VNU Journal of Science: Com. Science & Com. Eng., Vol. 42, No. 1 (2026) 1-22

VNU Journal of Science:
Computer Science and Communication Engineering

Journal homepage: http://www.jcsce.vnu.edu.vn/index.php/jcsce

Original Article

Forecasting the Stock Price in Vietnam: An Application of the Bidirectional Long Short-Term Memory Neural Network

The Long Nguyen

Vietnam Export Import Commercial Joint - Stock Bank,
24B Truong Dinh Street, Xuan Hoa Ward, Ho Chi Minh City, Vietnam

Received 16th September 2024
Revised 22nd January 2025; Accepted 23rd May 2025

Abstract. Accurate stock market prediction can minimize investment risks for investors, improve the efficiency of investment returns, and promote the stable development of the market. However, the high frequency and substantial noise of stock price sequences make precise forecasting a challenging task. Machine learning and deep learning, including enabling computers to perform tasks that typically require human intelligence, are now the dominant trends in stock market forecasting. In this paper, a Bidirectional Long Short-Term Memory (BiLSTM) neural network from deep learning is applied to predict stock price trends in the Vietnamese stock market. The forecasting performance of the one-directional Long Short-Term Memory (LSTM) network and the Recurrent Neural Network (RNN) is compared. Using VN-Index data from 2010 to 2024 and technical indicators, including the Simple Moving Average (SMA), the Moving Average Convergence Divergence (MACD), and the Relative Strength Index (RSI), results show that the BiLSTM model achieves the highest prediction accuracy, nearly 99%. It effectively captures both past and future data information, predicting both short-term and long-term dynamic trends of financial time series. This demonstrates the suitability of the BiLSTM model for stock price forecasting in Vietnam.

Keywords: BiLSTM, deep learning, forecasting, LSTM, RNN, stock market, VN-Index.

1. Introduction

The stock market has existed for more than 400 years and serves as an effective channel for companies to raise capital [1]. By issuing shares, a large amount of capital flows into the stock market, promoting capital concentration, improving the capital structure of enterprises, and strongly driving the development of the

* Corresponding author.
E-mail address: long.nthe@gmail.com

https://doi.org/10.25073/2588-1086/vnucsce.3612

commodity economy. Consequently, the stock market is regarded as a barometer of economic and financial activities within a country or region [2].

Although it was established later than the stock markets of other countries, the Vietnamese stock market has experienced significant development since it began operations in 2000, becoming an important part of the national financial system. However, like many other emerging markets, the Vietnamese stock market faces high levels of volatility, rapid growth, and is more difficult to predict compared to developed markets. Stock prices are influenced by various factors, such as changes in national policies, domestic and international economic environments, global political situations [3], [4], which often lead to non-linear fluctuations in stock prices. While stock price forecasting is crucial for investors, traders, and policymakers, as it helps make more informed investment decisions and contributes to the overall stability and efficiency of the market, these characteristics make stock price forecasting in Vietnam particularly challenging.

Mainstream financial time series forecasting methods are primarily divided into two categories: traditional time series models and machine learning-based forecasting methods. Traditional time series models assume specific models to describe time series, which can be applied in certain cases. For example, Ariyo used the AutoRegressive Integrated Moving Average (ARIMA) model to forecast stock prices of the New York Stock Exchange and the Nigerian Stock Exchange and found that this model has potential for short-term forecasting [5]. Although widely used, traditional time series forecasting methods such as AutoRegressive (AR), Seasonal Naïve, ETS, and ARIMA are designed to fit individual time series [6]. However, when using traditional models to forecast financial time series, practitioners must manually select specific trends, seasonal components, and other data elements. Moreover, actual stock data are highly complex, nonlinear, and noisy, which cannot be captured by analytical equations with parameters. As a result, while traditional forecasting methods are valuable, they often struggle to address the inherent complexities of financial markets.

In recent years, machine learning and deep learning models have become prominent in the field of financial forecasting due to their ability to handle large datasets and identify nonlinear, nonstationary, and multidimensional complex relationships. Mainstream machine learning models primarily include Artificial Neural Networks (ANN), Support Vector Machines (SVM), Random Forests (RF), and other models. Trafalis applied SVM to forecast IBM’s stock prices and found that it performed better than BP neural networks and RF [7]. FengLi forecasted the Shanghai Composite Index using BP neural networks and demonstrated the model's effectiveness for short-term predictions [8]. However, machine learning algorithms often have simple structures, poor generalization abilities, and are prone to falling into local optima, which limits their effectiveness when handling raw data [9].

Deep learning generates higher-level abstract features by combining simple yet nonlinear modules, thereby learning to efficiently represent features from large amounts of input data [10]. Among these, RNN have shown potential in time series forecasting as they are designed to recognize patterns in sequential data. RNNs can account for short-term correlations in time series, and their hidden layers not only receive current data but also incorporate information from previous data, theoretically allowing them to utilize data from any time interval [9]. However, Hochreiter identified the long-term dependency problem in RNNs. When learning sequences, the vanishing and exploding gradient phenomena occur, leading to an inability to capture nonlinear relationships over longer time spans.

Based on this, Hochreiter and Schmidhuber proposed the LSTM, an advanced variant of RNN that mitigates the vanishing and exploding gradient issues through its gating mechanisms

[11]. LSTM has proven effective in various financial forecasting applications, including stock price prediction. Ouyang Hongbing combined wavelet analysis with LSTM to forecast the Dow Jones Industrial Average, improving the model's generalization ability [12]. Fischer & Krauss have also employed LSTM for stock market index predictions [13]. Overall, standard LSTM networks process sequences in a time-ordered fashion, with information flowing in a single direction. As a result, LSTM can only capture and utilize past information, limiting its ability to fully grasp the dynamic nature of financial time series, particularly in a volatile environment like the Vietnamese stock market.

To address these limitations, the BiLSTM network has been proposed. BiLSTM consists of two LSTM models with opposing directions, allowing the model to capture patterns and dependencies in both past and future data. During training, the process involves not only forward propagation from input to output but also backward propagation from output to input. In BiLSTM, each unit is split into two units with the same input, connected to the same output. This bidirectional approach enhances the model's ability to recognize patterns and complex relationships within the data, potentially leading to more accurate forecasts with large-scale time series data [14]. Cheng proposed using a hybrid model of Support Vector Regression (SVR) and BiLSTM for electricity price forecasting, incorporating optimization mechanisms, with promising forecasting performance [15]. Wang et al. employed the BiLSTM model as the foundational framework for short-term load forecasting, applying weighting through an attention mechanism, which ultimately reduced RMSE and MAPE, demonstrating the model's high accuracy [9].

Given the characteristics of stock price data, such as non-linearity, high frequency, complex structural relationships, and the interplay between past and future data, the BiLSTM model theoretically captures the intrinsic relationships within stock price time series effectively. However, current research rarely explores the effectiveness of BiLSTM in stock price prediction, particularly in the context of Vietnam. Addressing the specific challenges posed by the Vietnamese stock market and leveraging the proven advantages of BiLSTM in handling complex time series data, this study aims to explore the application of BiLSTM for stock price forecasting in Vietnam. By comparing the performance of BiLSTM with traditional LSTM and RNN models, this research demonstrates the superiority of BiLSTM in capturing the intricate dynamics of the Vietnamese market. Additionally, this study will provide insights into the practical application of deep learning models in emerging markets, contributing valuable knowledge to the field of financial forecasting.

The main contributions of this paper are as follows: First, it analyzes the applicability of the BiLSTM model to financial time series, specifically stock prices. Second, it investigates the prediction of financial time series using BiLSTM and explores the effectiveness of incorporating a backward LSTM layer into the LSTM layer. Finally, the proposed BiLSTM model is applied to stock price forecasting in Vietnam using the VN-Index to assess its effectiveness in practical applications. This model predicts the closing price for the next trading day by considering various input factors, including opening price, high price, low price, volume, closing price, and technical indicators such as SMA, MACD, and RSI. The model is evaluated using metrics such as Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and R-squared (R^2). Compared to traditional LSTM and RNN models, the BiLSTM model demonstrates the potential to achieve the highest prediction accuracy, with a probability of up to nearly 99%.

2. Related Work

In the recent past, Neural Networks (NN) were often criticized by many forecasting

practitioners as unsuitable and non-competitive in forecasting fields [16]. As a result, practitioners frequently opted for statistical methods, which were considered easier to apply [6]. However, due to the non-linear nature and high noise of financial data, traditional statistical models often fail to capture the characteristics of the data, leading to suboptimal forecasting performance. Consequently, some research has addressed the use of machine learning in quantitative finance, including portfolio management predictions and various investment processes that can be covered by machine learning algorithms. A large body of research is currently underway on machine learning methods used in finance. Moritz & Zimmermann employed tree-based models to predict portfolio returns [17]. Paiva et al. conducted stock return forecasting using a single decision model for day trading on the stock market, utilizing SVM and Mean Variance (MV) methods for portfolio selection [18]. Emerson et al. discussed a wide range of trends and applications of machine learning in quantitative finance [19], including return forecasting, portfolio construction, ethics, fraud detection, decision-making, natural language processing, and sentiment analysis. Wang et al. combined Decision Tree (DT) algorithms with SVM models. They initially filtered most noise data using DT algorithms and then processed the training data in the second stage using SVM to predict future price trends [20]. Le Hong Hanh et al. applied machine learning with text mining techniques to forecast the VN-Index in Vietnam. Utilizing data from 70,000 articles, the study employed models such as RF, decision trees, KNN, and SVM. The results indicate that SVM is the most effective model for predicting Vietnam's stock prices based on financial news [21].

However, with the continuous increase in data volume, neural networks and deep learning have revolutionized and achieved significant success in various research and practical fields, including medical prediction, natural language processing (NLP), and image recognition, among others. Thanks to their ability to recognize complex non-linear patterns and uncover previously unidentified relationships, these technological breakthroughs have garnered considerable attention from the research community, leading to the development of numerous complex and novel neural network architectures for time series forecasting. In recent decades, there has been a substantial amount of research and work employing deep learning for forecasting, including stock price and exchange rate prediction [22] Consequently, AI applications are becoming increasingly popular among investors to enhance profits and mitigate risks [23]. Heaton et al. discussed deep learning models for intelligent indexing [24].

Selvin et al. demonstrated how deep neural network architectures can capture underlying dynamics and be employed for forecasting [25]. Guresen et al. used multilayer perceptrons (MLP), dynamic artificial neural networks, and hybrid models to predict the NASDAQ index [26]. Nayak et al. applied the Artificial Chemical Reaction Optimization (ACRO) algorithm to train a multilayer perceptron (MLP) for predicting stock market indices [27].

Since its introduction by Hochreiter and Schmidhuber [11], the LSTM network, a variant of RNN, has become the most widely used architecture for time series forecasting [28]. Unlike traditional RNNs, LSTMs are designed to detect long-term dependencies and address the vanishing gradient problem. They leverage historical information through input, forget, and output gates. In the study by Nikou et al. [29], LSTM was employed to predict the closing price of the iShares MSCI United Kingdom Index, performing significantly better than ANN, SVR, and RF. LSTMs were also utilized in another study to forecast future stock returns [30]. Additionally, the ARIMA model combined with LSTM has been used to enhance forecasting accuracy. According to Nelson, Pereira, and De Oliveira [31], the average accuracy for predicting the direction of several stocks traded on the Brazilian stock exchange can reach up to 55.9% using LSTM. Xiao et al. compares the accuracy of the ARIMA model and LSTM,

illustrating these techniques in time series forecasting. The results, obtained from a financial dataset, show that LSTM significantly outperforms ARIMA [32].

Gülmez argued that LSTM models are well-suited for time series data in financial markets, particularly where stock prices are influenced by supply and demand relationships [33]. The research, which focused on the Dow Jones index, stock markets, bonds, and other securities in the U.S., also included stock predictions for the period from 2019 to 2023. Another study by Usmani Shamsi [34] on the Pakistani stock market, which examined general market news categories, industry-specific information, and their impact on stock price predictions, confirmed that LSTM models are increasingly being used for recent stock price forecasting. Phuoc et al. [35] forecasted stock price trends in the Vietnamese stock market using the LSTM algorithm. The research findings indicate that the forecasting model achieved a high accuracy of 93% for most of the stock data utilized, demonstrating the suitability of the LSTM model and the test dataset employed to evaluate the model's performance.

Thanks to its ability to learn from both past and future data, BiLSTM stands out as an advanced method in financial forecasting. Numerous studies have demonstrated BiLSTM's advantages over traditional models. For example, Zhang et al. [36] affirmed that applying BiLSTM to stock price prediction improved forecasting performance. Their research also indicated that BiLSTM can effectively handle high-volatility financial data, a critical characteristic of stock markets.

Moreover, Wang and Yang applied BiLSTM to forecast financial time series and observed that this model outperformed others, particularly in volatile market conditions [37]. Zeng et al. used BiLSTM to predict the S&P 500 index, showing that BiLSTM yielded more accurate predictions compared to existing forecasting models. Wu et al. focused on utilizing deep learning models, especially LSTM and BiLSTM, for stock price forecasting based on time series data [38]. Istiake Sunny et al. compared BiLSTM with LSTM and RNN for network-wide traffic prediction, demonstrating BiLSTM's superior performance in capturing temporal dependencies and improving prediction accuracy, especially with large datasets [39]. Han & Fu applied BiLSTM to historical stock price data of Apple Inc., revealing that BiLSTM could make accurate predictions on test data and capture trends and patterns, although it may struggle with sudden market changes [40]. Liu et al. used BiLSTM to forecast financial time series, including exchange rates and interest rates. This study highlighted BiLSTM's strengths not only in stock forecasting but also in other financial domains, particularly when dealing with volatile and rapidly changing time series [41]. These results underscore BiLSTM's significant potential in financial time series forecasting, especially in the context of complex and volatile stock market data.

3. Methodology

3.1. Methodology Overview

The overview of methodology we applied is outlined in Figure 1. It comprises seven stages divided into two main processes as highlighted in the figure. The first process involves extracting VN-Index data from the Ho Chi Minh City Stock Exchange, followed by data cleaning and normalization. The outcome of this process is that we obtain data suitable for machine learning algorithms. Subsequently, we select five features (opening price, lowest price, highest price, previous closing price, and trading volume) and calculate technical indicators (SMA, MACD, RSI) to be used by the model. Next, the data is classified into non-overlapping batches and fed into the model until the performance metrics are optimized. Finally, the optimized model is employed to forecast future VN-Index closing prices.

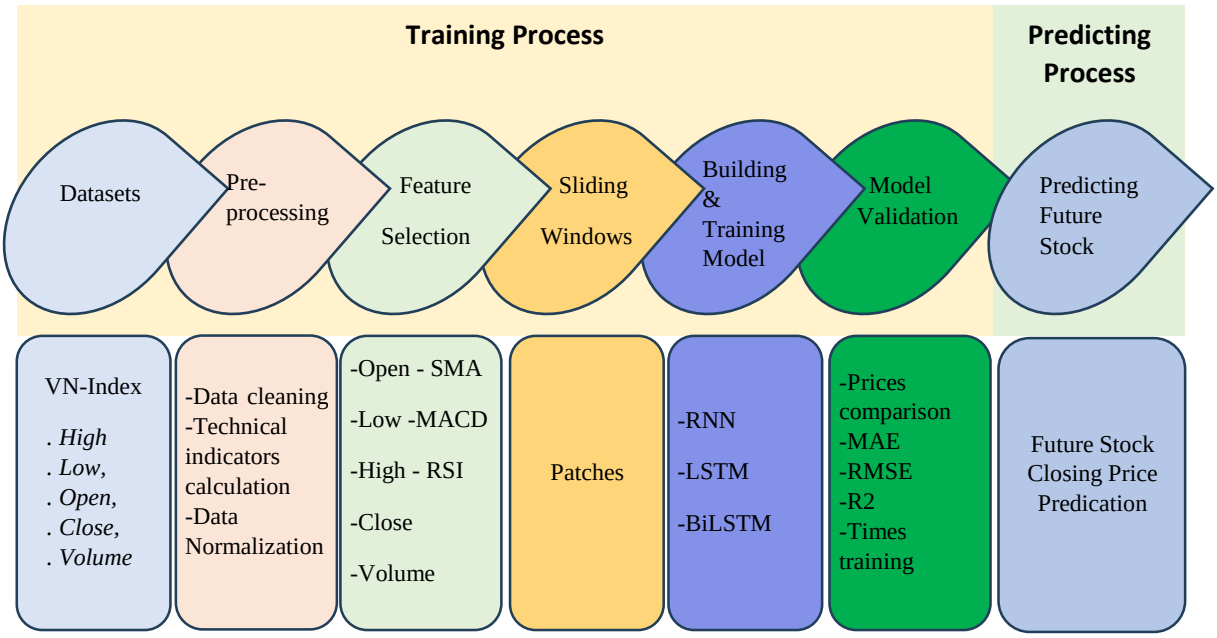

Figure 1. Overview of Methodology.

3.2. Method

Recurrent Neural Networks (RNN)

RNNs have become a crucial tool for analyzing and forecasting sequential data, particularly in fields such as natural language processing, pattern recognition, and time series forecasting. The key capability of RNNs lies in their ability to process and learn from sequentially structured data by maintaining a hidden state to remember information from previous steps. This is especially important when data exhibits temporal dependencies and interrelated connections over time.

RNNs predict future values based on preceding observations, which is why they are referred to as Recurrent Neural Networks. The earlier stages of data need to be remembered to forecast and estimate future values, in this context, the hidden layer functions as a repository for past information from sequential data. The term "recurrent" is used to describe the process of leveraging factors from previous sequences to forecast future data.

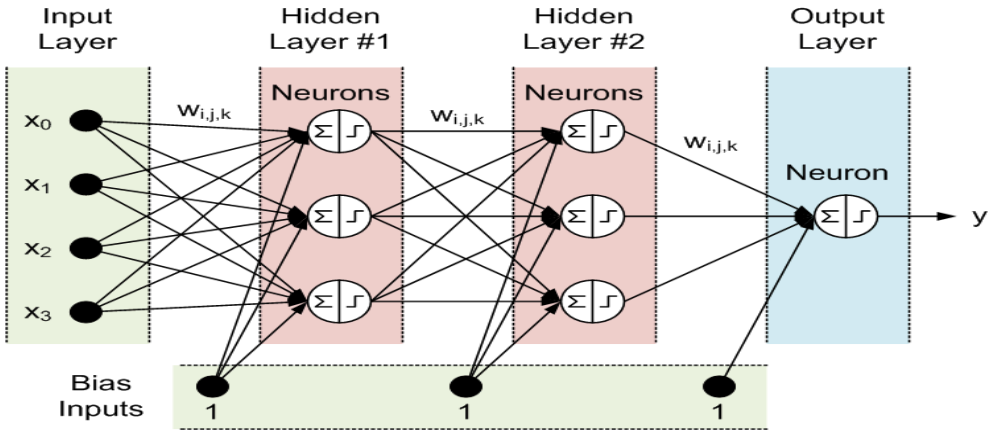

Figure 2. RNN hidden layer. Source: Zhu (2020)[42].

The goal of an RNN is to process sequential data. In traditional neural network models, layers are fully connected from input to hidden to output, with nodes between layers remaining unconnected. However, this standard neural network structure is inefficient for many problems. RNNs are called recurrent because the current output of a sequence is also influenced by previous outputs. Specifically, the network retains prior information and applies it to compute the current output. In other words, nodes between hidden layers are no longer unconnected but are linked, and the input to the hidden layer includes not only the output from the input layer but also the output from the previous time step of the hidden layer. In theory, RNNs can handle sequential data of any length.

The structure of the hidden layer in an RNN is depicted in Figure 2. Here, t represents time, x denotes the input layer, h indicates the hidden layer, and y signifies the output layer. The value h of the hidden layer in an RNN depends not only on the current input x but also on the value h from the previous time step of the hidden layer, as illustrated in Figure 3.

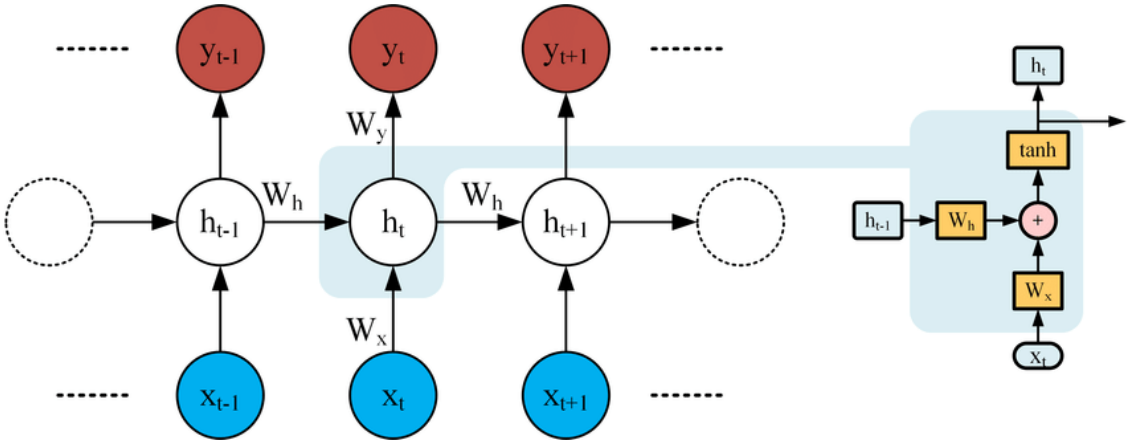

Figure 3. RNN hidden layer calculate process. *Source: Zhu (2020)[42].*

The RNN model is designed with a set of recurrent units, where each unit is responsible for maintaining and updating the hidden state through linear operations combined with nonlinear activation functions. Specifically, at each time step t, the hidden state h_t is computed as follows:

$$h_t = \tanh(W_x x_t + W_h h_{t-1} + b_h) \quad (1)$$

$$y_t = \phi(W_y h_t + b_y) \quad (2)$$

The matrices W_x, W_y, and W_h represent weight matrices, b is the bias parameter, and $tanh$ is the nonlinear activation function commonly used to limit the hidden state values, while ϕ is typically the softmax function, used to compute the probabilities of the output classes.

The RNN training process uses the Backpropagation (BP) algorithm. During training, the parameters W_x, W_y, and W_h are shared, unlike traditional fully connected neural networks. When using the gradient descent algorithm, the output at each step depends not only on the current step but also on the network's previous states.

One of the main advantages of RNNs is their ability to share weights across time steps, which reduces the number of parameters that need to be learned and allows the model to identify repeating patterns or trends in sequential data. This makes RNNs ideal for time series forecasting problems, where observations at each point in time depend on previous observations.

However, basic RNNs face several challenges, such as the vanishing gradient and exploding gradient problems, especially when dealing with long data sequences. These issues make model optimization difficult, reducing the ability to learn long-term patterns in the data. The primary reason for the difficulty in training RNNs is the transmission of the hidden parameter ω. Since error propagation in RNNs is not handled, the value of ω multiplies in both forward and backward propagation. First, the vanishing gradient problem occurs when the gradient becomes small, decreasing exponentially, and almost has no effect on the output. Second, the exploding gradient problem, on the other hand, occurs when the gradient becomes large, exponentially multiplying, leading to a gradient explosion. Although this issue exists in any deep neural network, it is particularly evident due to the recursive structure of RNNs. Moreover, RNNs differ from traditional neural networks in that they do not only have unidirectional neural connections, meaning that neurons can pass data to previous layers or layers of the same type. This lack of storing information in a single direction is a practical characteristic of the existence of short-term memory, alongside long-term memory that neural networks acquire through training.

Long Short-Term Memory (LSTM)

LSTM is a neural network model proposed by Hochreiter and Schmidhuber in 1997 [11]. It is designed to address the long-standing issues of gradient explosion and vanishing gradient in RNNs [43]. A standard RNN has only one repeating module, with a simple internal structure, usually consisting of just a tanh layer. However, LSTM features four modules similar to those of standard RNNs, and they operate in a particularly interactive manner [44-46]. The memory of LSTM consists of three components: an input gate, a forget gate, and an output gate, as shown in Figure 4.

LSTM adds memory cell structures and gates to the traditional recurrent neural network. The cell (C_t) is used to record the state of the neurons, while the gates selectively allow information to pass through, utilizing a sigmoid layer and pointwise multiplication mechanism. With three gate structures functioning like filters, LSTM protects and controls information: the forget gate (f_t) decides which information is discarded from the cell state, the input gate (i_t) determines which new information is added to the cell, and the output gate (O_t) filters the final output information. Through its gated structure, LSTM can capture long-term dependencies in the input features, effectively mitigating the vanishing and exploding gradient problems.

In the LSTM module structure shown in Figure 4, x is the input vector of the LSTM model, and h is the output vector. f, i, and O represent the activation values of the forget gate, input gate, and output gate, respectively. C and C' represent the cell state and its candidate values; the index t indicates time; σ and $tanh$ represent the sigmoid and tanh activation functions, respectively; and w and b represent the weight matrix and bias.

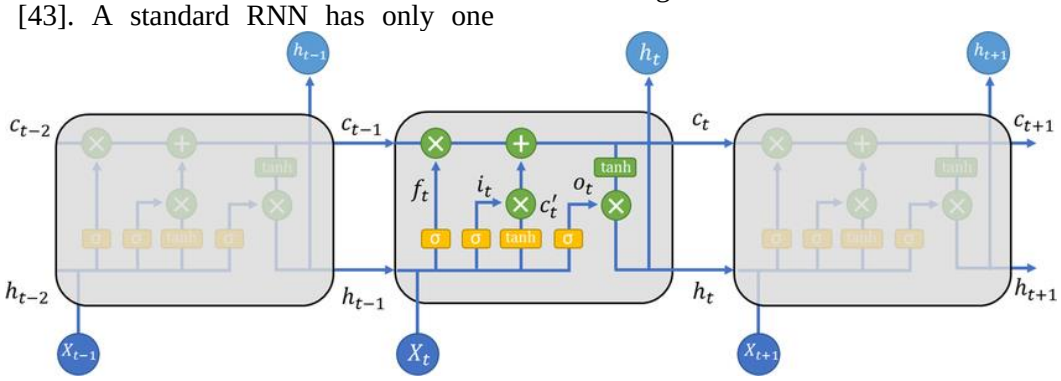

Figure 4. LSTM module structure. *Source: Yang & Wang [37].*

In the first step, the LSTM layer determines which information will be discarded from the previous cell state C_{t-1}. Therefore, the activation value f_t of the forget gate at time ttt is computed based on the input x_t at time t, the output at time $t-1$, and the bias matrix b_f of the forget gate. The sigmoid function normalizes all activation values f_t, i_t, C_t, and O_t between 0 (completely forgotten) and 1 (completely retained).

$$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f) \quad (3)$$

In the second step, the LSTM layer decides which information will be added to the cell state C_t. This step consists of two parts: first, the calculation of C_t', which can be added to the cell state; second, the computation of the activation value i_t of the input gate.

$$\widetilde{C_t} = \tanh(w_C[h_{t-1}, x_t] + b_C) \quad (4)$$

$$i_t = \sigma[w_i[h_{t-1}, x_t] + b_i) \quad (5)$$

In the third step, the current cell state C_t is updated according to the formula:

$$C_t = f_t C_{t-1} + i_t \widetilde{C_t} \quad (6)$$

In the final step, the previous time step's output value and the current time step's input value are fed into the output gate. The output gate's value is computed using the formula:

$$O_t = \sigma[w_O[h_{t-1}, x_t] + b_O) \quad (7)$$

The LSTM output value is determined by calculating the output of the output gate and the cell state, as shown in the formula:

$$h_t = O_t \tanh(C_t) \quad (8)$$

The output value h_t is a combination of the output from the output gate and the current cell state, adjusted through the tanh activation function. This allows the LSTM model to store important information from previous time steps and use that information for predictions at the current time step.

LSTMs are distinguished by their ability to remember long-term dependencies in data sequences, making them ideal for tasks such as time series forecasting, speech recognition, and machine translation. The ability to control information through gates enables LSTMs to overcome the vanishing gradient problem, allowing them to learn complex data patterns with varying sequence lengths more effectively.

In the context of time series forecasting, such as stock price prediction, LSTMs are favored for their capacity to model complex and nonlinear relationships in the data.

Bidirectional Long Short-Term Memory (BiLSTM)

BiLSTM is a variant of the LSTM model designed to enhance contextual understanding in data sequences by processing information in both forward and backward directions. This model not only retains information from previous states but also incorporates information from future states, thereby improving forecasting and analysis capabilities for sequential data. This characteristic proves beneficial for tasks such as speech recognition, natural language processing, and stock price prediction.

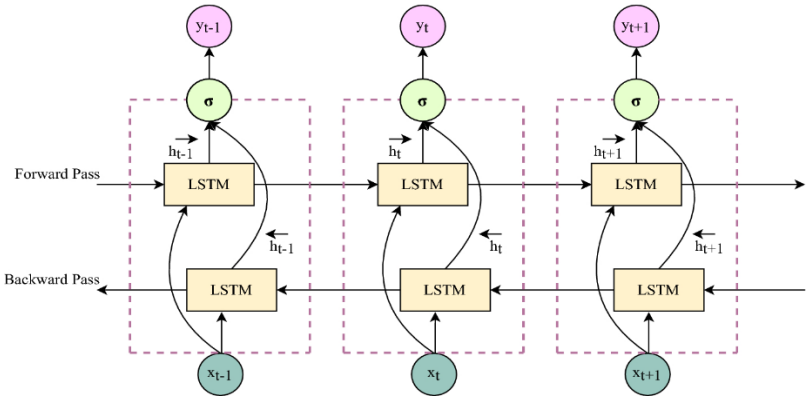

Figure 5. BiLSTM module structure. Source: Yang & Wang [37].

The structure of BiLSTM includes two separate LSTM layers: one layer processes the data sequence in the forward direction (forward LSTM), and another layer processes it in the backward direction (backward LSTM). The outputs of each layer are then combined to produce the final output. The structure of BiLSTM is illustrated in Figure 5. It can be described by the following equation:

Forward LSTM: Processes information from past states to future states

$$\overrightarrow{h_t} = LSTM(x_t, \overrightarrow{h_{t-1}}) \quad (9)$$

Backward LSTM: Processes information from future states to past states

$$\overleftarrow{h_t} = LSTM(x_t, \overleftarrow{h_{t-1}}) \quad (10)$$

Integration of Information: The outputs of the two LSTM layers are typically combined, either through concatenation or addition, to create a richer contextual representation.

$$h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}] \quad (11)$$

$$y = dense\ (h_T) \quad (12)$$

In this context, $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ denote the hidden states of the forward LSTM and backward LSTM at time step t, respectively. x_t is the input at time step t, h_T is the final hidden state, y is the output, and ";" represents concatenation.

One advantage of the BiLSTM model over the basic LSTM is its enhanced capability to capture long-term dependencies within the input sequence. This is due to the backward LSTM layer providing information from the end of the sequence, which can be useful for predicting future values.

In this study, we apply the BiLSTM model to forecast stock prices in the Vietnamese stock market. Historical data on stock prices and technical indicators are used as input for the model, considering both the forward and backward directions of the time series to optimize prediction performance. We train the BiLSTM model on a large dataset and use techniques such as cross-validation to evaluate its performance. The results from the BiLSTM will be compared with unidirectional LSTM models and RNN models to assess improvements in forecasting accuracy. Additionally, this study provides an in-depth analysis of how BiLSTM processes information in both directions, offering valuable insights for applying BiLSTM to similar tasks, especially in forecasting and analyzing complex time series data.

3.3. Input Data

This study uses the BiLSTM model to predict the closing price of the VN-Index, which represents the Vietnamese stock market. The VN-Index has been published since July 28, 2000, marking the beginning of the Vietnamese stock market. To avoid model instability due to significant market fluctuations at the onset and to exclude strong fluctuations during the 2008 financial crisis, data is selected starting from 2010. The dataset used in this analysis includes daily prices such as Close, Open, High, Low prices, and Volume of the VN-Index from January 2, 2010, to July 31, 2024. The initial data are used to compute technical indicators for the model, including SMA, MACD, and RSI. Missing data have been removed, resulting in a total of 3,623 valid data points.

Opening Price is the price of the stock at the first trade of the trading day, after the exchange opens. Highest Price is the highest price of the stock from market open to market close for each trading day. Lowest Price is the lowest price of the stock from market open to market close for each trading day. Closing Price is the weighted average price of transactions in the minute before the close of trading for that day. Volume is the total number of shares traded during the day.

SMA (Simple Moving Average) is a type of moving average calculated by averaging the stock prices over a specific period. For a 14-day period, the SMA is calculated by summing the closing prices of the stock over 14 days and dividing by 14. The formula for calculating the SMA technical indicator is as follows:

$$SMA_{14} = \frac{P_1 + P_2 + P_3 + \cdots + P_{14}}{14} \quad (13)$$

In which P_i is the closing price on the i-th day within the 14-day period.

MACD (Moving Average Convergence Divergence) is a momentum indicator that uses the difference between two moving averages to identify trends and trading signals. Typically, MACD includes a MACD line (the difference between the 12-day Exponential Moving Average (EMA) and the 26-day EMA), a signal line (the 9-day EMA of the MACD line), and a histogram (the difference between the MACD line and the signal line).

The formula for calculating MACD is as follows:

$$MACD_{12,26} = EMA_{12} - EMA_{26} \quad (14)$$

$$Signal_9 = EMA_9(MACD_{12,26}) \quad (15)$$

$$Histogram = MACD_{12,26} - Signal_9 \quad (16)$$

In which, **EMA** is a type of moving average that assigns greater weight to more recent prices, making it more responsive to new information. EMA_{12} is Exponential Moving Average over 12 days and EMA_{26} is Exponential Moving Average over 26 days. The formula for calculating the EMA for a period of n days is as follows:

- Calculate the Smoothing Factor (k):

$$k = \frac{2}{n+1} \quad (17)$$

where n is the number of days or periods for which the EMA is being calculated. For example, to calculate EMA_{12}, $n = 12$.

- Calculate the EMA:

+ EMA for the First Day: The EMA for the first day is typically initialized using the SMA (Simple Moving Average) of the first n days.

$$EMA_{first\ day} = SMA_{first\ day} = \frac{\sum_{i=1}^{n} P_i}{n} \quad (18)$$

where P_i is the closing price on the i-th day

+ EMA for Subsequent Days: The EMA for subsequent days is calculated using the following formula:

$$EMA_t = P_t \times k + EMA_{t-1} \times (1 - k) \quad (19)$$

where, EMA_t is the EMA for day t; EMA_{t-1} is the EMA of the previous day, and P_t is closing price on day t.

RSI (Relative Strength Index) is a momentum indicator that measures the speed and change of price movements. It is used to identify overbought or oversold conditions of a stock. For a 14-day period, RSI is calculated by comparing the average gains and losses over the 14 days.

The formula for calculating RSI is as follows:

$$RSI_{14} = 100 - \frac{100}{1+RS} \quad (20)$$

In which, RS is the average gain to average loss ratio over a 14-day period.

$$RS = \frac{Average\ Gain}{Average\ Loss} \quad (21)$$

where Average Gain is the average of the gains over the 14 days, and Average Loss is the average of the losses over the 14 days.

To calculate the Average Gain and Average Loss:

Average Gain: The average of all gains over the 14-day period.

Average Loss: The average of all losses over the 14-day period.

The RSI value ranges from 0 to 100, with values above 70 typically indicating that a stock is overbought, and values below 30 indicating that it is oversold.

The choice of the parameter *n*-day for indicators such as SMA, MACD, and RSI significantly impacts the model's forecasting accuracy. A shorter period, such as 5 days, increases the sensitivity of the indicators to price changes, capturing short-term fluctuations but potentially introducing noise and reducing forecast stability. Conversely, a longer period, such as 50 days, smooths out price movements, focusing on long-term trends but possibly overlooking important short-term signals. An inappropriate choice of nnn may lead to overfitting or underfitting, adversely affecting the model's generalization capability. Thus, determining an optimal period for these indicators is crucial to enhancing the model's predictive accuracy and ensuring robustness in forecasting stock market trends.

The selection of input indicators for the BiLSTM model, particularly the 14-day period used to calculate the SMA, MACD and RSI, is based on several considerations. First, the 14-day period is widely adopted in technical analysis as it provides a balance between capturing short-term market trends and minimizing noise. This period is commonly used to reflect short- to medium-term price movements while maintaining responsiveness to market fluctuations. In the context of Vietnam's stock market, characterized by high volatility, the 14-day SMA can effectively capture market trends without being overly sensitive to temporary fluctuations. Furthermore, the choice of this period stems from empirical testing, which demonstrates that the 14-day window yields the best forecasting performance. The 14-day period has also been employed by Phuoc et al. in calculating technical indicators for stock price forecasting in Vietnam.

This paper uses a rolling window to compute the indicators for subsequent days.

Table 1 provides a small sample of the dataset. Specifically, the rows correspond to individual trading days and include the following features: trading date, closing price, opening price, highest price, lowest price, trading volume, SMA, MACD, and RSI.

Table 1. VN-Index simple data

Date	Close	Open	High	Low	Volume	SMA	MACD	RSI
2010-01-22	477.59	474.1	481.69	471.85	30080	506.36	-9.000	33.43
2010-01-25	480.91	476.75	481.04	476.75	21330	502.68	-9.65	25.86
---	---	---	---	---	---	---	---	---
2024-07-30	1245.06	1246.6	1248.73	1236.97	653070	1258.95	-8.59	28.55
2024-07-31	1251.51	1245.06	1255.77	1245.06	748850	1256.64	-8.03	33.79

Next, this paper split the dataset into training and testing sets. This paper use the first 80% of the dataset for training and the remaining 20% for testing.

3.4. BiLSTM Training Process

The training process for the BiLSTM model is conducted as follows:

1) Data Normalization

After collecting and computing the input data, we normalize the data before feeding it into the model. Normalization refers to the process of adjusting the range of values within a dataset. Since we use both price data and volume data, which are measured in different units and have large value ranges, normalization is crucial. Machine learning algorithms converge faster and perform better when the features are closer to a normal distribution and/or on the same scale. Additionally, in machine learning algorithms, activation functions like the sigmoid function have a saturation point beyond which the outputs become unchanged. Therefore, prior to training the BiLSTM model, the inputs need to be normalized.

This step is important because it ensures that all input features are on the same scale and prevents the model from being biased towards features with larger values. The normalization process is carried out using the MinMaxScaler from the scikit-learn library. When MinMaxScaler is applied to a feature, it subtracts the minimum value from each value in the feature and scales the result according to the range. As a result, the range of a feature is the difference between its maximum and minimum values. MinMaxScaler preserves the shape of the original distribution. The MinMaxScaler normalizes the input values to be within the range [0,1]. The normalization formula is as follows:

$$x_{norm,i} = \frac{x_i - x_{min}}{x_{max} - x_{min}} \quad (22)$$

where x_i is the original data for day i, $x_{\min}$ and $x_{\max}$ are the minimum and maximum values in the data, respectively, and $x_{\text{norm,i}}$ is the normalized data of x_i.

2) Model Training

The training environment for this model is as follows: MacBook Pro with Apple M2 Pro chip, 16GB LPDDR5 RAM, and macOS Sonoma. Based on the Python programming environment, this study uses TensorFlow as the deep learning framework for training and predicting the model. The Python version is 3.11.8; TensorFlow version is 2.17.0; and Keras version is 3.4.1.

We trained the BiLSTM model using the training dataset. The training process involves providing the model with a sequence of input variables including the opening price, closing price, highest price, lowest price, trading volume, and technical indicators SMA, MACD, RSI, and predicting the next closing price in the sequence. The predicted values are then compared with the actual values, and model parameters are adjusted to minimize the difference between predicted and actual values.

The LSTM layers are the core building blocks of the model and are responsible for processing the input sequence and learning the relationships between input features. The BiLSTM model consists of two LSTM layers: one processing the input sequence in the forward direction and the other in the backward direction. These layers are designed to capture both short-term and long-term dependencies in the data, helping the model learn complex patterns and trends.

The BiLSTM model architecture implemented in this study comprises two stacked BiLSTM layers, each followed by a Dropout layer to mitigate overfitting by randomly deactivating a portion of neurons during training. Each Dropout layer applies a dropout rate of 0.2, which is commonly recommended in prior studies for improving model generalization without significantly hindering learning performance [24].

The final layer is a Dense layer with a single output neuron, which produces a scalar value representing the predicted stock price for the next time step. This design enables the model to capture temporal dependencies while ensuring robustness and prediction stability.

The model is built on the Python platform, as shown in Figure 6. Figure 6 demonstrates that the model is compiled with the following parameters: optimizer='adam', loss='mean_squared_error'.

Layer (type)	Output Shape	Param #
bidirectional (Bidirectional)	(None, 10, 128)	37,376
dropout_4 (Dropout)	(None, 10, 128)	0
bidirectional_1 (Bidirectional)	(None, 128)	98,816
dropout_5 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 1)	129

Total params: 408,965 (1.56 MB)
Trainable params: 136,321 (532.50 KB)
Non-trainable params: 0 (0.00 B)

Figure 6. BiLSTM model.

Figure 6 provides a visual illustration of the entire architecture of the BiLSTM model. Specifically, the input time series is first passed through the initial BiLSTM layer, where temporal features are learned in both forward and backward directions. The output is then fed into the first Dropout layer, followed by a second BiLSTM layer and another Dropout layer. Finally, the output is directed to a Dense layer that generates stock price predictions.

The first BiLSTM layer consists of 128 hidden units (64 in each direction), returning the full sequence output with a shape of (None, 10, 128), where *None* denotes the batch size, *10* is the number of time steps, and *128* is the total number of output features. This layer contributes 37,376 trainable parameters. Subsequently, a Dropout layer is applied to randomly deactivate a subset of outputs, thereby improving the model's generalization ability. Next, the second BiLSTM layer, also with 128 hidden units, returns only the final output of the sequence, resulting in an output shape of (None, 128). This layer adds 98,816 trainable parameters. Another Dropout layer is inserted afterward to further reduce the risk of overfitting. Finally, the output layer is a Dense layer with a single neuron and a linear activation function to produce the predicted stock price. This layer contains 129 trainable parameters. Overall, the model includes 408,965 parameters, of which 136,321 are trainable, and 0 are non-trainable. This architecture strikes a balance between model complexity and robustness, making it suitable for financial forecasting tasks where price dynamics are influenced by complex and nonlinear temporal patterns.

The model's key hyperparameters include the number of hidden units in each BiLSTM layer, batch size, number of training epochs, and the use of early stopping. In this study, each BiLSTM layer is configured with 64 hidden units, enabling the model to effectively capture complex temporal dependencies in financial time series data. Mini-batch gradient descent is adopted for optimization, in which the training data is divided into smaller batches to update model parameters iteratively. A batch size of 32 is selected to strike a balance between convergence speed and model stability, as smaller batch sizes tend to introduce stochasticity that helps avoid local minima. Training is performed over multiple epochs, where each epoch represents one complete pass through the entire training dataset, including both forward and backward propagation. The number of epochs is set to 100, which provides the model with sufficient iterations to learn underlying data patterns without overfitting. This choice aligns with common practice in time-series forecasting literature, where 100 epochs are frequently adopted as a standard due to their balance between learning capacity and computational efficiency. The optimal number of epochs is typically validated by observing training and validation loss curves to ensure convergence without overtraining. To further prevent overfitting and reduce unnecessary computations, this study employs the EarlyStopping technique with the parameters monitor='loss', patience=8, and restore_best_weights=True. This allows training to halt when the model's performance no longer improves, thereby preserving the best-performing parameters and ensuring model generalization.

The Adam optimizer is chosen to optimize the training process, as proposed by Kingma and Ba [47]. The Adam algorithm combines the following advantages: it maintains a learning rate for each parameter to improve performance on sparse gradients. Additionally, the learning rate is adapted for each parameter based on the exponentially weighted moving average of the gradient magnitudes, which results in excellent performance on unstable and nonlinear problems. Therefore, Adam is considered an optimizer with outstanding default performance in many scenarios.

This study uses a learning rate of 0.001 and sets timesteps to 10. This means that the stock price for a given day is predicted based on the stock prices of the previous 10 days. For example, the value for August 11, 2010, is

predicted based on the values from August 01, 2010, to August 10, 2010. The choice of a 10-timestep window is designed to capture sufficient historical context without overwhelming the model with excessive data. This balance is critical, as using a timestep that is too short may fail to capture important temporal dependencies, while a timestep that is too long could lead to overfitting and unnecessary computational complexity.

To compare the predictive performance of the BiLSTM model, this study uses LSTM and RNN models as control variables to predict the closing price of the VN-Index. The parameter settings for the LSTM model are identical to those of the BiLSTM model, except that the BiLSTM layers in the BiLSTM model are replaced with LSTM layers, and the training process is also optimized using the Adam optimizer.

3) Model Evaluation

After training the model, we evaluate its performance using a test set. We input a sequence of closing prices from the test set into the model and predict the subsequent value in the sequence. We then compare the predicted values with the actual values.

Quantitative evaluation of deep learning models is categorized into accuracy metrics, financial metrics, and error metrics [48]. Accuracy and financial metrics are widely used for classification tasks (e.g., predicting price direction), stock trading, and portfolio management. On the other hand, error metrics are commonly used to assess predictions of numerical dependent variables (e.g., exchange rates or stock market forecasts). Error metrics evaluate the difference between actual values y_i and predicted values $\hat{y}_i$ using criteria such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R^2). These metrics help determine the effectiveness and accuracy of the predictive model in estimating the closing price of stocks for the following day. Detailed information about these measures is provided below.

MAE (Mean Absolute Error) measures the average deviation between predicted values and actual values. It quantifies the accuracy of the forecast. MAE can be calculated using the following formula:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}(|y_i - \hat{y}_i| \quad (23)$$

where y_i is the actual closing price and $\hat{y}_i$ is the predicted closing price. A smaller MAE indicates higher accuracy of the model.

MSE (Mean Squared Error) is used to evaluate model performance based on the average forecasting error. The formula for MSE is as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad (24)$$

RMSE (Root Mean Squared Error) is one of the most used error metrics in regression. It is the square root of MSE and measures the dispersion of residuals. Based on the RMSE formula, it can be determined how well the data is concentrated around the optimal line. The optimal RMSE value is close to zero. The formula for RMSE is:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \quad (25)$$

where n is the number of data points, y_i is the actual closing price, and $\hat{y}_i$ is the predicted closing price.

To further explore the accuracy of the prediction models and in conjunction with financial evaluation methods, this study also calculates the goodness-of-fit for each model. Goodness-of-fit is defined as the degree to which the regression line fits the observed values. The coefficient of determination $\mathbf{R^2}$ is used to measure the goodness-of-fit, with values closer to 1 indicating better fit. The formula for the coefficient of determination is:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(\bar{y}_i - \hat{y}_i)^2} \quad (26)$$

where y_i is the actual closing price, $\hat{y}_i$ is the predicted closing price, and $\bar{y}_i$ is the mean value of the closing price series.

4) Determining the Termination Conditions for the Training Process

Successful termination conditions include: completing a predefined number of iterations, having weights lower than a certain threshold, and achieving a prediction error rate below a specified threshold. If at least one of these termination conditions is met, the training process will end. Otherwise, the training will continue.

Backpropagation of Error: The calculated error is propagated backward through the network, updating the weights and biases of each layer. Subsequently, the process will return to step (2) to continue training the network.

Similar evaluation procedures are also applied to the LSTM and RNN models for comparison with the BiLSTM model.

3.5. BiLSTM Prediction Process

The prerequisite for forecasting with BiLSTM is that the BiLSTM model has completed the training process. The BiLSTM forecasting process is carried out as follows:

Forecasting: Normalized data is fed into the trained BiLSTM model to obtain the corresponding output values.

Restoring Normalized Data: The output values from the BiLSTM are normalized. These values are restored to their original scale using the formula:

$$x_i = x_{min} + y_i(x_{max} - x_{min}) \quad (27)$$

where x_i is the restored value, y_i is the output value from BiLSTM, and x_{max} and x_{min} are the maximum and minimum values of the input data.

Output Results: The restored results are outputted to complete the forecasting process. Similar procedures are applied to the LSTM and RNN models for comparison with the BiLSTM.

4. Results

The processed training data was used to train the RNN, LSTM, and BiLSTM models. The trained models were then used to predict the test data, and the actual values were compared with the predicted values. Before visualizing the model's prediction results and computing the loss function, the predicted data was denormalized to compare it with the original closing price labels and assess the model's performance.

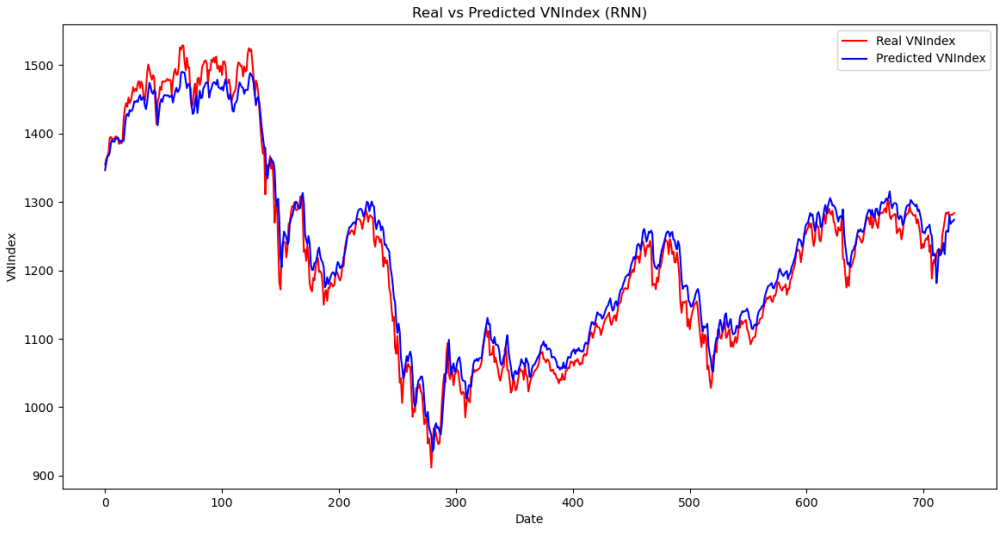

Figure 7. Predicted and Actual Values of the VN-Index Using the RNN Model.

Figures 7-9 respectively show the comparison between the predicted closing prices of the VN-Index and the actual closing prices using the RNN, LSTM, and BiLSTM models. These charts illustrate the best results based on the comparison between the actual VN-Index values and the predicted values (closing prices). In each chart, the red and blue lines represent the actual and predicted values, respectively. The charts provide a timeline of the entire dataset.

Note from the figures that there is a relatively larger discrepancy between the actual and predicted closing prices of the VN-Index during the early stages of the data. However, the discrepancy decreases in the later stages. Overall, all three figures demonstrate a relatively small difference between the actual and predicted values, indicating that the deep learning models perform well. Our research results suggest that the proposed model is highly effective in analyzing and capturing trends, as well as predicting them accurately.

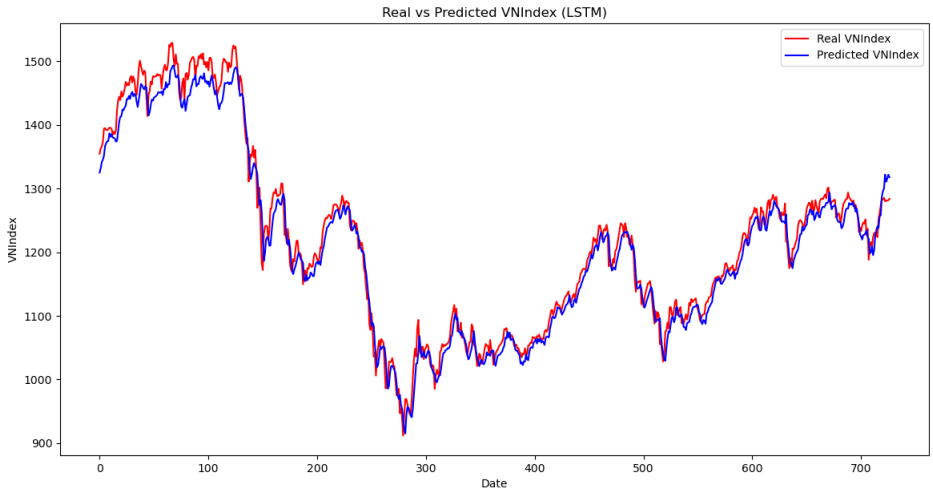

Figure 8. Predicted and Actual Values of the VN-Index Using the LSTM Model.

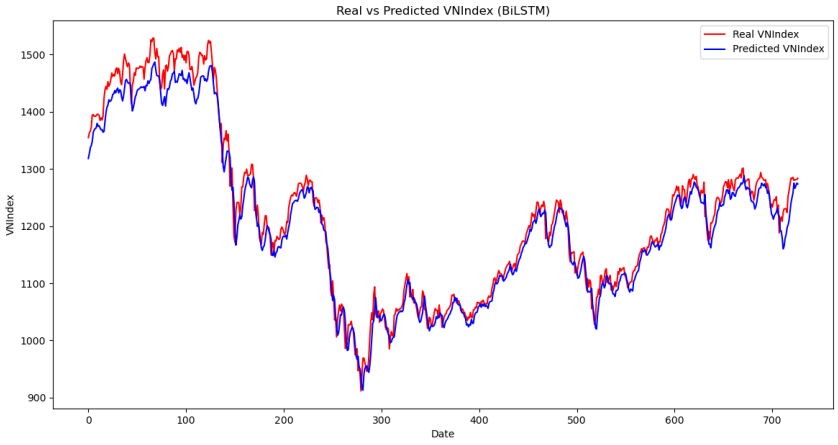

Figure 9. Predicted and Actual Values of the VN-Index Using the BiLSTM Model.

Based on the predicted values from each method and the actual values, the error metrics for each method were calculated, and the comparison results of the three methods are presented in Table 2. The results in Table 2 show that the MAE and RMSE of the RNN model are the highest among the three models, indicating a relatively large model loss. According to Figure 7, with the RNN model, the predicted closing prices are generally lower than the actual closing prices. The LSTM model performs better than the RNN model, possibly because the LSTM's recurrent structure can describe the dynamic changes in the data more effectively. In modeling the current timestep data, the information from the previous timestep is utilized. The storage unit and gate mechanisms can quickly and collaboratively update the contents and dynamics within the cells, effectively addressing the issue of long-term data dependencies, thus resulting in more accurate predictions.

The proposed BiLSTM model achieved the minimum RMSE and MAE values of 17.296 and 12.672, respectively. Additionally, the BiLSTM model also achieved the best fit with an R^2 value of 0.9862, indicating that the model can effectively balance the data between the training and test sets with the highest prediction accuracy. Furthermore, as the BiLSTM layer can capture both spatial and temporal dependencies from historical data in both forward and backward directions, the prediction accuracy of the BiLSTM model has been significantly improved compared to the unidirectional LSTM. This suggests that processing data in both forward and backward directions with the multi-layer structure can exploit the data more effectively and capture more information.

Table 2. Model Performance Evaluation

Model	Training Time (second)	MAE	RMSE	R2
RNN	18.164424	14.281684	18.934371	0.983527
LSTM	37.564810	12.907434	17.711889	0.985586
BiLSTM	81.523974	12.671913	17.296250	0.986254

To explore the feasibility of the model in practical applications, this study also recorded the training time of each model to assess whether the model can be trained efficiently in practice. The training time is based on the operating environment: MacBook Pro with Apple M2 Pro chip, 16GB LPDDR5 RAM, and macOS Sonoma operating system.

Based on the training times of the models presented in the table, it is evident that while the BiLSTM model offers high prediction accuracy, it also comes with distinct drawbacks. Compared to the one-dimensional LSTM model, the BiLSTM model requires more training time. This indicates that when dealing with large and high-dimensional datasets, BiLSTM may sacrifice some efficiency in order to improve model accuracy.

In summary, the BiLSTM model demonstrates high prediction accuracy when applied to financial time series forecasting. It not only surpasses the linear single-dimensional data descriptions of traditional time series models but also addresses issues related to overfitting and long-term data dependencies commonly faced by shallow learning models. Furthermore, it leverages both past and future data by incorporating bidirectional information, enhancing data structure and information extraction more effectively.

However, the added bidirectional structure increases the training time of the BiLSTM model compared to one-dimensional LSTM, shallow learning models, and linear models, leading to lower training efficiency. Therefore, when selecting a model for prediction tasks, it is

important to balance between prediction efficiency and accuracy, choosing the most suitable model framework based on the specific requirements of the application.

5. Conclusions

Due to characteristics such as high frequency, non-linearity, and significant noise, predicting financial time series remains a focal point and challenge in academic research. Therefore, theoretical perspectives suggest that artificial neural network-based analysis and prediction methods are more suitable compared to existing statistical methods. To understand anomalies in time series data, this study explores the practical value of the advanced deep learning BiLSTM model in financial time series forecasting and compares the performance of BiLSTM, LSTM, and RNN models in financial time series prediction. Through empirical testing with the closing prices of the VN-Index, the study finds that BiLSTM exhibits strong applicability in forecasting highly noisy financial time series.

The study draws the following conclusions: First, the BiLSTM learning method can incorporate both past and future time series data as input and produce the final predicted value. BiLSTM achieves the lowest RMSE and MAE, and the highest fit among the models, indicating that BiLSTM can extract more data information with its bidirectional layer compared to LSTM, thereby improving the model's prediction accuracy. Given its outstanding performance in predicting the VN-Index closing prices, the study suggests that BiLSTM has certain advantages in forecasting non-linear and noisy financial time series. Second, deep learning models, including BiLSTM, LSTM, and RNN, exhibit various levels of non-linear operation, allowing deep learning to leverage data more effectively, leading to superior prediction accuracy in financial time series forecasting and stock price prediction. Third, despite its advantages, the BiLSTM model has some limitations. Due to the added bidirectional structure on top of LSTM, the training time for the model is longer, and the operational efficiency decreases. When researchers and investors select prediction models, they may need to balance between model training efficiency and prediction accuracy to choose a more suitable training framework.

Future research could focus on optimizing model parameters to improve result accuracy. Additionally, future studies should explore the applicability of this model in other time series forecasting fields, such as predicting gold prices, oil prices, weather conditions, and earthquakes, as well as other potential applications.

References

[1] Z. Jin, Y. Yang, Y. Liu, Stock Closing Price Prediction Based on Sentiment Analysis and LSTM, Neural Comput Appl, Vol. 32, No. 13, 2020, pp. 9713–9729, https://doi.org/10.1007/s00521-019-04504-2.

[2] L. Badea, V. Ionescu, A.-A. Guzun, What is the Causal Relationship between STOXX EUROPE 600 Sectors? But between Large Firms and Small Firms? Econ Comput Econ Cybern Stud Res, Vol. 53, No. 3, 2019, pp. 5–20, https://doi.org/10.24818/18423264/53.3.19.01.

[3] K. Kohara, Y. Fukuhara, Y. Nakamura, Selective Presentation Learning for Neural Network Forecasting of Stock Markets, Neural Comput Appl, Vol. 4, No. 3, 1996, pp. 143–148, https://doi.org/10.1007/BF01414874.

[4] K.-S. Moon, H. Kim, Performance of Deep Learning in Prediction of Stock Market Volatility, Econ Comput Econ Cybern Stud Res, Vol. 53, No. 2, 2019, pp. 77–92, https://doi.org/10.24818/18423264/53.2.19.05.

[5] A. A. Ariyo, A. O. Adewumi, C. K. Ayo, Stock Price Prediction Using the ARIMA Model, in 2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation, IEEE, Mar. 2014, pp. 106–112, https://doi.org/10.1109/UKSim.2014.67.

[6] H. Hewamalage, C. Bergmeir, K. Bandara, Recurrent Neural Networks for Time Series Forecasting: Status and Future Directions, Int J Forecast, Vol. 37, No. 1, 2021, pp. 388–427, https://doi.org/10.1016/j.ijforecast.2020.06.008.

[7] T. B. Trafalis, H. Ince, Support Vector Machine for Regression and Applications to Financial Forecasting, in Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium, IEEE, Vol. 6, 2000, pp. 348–353, https://doi.org/10.1109/IJCNN.2000.859420.

[8] F. Li, C. Liu, Application Study of BP Neural Network on Stock Market Prediction, in 2009 Ninth International Conference on Hybrid Intelligent Systems, IEEE, 2009, pp. 174–178, https://doi.org/10.1109/HIS.2009.248.

[9] S. Wang, X. Wang, S. Wang, D. Wang, Bi-Directional Long Short-term Memory Method Based on Attention Mechanism and Rolling Update for Short-term Load Forecasting, International Journal of Electrical Power & Energy Systems, Vol. 109, 2019, pp. 470–479, https://doi.org/10.1016/j.ijepes.2019.02.022.

[10] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, Y. Zhang, Short-Term Residential Load Forecasting Based on LSTM Recurrent Neural Network, IEEE Trans Smart Grid, Vol. 10, No. 1, 2019, pp. 841–851, https://doi.org/10.1109/TSG.2017.2753802.

[11] S. Hochreiter, J. Schmidhuber, "Long Short-Term Memory," Neural Comput, Vol. 9, No. 8, 1997, pp. 1735–1780, https://doi.org/10.1162/neco.1997.9.8.1735.

[12] Ouyang Hong-bing, Huang Kang, Yan Hong-ju, Prediction of Financial Time Series Based on LSTM Neural Network, Chinese Journal of Management Science, Vol. 28, No. 4, 2020, pp. 27–35, https://doi.org/10.16381/j.cnki.issn1003-207x.2020.04.003.

[13] T. Fischer, C. Krauss, Deep Learning with Long Short-term Memory Networks for Financial Market Predictions, Eur J Oper Res, Vol. 270, No. 2, 2018, pp. 654–669, https://doi.org/10.1016/j.ejor.2017.11.054.

[14] J. Eapen, D. Bein, A. Verma, Novel Deep Learning Model with CNN and Bi-Directional LSTM for Improved Stock Market Index Prediction, in 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), IEEE, 2019, pp. 0264–0270, https://doi.org/10.1109/CCWC.2019.8666592.

[15] H. Cheng, X. Ding, W. Zhou, R. Ding, A Hybrid Electricity Price Forecasting Model with Bayesian Optimization for German Energy Exchange, International Journal of Electrical Power & Energy Systems, Vol. 110, 2019, pp. 653–666, https://doi.org/10.1016/j.ijepes.2019.03.056.

[16] R. J. Hyndman, A Brief History of Forecasting Competitions, Int J Forecast, Vol. 36, No. 1, 2020, pp. 7–14, https://doi.org/10.1016/j.ijforecast.2019.03.015.

[17] B. Moritz, T. Zimmermann, Tree-Based Conditional Portfolio Sorts: The Relation between Past and Future Stock Returns, SSRN Electronic Journal, 2016, https://doi.org/10.2139/ssrn.2740751.

[18] F. D. Paiva, R. T. N. Cardoso, G. P. Hanaoka, W. M. Duarte, Decision-Making for Financial Trading: A Fusion Approach of Machine Learning and Portfolio Selection, Expert Syst Appl, Vol. 115, 2019, pp. 635–655, https://doi.org/10.1016/j.eswa.2018.08.003.

[19] S. Emerson, R. Kennedy, L. O'Shea, J. O'Brien, Trends and Applications of Machine Learning in Quantitative Finance, in 8th International Conference on Economics and Finance Research (ICEFR 2019), 2019, https://ssrn.com/abstract=3397005.

[20] S. Wang, A. Mathew, Y. Chen, L. Xi, L. Ma, J. Lee, Empirical Analysis of Support Vector Machine Ensemble Classifiers, Expert Syst Appl, Vol. 36, No. 3, 2009, pp. 6466–6476, https://doi.org/10.1016/j.eswa.2008.07.041.

[21] L. H. Hanh, N. N. Nam, N. T. Linh, N. L. Diep, N. N. Hai, Stock Market Prediction: The Application of Text-Mining in Vietnam, VNU Journal of Economics and Business, Vol. 2, No. 2, 2022, pp. 49-58, https://doi.org/10.25073/2588-1108/vnueab.4715.

[22] Z. Hu, Y. Zhao, M. Khushi, A Survey of Forex and Stock Price Prediction Using Deep Learning, Applied System Innovation, Vol. 4, No. 1, 2021, pp. 9, https://doi.org/10.3390/asi4010009.

[23] J. Sirignano, R. Cont, Universal Features of Price Formation in Financial Markets: Perspectives from Deep Learning, Quant Finance, Vol. 19, No. 9, 2019, pp. 1449–1459, https://doi.org/10.1080/14697688.2019.1622295.

[24] J. B. Heaton, N. G. Polson, J. H. Witte, Deep Learning for Finance: Deep Portfolios, Appl Stoch Models Bus Ind, Vol. 33, No. 1, 2017, pp. 3–12, https://doi.org/10.1002/asmb.2209.

[25] S. Selvin, R. Vinayakumar, E. A. Gopalakrishnan, V. K. Menon, K. P. Soman, Stock Price Prediction Using LSTM, RNN and

CNN-Sliding Window Model, in 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), IEEE, 2017, pp. 1643–1647, https://doi.org/ 10.1109/ICACCI.2017.8126078.

[26] E. Guresen, G. Kayakutlu, T. U. Daim, Using Artificial Neural Network Models in Stock Market Index Prediction, Expert Syst Appl, Vol. 38, No. 8, 2011, pp. 10389–10397, https://doi.org/10.1016/j.eswa.2011.02.068.

[27] S. C. Nayak, B. B. Misra, H. S. Behera, Artificial Chemical Reaction Optimization of Neural Networks for Efficient Prediction of Stock Market Indices," Ain Shams Engineering Journal, Vol. 8, No. 3, 2017, pp. 371–390, https://doi.org/10.1016/j.asej.2015.07.015.

[28] R. Mehmood, F. Alam, N. N. Albogami, I. Katib, A. Albeshri, S. M. Altowaijri, UTiLearn: A Personalised Ubiquitous Teaching and Learning System for Smart Societies, IEEE Access, Vol. 5, 2017, pp. 2615–2635, https://doi.org/10.1109/ ACCESS.2017.2668840.

[29] M. Nikou, G. Mansourfar, J. Bagherzadeh, Stock Price Prediction Using DEEP Learning Algorithm and Its Comparison with Machine Learning Algorithms, Intelligent Systems in Accounting, Finance and Management, Vol. 26, No. 4, 2019, pp. 164–174, https://doi.org/10. 1002/isaf.1459.

[30] N. Naik, B. R. Mohan, Study of Stock Return Predictions Using Recurrent Neural Networks with LSTM. In: J. Macintyre, L. Iliadis, I. Maglogiannis, C. Jayne (eds), Engineering Applications of Neural Networks, EANN 2019, Communications in Computer and Information Science, Vol. 1000, Springer, Cham, https://doi.org/10.1007/978-3-030-20257-6_39.

[31] D. M. Q. Nelson, A. C. M. Pereira, R. A. de Oliveira, Stock Market's Price Movement Prediction with LSTM Neural Networks, in 2017 International Joint Conference on Neural Networks (IJCNN), IEEE, 2017, pp. 1419–1426, https://doi.org/10.1109/IJCNN.2017.7966019.

[32] C. Xiao, W. Xia, J. Jiang, Stock Price Forecast Based on Combined Model of ARI-MA-LS-SVM, Neural Comput Appl, Vol. 32, No. 10, 2020, pp. 5379–5388, https://doi.org/10.1007/ s00521-019-04698-5.

[33] B. Gülmez, Stock Price Prediction with Optimized Deep LSTM Network with Artificial Rabbits Optimization Algorithm, Expert Syst Appl, Vol. 227, 2023, pp. 120346, https://doi.org/10.1016/j.eswa.2023.120346.

[34] S. Usmani, J. A. Shamsi, LSTM Based Stock Prediction Using Weighted and Categorized Financial News," PLoS One, Vol. 18, No. 3, 2023, pp. e0282234, https://doi.org/10.1371/ journal.pone.0282234.

[35] T. Phuoc, P. T. K. Anh, P. H. Tam, C. V. Nguyen, Applying Machine Learning Algorithms to Predict the Stock Price Trend in the Stock Market – The Case of Vietnam, Humanit Soc Sci Commun, Vol. 11, No. 1, 2024, pp. 393, https://doi.org/10.1057/s41599-024-02807-x.

[36] J. Zhang, L. Ye, Y. Lai, Stock Price Prediction Using CNN-BiLSTM-Attention Model, Mathematics, Vol. 11, No. 9, 2023, pp. 1985, https://doi.org/10.3390/math11091985.

[37] M. Yang, J. Wang, Adaptability of Financial Time Series Prediction Based on BiLSTM, Procedia Comput Sci, Vol. 199, 2022, pp. 18–25, https://doi.org/10.1016/j.procs.2022.01.003.

[38] J. M.-T. Wu, Z. Li, N. Herencsar, B. Vo, J. C.-W. Lin, A Graph-Based CNN-LSTM Stock Price Prediction Algorithm with Leading Indicators, Multimed Syst, Vol. 29, No. 3, 2023, pp. 1751–1770, https://doi.org/10.1007/s00530-021-00758-w.

[39] Md. A. Istiake Sunny, M. M. S. Maswood, A. G. Alharbi, Deep Learning-Based Stock Price Prediction Using LSTM and Bi-Directional LSTM Model, in 2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES), IEEE, 2020, pp. 87–92, https://doi.org/ 10.1109/NILES50944.2020.9257950.

[40] C. Han, X. Fu, Challenge and Opportunity: Deep Learning-Based Stock Price Prediction by Using Bi-Directional LSTM Model, Frontiers in Business, Economics and Management, Vol. 8, No. 2, 2023, pp. 51–54, https://doi.org/10. 54097/fbem.v8i2.6616.

[41] S. Liu, Q. Huang, M. Li, Y. Wei, A New LASSO-BiLSTM-Based Ensemble Learning Approach for Exchange Rate Forecasting," Eng Appl Artif Intell, Vol. 127, 2024, pp. 107305, https://doi.org/10.1016/j.engappai.2023.107305.

[42] Y. Zhu, Stock Price Prediction Using the RNN Model, J Phys Conf Ser, Vol. 1650, 2020, pp. 032103, https://doi.org/10.1088/1742-6596/ 1650/3/032103.

[43] V.-D. Ta, C.-M. Liu, D. A. Tadesse, Portfolio Optimization-Based Stock Prediction Using

Long-Short Term Memory Network in Quantitative Trading, Applied Sciences, Vol. 10, No. 2, 2020, pp. 437, https://doi.org/10.3390/app10020437.

[44] S. Borovkova, I. Tsiamas, An Ensemble of LSTM Neural Networks for High-Frequency Stock Market Classification," J Forecast, Vol. 38, No. 6, 2019, pp. 600–619, https://doi.org/10.1002/for.2585.

[45] I. E. Livieris, E. Pintelas, P. Pintelas, A CNN–LSTM Model for Gold Price Time-Series Forecasting, Neural Comput Appl, Vol. 32, No. 23, 2020, pp. 17351–17360, https://doi.org/10.1007/s00521-020-04867-x.

[46] X. Yan, W. Weihan, M. Chang, Research on Financial Assets Transaction Prediction Model Based on LSTM Neural Network, Neural Comput Appl, Vol. 33, No. 1, 2021, pp. 257–270, https://doi.org/10.1007/s00521-020-04992-7.

[47] D. P. Kingma, J. Ba, Adam: A Method for Stochastic Optimization, in Proceedings of the 3rd International Conference on Learning Representations (ICLR), Banff, 2014, https://arxiv.org/abs/1412.6980

[48] J. Huang, J. Chai, S. Cho, Deep Learning in Finance and Banking: A Literature Review and Classification, Frontiers of Business Research in China, Vol. 14, No. 1, 2020, pp. 13, https://doi.org/10.1186/s11782-020-00082-6