tag{4}$$

In the forward pass, input $\overrightarrow{x_t}$ at time step t is processed using the parameters $\overrightarrow{W_h}$ and $\overrightarrow{U_h}$ with previous state $\vec{h}_{t-1}$ to produce output $\vec{h}_t$. It represents the information obtained from the beginning to the end of the sequence. In the backward pass, input $\overleftarrow{x}_t$ at the time step *t* processed using parameters $\overleftarrow{W_h}$ and $\overleftarrow{U_h}$ with previous state $\overleftarrow{h}_{t+1}$ to produce output $\overrightarrow{h_t}$. It represents the information obtained from the end to the beginning of the sequence. Output at timestep *t* where h_t is a composite of outputs from both directions $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ which combines information from both directions and can provide a better understanding of the context of each time step.

2.4. LSTM
LSTM (Long Short-Term Memory) is a variant of RNN designed to address the vanishing gradient problem and is capable of preserving long-range dependencies in time series data. By incorporating

multiple gates that regulate the flow of information, LSTM can model complex relationships in sequential data.
Here is the equation and explanation of each part of the LSTM equation:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{5}$$
$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{6}$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{7}$$
$$g_t = tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{8}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{9}$$
$$h_t = o_t \odot \tanh(c_t) \tag{10}$$

i_t is an input gate or input gate that controls how much new information is from the current input (x_t) that will be inserted into the cell state (c_t). This gateway can set whether new information is relevant to updating cell state. f_t is a forget gate that functions to control how much information from the previous cell state (c_{t-1}) will be deleted or forgotten. This gateway allows LSTM to adjust dependencies on previous data. o_t is the output gate. It controls how much information is from the cell state (c_t) will be used in generating actual output at the time step *t*. This gate can set how relevant the information of the cell state is to the output. g_t is a candidate cell state, i.e. the resulting value by combining the current inputs (x_t) and previous state (h_{t-1}) After going through the hyperbolic tangent function $(tanh)$. c_t is the actual output at time step *t* produced by combining information from the cell state (c_t) by going through the output gate (o_t) After going through the hyperbolic tangent function $(tanh)$. The architecture of LSTM can be seen in Figure 3.

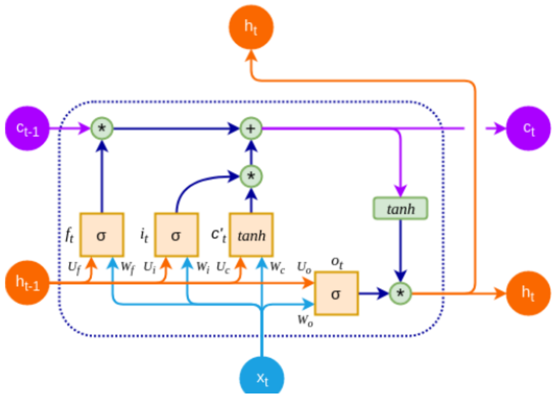

Figure 3. LSTM Architecture [12]

2.5. GRU

GRU (Gated Recurrent Unit) is a variant of the RNN designed to overcome weaknesses in the training process in the RNN such as vanishing gradients. GRU has simpler features than LSTM because it only has two gates (reset and update) compared to three gates (input, forget, and output) on LSTM. Although more concise, GRU is still able to handle remote dependency problems and overcome vanishing gradient problems that often occur in traditional RNNs.
The equation in GRU can be denoted as follows:

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \tag{11}$$
$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \tag{12}$$
$$\tilde{h}_t = tanh(W_z x_t + U_h(r_t \odot h_{t-1}) \tag{13}$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{14}$$

z_t is a reset gateway that helps the model decide how much information from previous time steps h_{t-1} which must be updated in generating h_t. If z_t close to 0 then the old information will be ignored. r_t is an update gateway that helps the model determine how much information from the current input (x_t) and previous information (h_{t-1}) which is combined to produce the value h_t. If r_t close to 0 then the old information will be ignored. $\tilde{h}_t$ is a candidate state that is a provisional estimate for h_t generated taking into account the current element (x_t) and previous information (h_{t-1}) After going through the reset gate r_t. h_t is the actual state at step time t calculated taking into account the reset gate (z_t) to replace some of the previous information (h_{t-1}) with a new candidate state $(\tilde{h}_t)$. The architecture of GRU can be seen in Figure 4.

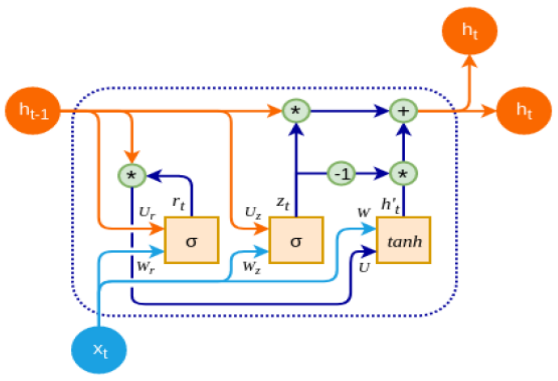

Figure 4. GRU Architecture [12]

2.6. Data

The primary focus of this research is the stock price movement data of BCA (Bank Central Asia) throughout the year 2022. This data was obtained from Google Finance, a reliable source for providing financial market data.

Table 1. BBCA Stocks Data 2022

Time Stamp	Open	Low	High	Close	Volume
03/01/2022 09:00	7325	7300	7350	7350	992300
03/01/2022 09:01	7325	7325	7350	7325	216500
03/01/2022 09:02	7325	7325	7350	7350	12100
.	.	.	.	.	.
.	.	.	.	.	.
30/12/2022 14:59	8550	8550	8550	8550	0
30/12/2022 15:00	8550	8550	8550	8550	11207900

Table 1 shows that the provided data includes information about stock price movements within 1-minute intervals from January 3rd, 2022, to December 30th, 2022. Each data row contains several crucial elements reflecting the dynamics of stock trading. Firstly, the 'Time Stamp' column indicates the time at which the stock price data was recorded. This time interval ranges from 09:00 AM to 03:00 PM daily.

Table 2. Descriptive Statistics

Statistik	Count	Mean	STD	Min	25%	50%	75%	Max
Value	59.532	7.999	511	7.000	7.600	7.900	8.475	9.375

Descriptive statistical analysis of the closing price data in Table 2 reveals several interesting insights. The average closing price over the observed period is approximately 7,998.86. This indicates that closing prices tend to cluster around this value overall. However, the relatively low standard deviation of around 510.50 suggests that individual closing prices deviate moderately from the mean. The closing price range from 7,000 to 9,375 reflects the existing variability in prices during the data period. The quartiles also provide valuable insights. The first quartile (25%) is around 7,600, while the third quartile (75%) is around 8,475. This implies that 25% of the data has closing prices below or equal to 7,600, while another 25% has closing prices above or equal to 8,475. The median closing price, located at 7,900, divides the data into two halves, indicating a balance in the distribution of prices around this value. Overall, this statistical analysis offers a comprehensive overview of the variability, central tendencies, and distribution of closing prices within the observed dataset.

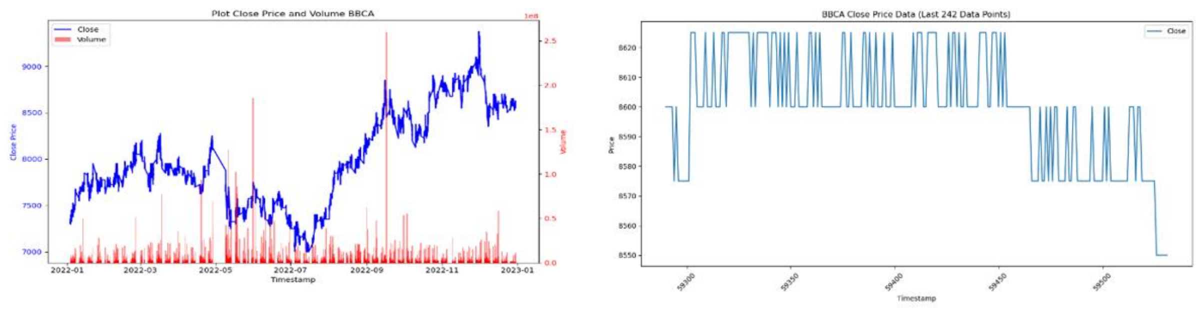

Figure 5. Closing Price Plot and Trading Volume of BBCA Shares

Figure 5 depicts the minute-by-minute stock price movement data of BCA throughout the year 2022, exhibiting characteristics that mirror financial market dynamics. These characteristics encompass rapid price fluctuations and potentially high volatility. Over the course of that year, the data demonstrates daily and weekly trends as well as price movements that may correlate with significant news or market events. The plot of minute-by-minute closing prices on the final trading day of 2022 also reveals a similar scenario, with changes in price trends observed throughout the day. This information establishes a vital foundation for the subsequent steps, wherein feature engineering techniques will be applied, and RNN models will be developed to predict intraday price movements based on this high-frequency data.

2.7. Analysis Methodology

The research workflow can be seen in Figure 6.

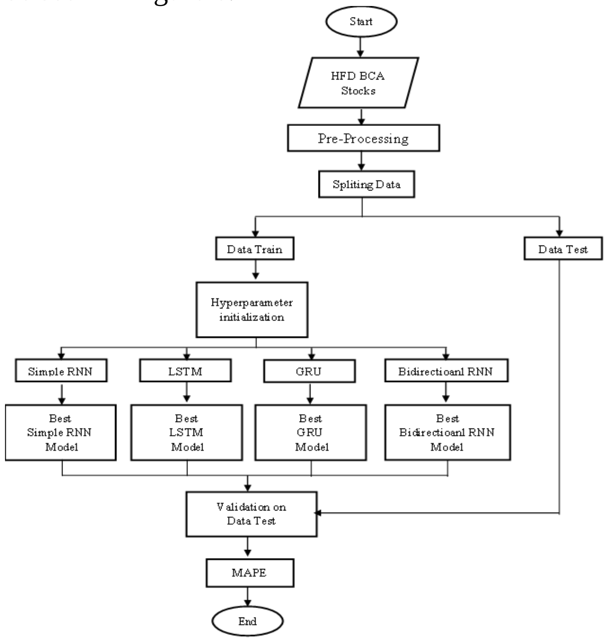

Figure 6. Stage Of Research

The initial step in this study's analysis involves extracting the closing prices from the data as the primary focus of analysis. This is crucial as the closing prices reflect the stock's value at the end of each trading interval. The subsequent step involves cleaning data outside the trading hours of the Indonesia Stock Exchange (IDX). This ensures that relevant data is used for analysis, thereby leading to a better understanding of market conditions.

The following step involves partitioning the data into training and testing sets. The training data, covering the period from January 1st, 2022, to October 31st, 2022, is used to train the model. Meanwhile, the testing data, spanning from November 1st, 2022, to December 31st, 2023, is used to assess the performance of the trained model. By segregating the data into two distinct subsets, we can observe how well the model applies what it has learned from the training data to the testing data [13]. The plot illustrating the division between training and testing data can be seen in Figure 7.

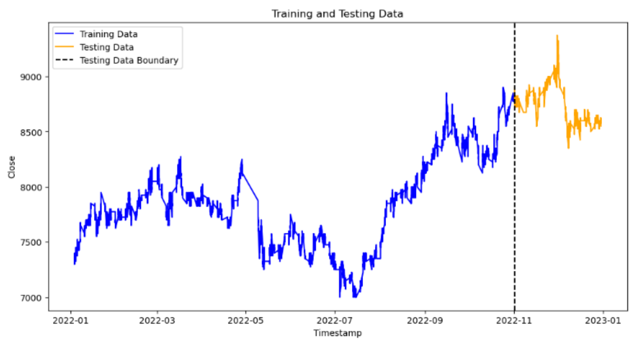

Figure 7. Training Data and Test Data Visualization

The more technical stage involves constructing various models. Four types of models will be built: Simple RNN, Bidirectional RNN, LSTM, and GRU. Each model will be constructed by considering the hyperparameter combinations in Table 3. In the context of machine learning or deep learning model development, hyperparameters are configurations that need to be specified by the user before the training process begins. First, the "Window Size" with a value of 50 refers to the number of recent data points provided as input to the model. This can affect the model's ability to capture temporal patterns in time series data. Next, "Units" refers to the number of neurons in a layer. With three variations [32, 64, 128], Units will explore the appropriate model complexity level. The learning rate or "Learning Rate" has two options [0.001, 0.0001], which will influence how much the model's weights are adjusted during training. "Epochs" are indicated using the "Early Stop" mechanism, a technique where model training is stopped early if there's no improvement in performance on validation data. The goal is to prevent overfitting, where the model adapts too closely to random variations in the training data and fails to generalize well to new data [14]. A "Batch Size" of 32 determines how many data points are processed together in each training iteration. The value of "Validation Split" at 0.2 refers to the proportion of data allocated as validation data during the model training process. In this context, 0.2 means that 20% of the total data will be used as validation data, while the remaining 80% will be used as training data.

Table 3. Model Hyperparameters

Hyperparameter	Value
Window Size	50
Units	[32, 64, 128]
Learning Rate	[0.001, 0.0001]
Epochs	Early Stop
Activation	Relu
Batch Size	32
Validation Split	0,2

By combining and adjusting these hyperparameters, we can guide the model to learn effectively and generate accurate predictions in line with the data characteristics. The main objective is to find the hyperparameter combinations that support the model in achieving optimal performance. This process can be a significant part of the deep learning model training process.

After the process of training the model using the training data with various combinations of hyperparameters is complete, the next step is to test the model's performance on the test data and calculate the Mean Absolute Percentage Error (MAPE) for each model. Forecast accuracy can be determined by assessing how well a model performs when applied to new data that wasn't used in the model's training process [15]. MAPE provides insight into how accurate a model's predictions are in terms of a percentage error compared to the actual values. By comparing the MAPE values of different models, we can identify the model that provides the most accurate and closest predictions to the actual values. The reason for choosing MAPE as the accuracy metric over RMSE and MAE in the context of comparing prediction errors in this study is that MAPE measures prediction errors in percentage form, thus providing a clearer picture of the extent of errors in the context of the original data values [16].

3. Results and Discussion

The results of the analysis in the form of a comparison of MAPE values on the test data are summarized in Table 4. This table serves as a crucial reference for understanding and comparing the performance of each model in forecasting. From the average MAPE values generated, we can clearly identify that the GRU model achieves the best performance with an average MAPE of 0.0255%. This indicates that GRU can effectively capture and model patterns in time series data, resulting in more accurate forecasts. Following that, the LSTM model has an average MAPE of 0.0377%, showing that LSTM can also handle long-range dependencies in the data. Bidirectional RNN also demonstrates good performance with an average MAPE of 0.0668%. On the other hand, the Simple RNN model has the highest average MAPE value at 0.5118. The comparison plot of the MAPE values for each model can be seen in Figure 8.

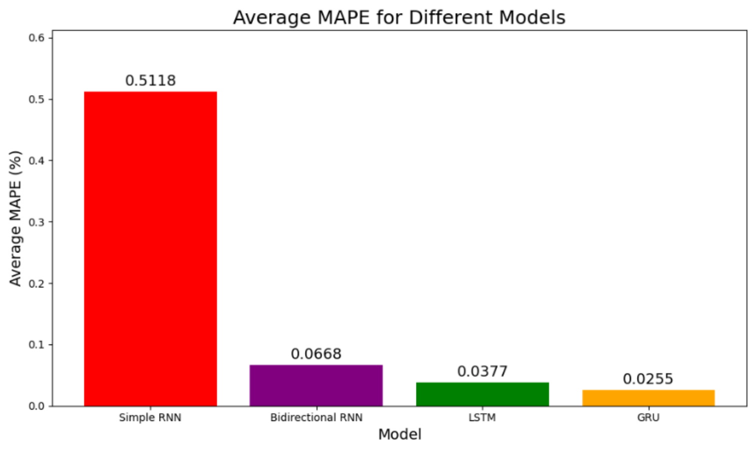

Figure 8. Average MAPE Comparison of Models

There are several reasons why the RNN yields the highest MAPE value. First, RNNs struggle with handling long-range relationships in time series data, leading to difficulties in capturing patterns associated with broader time frames, especially in datasets with complex characteristics. Second, the vanishing gradient problem when dealing with long sequences diminishes the model's adaptability to intricate patterns. Third, the simplistic architecture of RNN makes it less effective in addressing time series data with more complex interactions [17].

Table 4 unveils the crucial impact of the number of units and the learning rate on model performance. Increasing the number of units generally enhances the model's ability to understand patterns within the data, parallel to its complexity increment [18]. However, it's important to note that using a large number of units also leads to longer training times. Furthermore, lower learning rates seem to yield better performance outcomes by enabling the model to adapt more precisely to the data. This table also underscores the importance of adjusting hyperparameters to optimize model performance. The number of units, learning rate, and number of epochs play a pivotal role in the final forecasting outcome. Hence, the selection of hyperparameters must be meticulously done, and aligned with the data characteristics and desired forecasting objectives. Training time efficiency is also a vital consideration in model selection. The table illustrates that certain model configurations may have longer training times. Therefore, apart from performance, time efficiency also needs to be considered, especially in contexts requiring swift responses.

Table 4. Model MAPE Comparison

Model	Units	Learning Rate	Epoch	Total Training Time	MAPE (%)	Model	Units	Learning Rate	Epoch	Total Training Time	MAPE (%)
Simple RNN	32	0,001	40	81,77	0,4841	**LSTM**	32	0,001	77	183,39	0,0557
		0,0001	22	44,72	0,3090			0,0001	65	146,54	0,1329
	64	0,001	24	49,54	0,1320		64	0,001	41	97,09	0,0055
		0,0001	34	68,36	0,8348			0,0001	28	73,13	0,0095
	128	0,001	21	50,86	0,8737		128	0,001	19	74,91	0,0127
		0,0001	24	52,55	0,4370			0,0001	15	46,27	0,0100
Average				57,97	0,5118	Average				103,56	0,0377
Bidirectional RNN	32	0,001	16	56,22	0,3335	**GRU**	32	0,001	24	57,22	0,0060
		0,0001	25	81,95	0,0068			0,0001	65	145,05	0,0092
	64	0,001	15	54,50	0,0051		**64**	**0,001**	**63**	**150,38**	**0,0027**
		0,0001	36	128,45	0,0489			0,0001	17	42,44	0,0159
	128	0,001	14	58,71	0,0035		128	0,001	15	55,07	0,0220
		0,0001	11	48,88	0,0028			0,0001	15	50,35	0,0970
Average				71,45	0,0668	Average				83,42	0,0255

In the analysis of prediction results on the test data, it is evident that the GRU model performs well in following the patterns of the test data. The GRU model with a configuration of 64 units, a learning rate of 0.0001, and the 63rd epoch is chosen as the best model in this study. This configuration yields the smallest MAPE value of 0.0027%. This is observed in the plot illustrating the comparison between the GRU model's prediction results and the actual test data. In this plot, it's apparent that the line representing the GRU model's predictions tends to stay within the range of actual test data values, indicating the model's ability to approximate the actual movement of closing stock prices. In the sequence of data plots, particularly on the last day (the last 242 data points), the GRU model demonstrates exceptional adaptability to the movement of the test data's closing stock prices. This signifies that the GRU model is adept at swiftly responding to fluctuations in the test data and adjusting its predictions according to the actual trends that unfold. This adaptive capability stems from the GRU architecture, designed specifically to handle time series data more effectively.

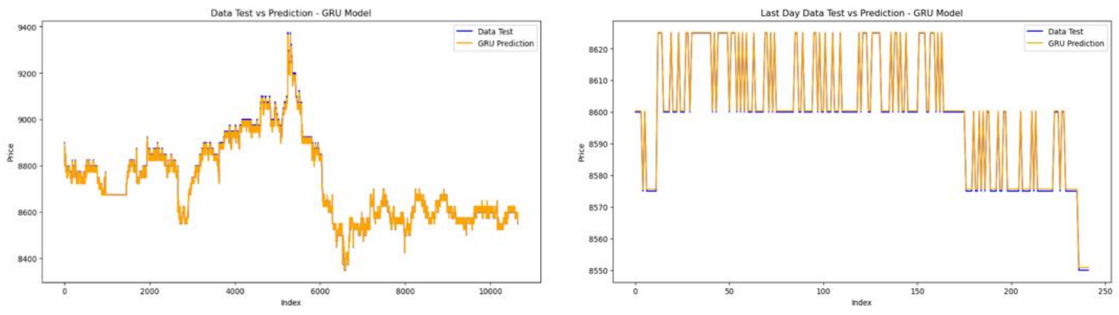

Figure 9. GRU Model Predictions on Test Data

Figure 10 shows the graph of train loss and validation loss of the GRU model with 64 units and a learning rate of 0.001. The train loss graph of the best model demonstrates a steady decreasing trend as the training progresses on the training data, indicating that the model consistently learns the patterns within the training data and minimizes prediction errors. The validation loss graph also exhibits a stable or consistent decreasing trend as the training proceeds on the training data, indicating that the model can generalize well to unseen validation data. An ideal graph is one that shows a steady decreasing trend in both train loss and validation loss, with a small difference between them [19].

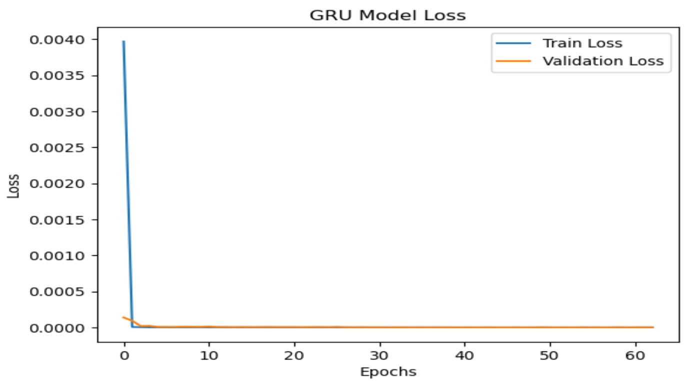

Figure 10. Training and Validation Loss of the GRU Model

Scientifically, the success of the GRU model in following the patterns of the test data can be explained by the gating mechanism inherent in the GRU architecture. The gating mechanism enables the GRU to selectively control the flow of information within the cell, allowing the model to recognize and address changes occurring in time series data. Moreover, GRU also possesses a strong short-term memory capability, allowing the model to retain relevant information from several previous time steps, which may contribute to its adaptive capacity [20]. With the GRU model's ability to closely track test data patterns and exhibit high adaptability, it provides empirical support for the effectiveness of the GRU architecture in forecasting high-frequency time series data such as stock prices.

4. Conclusion

This study presents a comparison of the performance of Simple RNN, Bidirectional RNN, LSTM, and GRU models in forecasting minute-level stock price time series data. The experimental results reveal that the GRU model outperforms the others, exhibiting the lowest average MAPE of 0.0255%, followed by the LSTM model with an average MAPE of 0.0377%. Meanwhile, the Bidirectional RNN model achieves an average MAPE of 0.0668%, and the Simple RNN model has the highest average MAPE of 0.5118%. This indicates that more complex recurrent architectures like GRU and LSTM have better capabilities in capturing patterns in high-frequency time series data. This study could be extended by exploring other models such as CNN, conducting tests on diverse types of datasets, and experimenting with a wider range of hyperparameter variations. Additional variables such as economic indicators, global market data, and social data could provide a more comprehensive understanding of the factors influencing stock prices.

References

[1] F. X. Diebold and G. D. Rudebusch, “Forecasting output with the composite leading index: A real-time analysis,” *J Am Stat Assoc*, vol. 86, no. 415, 1991, doi: 10.1080/01621459.1991.10475085.

[2] T. Donkers, B. Loepp, and J. Ziegler, “Sequential user-based recurrent neural network recommendations,” in *RecSys 2017 - Proceedings of the 11th ACM Conference on Recommender Systems*, 2017. doi: 10.1145/3109859.3109877.

[3] J. Zhao, D. Zeng, S. Liang, H. Kang, and Q. Liu, “Prediction model for stock price trend based on recurrent neural network,” *J Ambient Intell Humaniz Comput*, vol. 12, no. 1, 2021, doi: 10.1007/s12652-020-02057-0.

[4] Z. Li, J. Han, and Y. Song, “On the forecasting of high-frequency financial time series based on ARIMA model improved by deep learning,” *J Forecast*, vol. 39, no. 7, pp. 1081–1097, Nov. 2020, doi: 10.1002/for.2677.

[5] G. Taroon, A. Tomar, C. Manjunath, M. Balamurugan, B. Ghosh, and A. V. N. Krishna, “Employing Deep Learning in Intraday Stock Trading,” in *Proceedings - 2020 5th*

International Conference on Research in Computational Intelligence and Communication Networks, ICRCICN 2020, 2020. doi: 10.1109/ICRCICN50933.2020.9296174.

[6] I. Jebli, F. Z. Belouadha, M. I. Kabbaj, and A. Tilioua, "Deep learning based models for solar energy prediction," *Advances in Science, Technology and Engineering Systems*, vol. 6, no. 1, 2021, doi: 10.25046/aj060140.

[7] J. Qiu, B. Wang, and C. Zhou, "Forecasting stock prices with long-short term memory neural network based on attention mechanism," *PLoS One*, vol. 15, no. 1, 2020, doi: 10.1371/journal.pone.0227222.

[8] D. Saravanan and K. Santhosh Kumar, "Improving air pollution detection accuracy and quality monitoring based on bidirectional RNN and the Internet of Things," *Mater Today Proc*, 2022, doi: 10.1016/j.matpr.2021.04.239.

[9] R. Gençay, M. Dacorogna, U. A. Muller, O. Pictet, and R. Olsen, *An introduction to high-frequency finance*. Elsevier, 2001.

[10] Frank J. Fabozzi, *Encyclopedia of Financial Models*. Wiley, 2012. doi: 10.1002/9781118182635.

[11] T. G. Andersen, "Some Reflections on Analysis of High-Frequency Data," *Journal of Business & Economic Statistics*, vol. 18, no. 2, p. 146, Apr. 2000, doi: 10.2307/1392552.

[12] I. Vasilev, *Advanced Deep Learning with Python*. 2019.

[13] I. Oxaichiko Arissinta, I. Dwi Sulistiyawati, and D. Kurnianto Iqbal Kharisudin, "Pemodelan Time Series untuk Peramalan Web Traffic Menggunakan Algoritma Arima," *Prosiding Seminar Nasional Matematika*, vol. 5, 2022.

[14] X. Ying, "An Overview of Overfitting and its Solutions," in *Journal of Physics: Conference Series*, 2019. doi: 10.1088/1742-6596/1168/2/022022.

[15] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*. 2013.

[16] J. Purnama and A. Juliana, "Analisa prediksi indeks harga saham gabungan menggunakan metode ARIMA," *Cakrawala Management Business Journal*, vol. 2, no. 2, pp. 454–468, 2020.

[17] I. Nasirtafreshi, "Forecasting cryptocurrency prices using Recurrent Neural Network and Long Short-term Memory," *Data Knowl Eng*, vol. 139, 2022, doi: 10.1016/j.datak.2022.102009.

[18] M. A. Hannan, D. N. T. How, M. Bin Mansor, M. S. Hossain Lipu, P. Ker, and K. Muttaqi, "State-of-Charge Estimation of Li-ion Battery Using Gated Recurrent Unit with One-Cycle Learning Rate Policy," *IEEE Trans Ind Appl*, vol. 57, no. 3, 2021, doi: 10.1109/TIA.2021.3065194.

[19] Wei-Meng Lee, "Introduction to Deep Learning," https://codemagrefresh.azurewebsites.net/Article/2003071/Introduction-to-Deep-Learning.

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