VIETNAM NATIONAL UNIVERSITY, HANOI

INTERNATIONAL SCHOOL

———

— Assignment Report —

VIETNAM STOCK PRICE PREDICTION USING MACHINE LEARNING TECHNIQUES

Student Name:

Công Châu Anh (21070081)

Trần Thảo Chi (21070065)

Hanoi, August 2022.

ABSTRACT

There have been many approaches to solving the problem of predicting stock prices, from applying traditional algorithms like Support Vector Machine, Decision Tree, and Random Forest to deep learning models such as Convolutional Neural Network and Recurrent Neural Network. One advanced model that shows high suitability and effectiveness for problems regarding time-series data is the LSTM (Long-Short Term Memory) network.

In this study, we propose the CNN-BiLSTM deep learning model to predict the future values of stocks in the steel securities group. The model includes three major components: 1D CNN layers, Bidirectional Long-Short Term Memory layers, and Fully Connected layers. We were experimenting with HPG, HSG, VIS and DTL (as representatives of the steel sector in the stock market) and used them to try to predict daily stock price. The test results indicate that our CNN-BiLSTM model of proposals can get a good result in predicting stock prices compared to many other test models and score decently in evaluation metrics like accuracy, mean absolute error (MAE), and mean squared error (MSE).

INTRODUCTION

The stock market has always maintained an important position in the economy and has attracted a wide range of investors in recent years. However, the stock market possesses a random and nonlinear property that makes forecasting the market trend and future price of different stocks very challenging, especially when considering factors like market demand or social and political situations.

With the boom of artificial intelligence recently, deep learning techniques are being widely used in the financial sector for problems such as credit risk assessment, market analysis, customer segmentation, fraud detection, etc. Specifically, to minimize losses and maximize profits when investing in a portfolio, models for predicting stock prices have gained much interest and have become the most common research problem in this area.

OVERVIEW OF THE STOCK PRICE PREDICTION PROBLEM

1. The Stock Market

a. Definition of Securities

Securities are assets, including: Stocks, bonds, fund certificates; warranties, covered warrants, rights to buy shares, depository certificates, and other types of securities regulated by the government.

b. Definition of Stocks and Shares

A stock is a general term used to describe the ownership certificates of any company. A share, on the other hand, refers to the stock certificate that confirms the owner's lawful rights and interests in a portion of the share capital of the issuing organization. In other words, shares are certificates of money that shareholders invest in a business.

Joint stock companies issue shares to raise capital for their businesses, and stocks are bought and sold mainly on stock exchanges. In Vietnam's stock market, one base share represents 10,000 VND of charter capital of the enterprise.

2. The Stock Price Prediction Problem

In recent years, with the significant advances in science and technology, researchers have made great efforts to find solutions, evaluation, and optimization methods to improve the accuracy of machine learning models. Because of that, difficult problems that seemed impossible to solve have been solved more and more, promoting the development of the economy when technological barriers are slowly removed.

One of the problems most scientists and economists are interested in is the problem of forecasting stock prices. From the earliest days when the first stock market was established in the 1600s, economists saw the potential that stocks could bring. One by one, pioneers of time series prediction methods appeared, such as the AutoRegressive model invented in the 1920s by statistician Udny Yule and colleagues. This model was the basis for some later statistical and econometric models such as ARMA and ARIMA. In recent years, machine learning models have been applied to this problem to help investors generate profits, but with traditional machine learning models, the accuracy is still limited. However, with the development of deep learning models, identifying non-linear patterns in a stock's time series has become more feasible than ever.

The first to be mentioned is RNN - a neural network specially designed with the ability to process temporal data and extract information and predictions based on previous information series. Therefore, the RNN model and its variants seem to be suitable for the prediction of time series, and financial time series in particular.

CNN is one of the other deep learning models that can be applied to the stock price prediction problem thanks to its ability to efficiently extract information through filters. According to some experimental results, CNN has a significant role in input data processing and feature extraction. For example, CNN can be applied to a variety of data from different sources, covering multiple markets, and extract features to predict for these markets. In reality, when compared with the baseline algorithm, the performance for prediction when using CNN is significantly improved.

In addition, researchers have also applied the Reinforcement Learning model - a deep learning method that takes action based on the current situation to maximize profits. Reinforcement learning learns how to execute trades (be it to buy, sell, or hold stock) according to the current market situation. The model views contextual information (price, news, public opinion, transaction fees, trade action, profit, loss) as an environment for reinforcement learning, profit or losses as rewards, and transactions (sell, buy, hold) as actions that need to be optimized to gain optimal profits.

A fairly popular and effective approach in recent years for the time series prediction problem is the LSTM model, which is a variant of RNN that has attracted much attention from researchers domestically and internationally. LSTM is used a lot for problems with temporal or sequential data, such as machine translation, speech recognition, weather forecasting, and, of course, stock price prediction with high accuracy.

THEORETICAL BACKGROUND

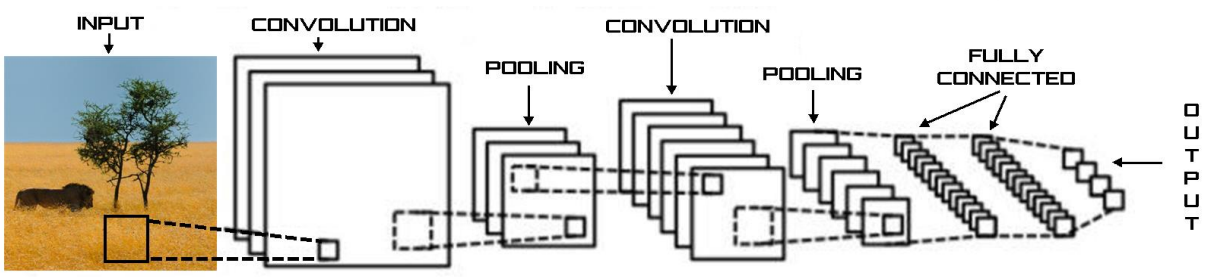
1. Convolutional Neural Network – CNN

The simple neural network model was born and has been applied to recognition problems. However, for image data, the feedforward neural network does not give very good results because the association is too complete, creating limitations for the model. If we consider a pixel as a feature, an RGB color image of size (64 x 64) will have 64 x 64 x 3 = 12288 features. If we increase the image size to 1000 x 1000, we will get a feature vector with 1000 x 1000 x 3 = 3000000 dimensions. Obviously, if we use more fully connected layers, the number of weights will increase tremendously. For example, with the next layer using 100 units, the weight matrix will have a size of 3000000 x 100, which is equivalent to 3,108 weights to be trained; if more fully connected layers are added, the matrix size will become larger and larger. This requires a huge amount of computation and often leads to overfitting due to insufficient training data. The full association of pixels in the network is redundant because the interdependence between distant pixels is small, but it is mainly the dependence between its neighbors that is important.

Based on this idea, the deep learning model CNN was born with a structure different from that of a normal neural network. First, the layers of the CNN are organized in 3 dimensions: width, height, and depth. And instead of having each layer fully connected to all the neurons in the previous layer like in a feed-forward network, the neurons in one layer of the CNN are connected only to a small part of the neurons in the next layers of it.

a. The basic architecture of CNN

The CNN model includes the following layers: the convolutional layer, the activation layer, the pooling layer, and finally the fully connected layer. Through the convolution layer, the features of the data are extracted and passed through the activation layer to produce a feature map. Then they continue to go through the pooling layer to reduce the number of data dimensions but still retain the necessary amount of information, which helps the model learn faster by reducing the computational cost. Finally, after having obtained the feature map, we flatten and pass the output to the fully connected layer to obtain the prediction result. In short, such a CNN model of feature extraction aims to reduce the number of features present in a dataset. It creates new features which summarize the existing features contained in an original set of features. There are many CNN layers, as shown in the CNN architecture diagram in image processing below.



b. Convolutional Layer

The convolution layer is the core layer in CNN. The function of the convolution layer is to extract

features of the input data through filters. A filter (or kernel) is a set of parameters learned during model training to extract features for data. In the field of image processing, convolution is widely used and very important. For example: for image filtering, the convolution between the filter matrix and the image, results in a noise-removed image.

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Input

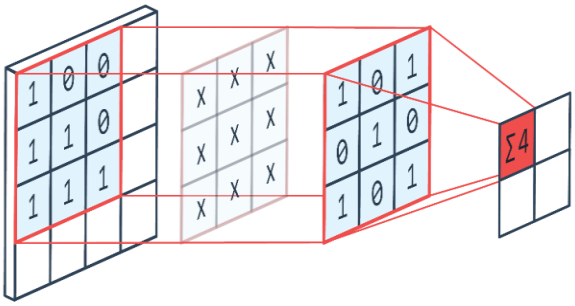
1	0	1
0	1	0
1	0	1

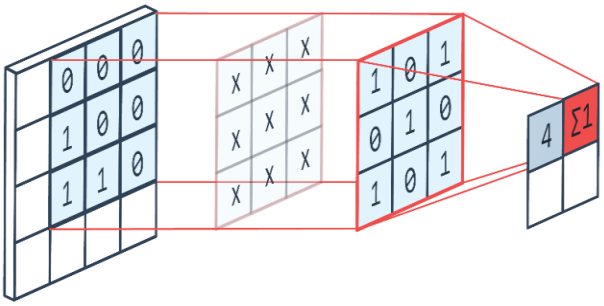
Filter / Kernel

The convolution formula between the image function $f(x, y)$ and the kernel $k(x, y)$ (of size $m \times n$) is:

$$f(x,y) * k(x,y) = \sum_{u=\frac{-m}{2}}^{\frac{m}{2}} \sum_{v=\frac{-n}{2}}^{\frac{n}{2}} k(u,v) f(x-u, y-v)$$

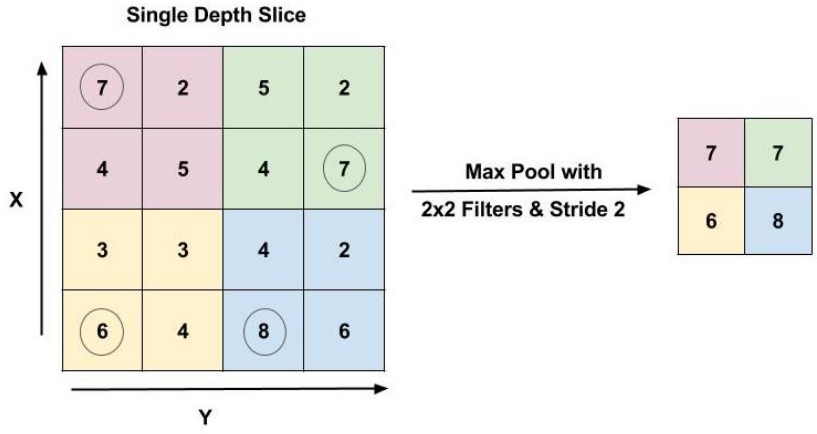
In the convolution calculation, we start with the convolution window in the upper left corner of the input data array, and move the convolution window from left to right, and top to bottom. For each position where the filter lies entirely in the input array, we perform the convolution by multiplying each element of the convolution window and the subarray at the corresponding position, then adding all the values together to obtain a single value. Each filter will produce different features, so each convolution layer will use many filters to find many features of the data. Assuming that we have all filters in the convolutional layer (e.g. k filters), each filter results in a distinctive two-dimensional matrix (e.g. matrix of size $n \times n$) and we can layer these matrices to create a new matrix of depth k as the output matrix (size $n \times n \times k$).





c. Pooling Layer

In most cases, a convolutional layer is followed by a pooling layer. The primary aim of this layer is to reduce the data size and the number of parameters in the network. Thereby also limiting overfitting. It should be noted that the pooling class only reduces the width and height dimensions, not the depth dimensions of the input data. There are



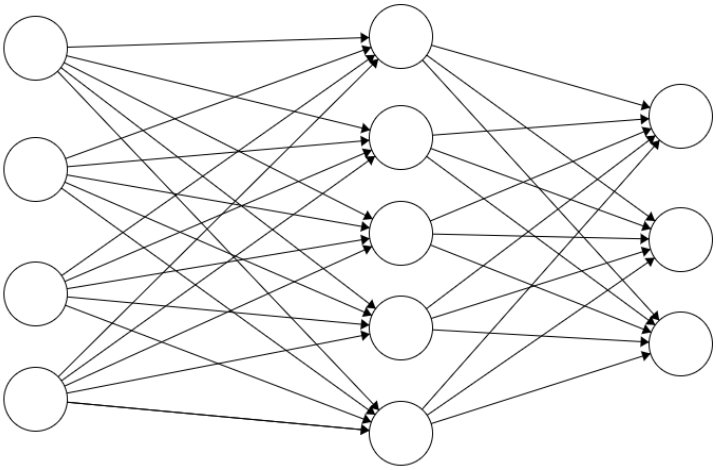
many different types of pooling class: max pooling - taking the largest element in the submatrix, average pooling - taking the average value of the elements, sum pooling - taking the total value of the elements. Of all the types, max pooling is the most widely used.

The pooling layer usually serves as a bridge between the convolutional layer and the fully connected layer. This CNN model generalizes the features extracted by the convolution layer and helps the networks recognize the features independently. With the help of this, the computations are also reduced in a network.

d. Fully Connected Layer

The fully connected (FC) layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the output layer and form the last few layers of a CNN architecture.

In this, the input data from the previous layers is flattened and fed to the FC layer. The flattened vector then undergoes a few more FC layers, where mathematical function operations usually take place. At this stage, the classification process begins to take place. The reason two layers are connected is that two fully connected layers will perform better than a single connected layer. These layers in CNN reduce the human supervision.



e. Dropout

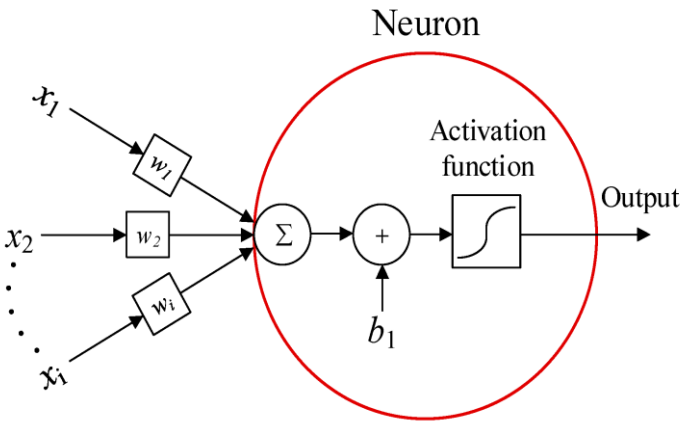
Usually, when all the features are connected to the FC layer, it can cause overfitting in the training dataset. Overfitting occurs when a particular model works so well on the training data that it has a negative impact on the model's performance when used on new data.

To overcome this problem, a dropout layer is utilized wherein a few neurons are dropped from the neural network during the training process, resulting in a reduced size of the model. On passing a dropout of 0.3, 30% of the nodes are dropped out randomly from the neural network.

Dropout results in improving the performance of a machine learning model as it prevents overfitting by making the network simpler. It drops neurons from the neural networks during training.

f. Activation Functions

Finally, one of the most important parameters of the CNN model is the activation function. They are used to learn and approximate any kind of continuous and complex relationship between variables in the network. In simple words, it decides which pieces of information in the model should fire in the forward direction and which ones should not at the end of the network.

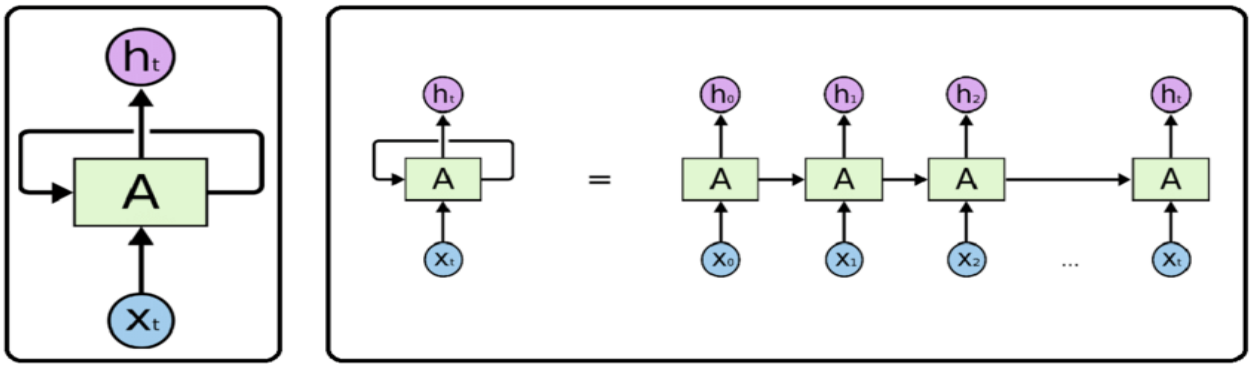


It adds non-linearity to the network. There are several commonly used activation functions, such as the ReLU, Softmax, tanH, and the Sigmoid functions. Each of these functions has a specific usage. For a binary classification CNN model, sigmoid and softmax functions are preferred. For a multi-class classification, softmax is generally used. In simple terms, activation functions in a CNN model determine whether a neuron should be activated or not. It decides whether the input to the work is important or not to predict using mathematical operations.

2. Recurrent Neural Network – RNN

The basic architecture of RNN

In recent years, as technology constantly develops and advances, tasks for processing time series data have also achieved many outstanding achievements. In particular, the neural network model was born and has been applied to many different problems. However, for temporal or sequential data, the feedforward neural model does not meet the requirements of the problem, thus leading to the birth of the Recurrent Neural Network (RNN) - a neural network model that can handle these types of sequential data. In essence, it is similar to an ordinary artificial neural network, but the current output data at any point in time is retained by back-propagation, so that the information of the same data point is kept from time to time.



In the above model, the reverse arrow represents the loop. On the left is a reduced image of the network (representing a loop) and on the right is the network after being unpacked. As mentioned above, the network

will receive a series of signal vectors $x_1, x_2, \ldots, x_t$ corresponding to the input signal of the data point at the current time $t = 1 -> t$. Furthermore, the network will also produce hidden results h_t, equivalent to the output signal vector at that time t. So, the regression network shown above will take the input signal x_t along with the hidden output h_{t-1} from the previous calculation. These two different signals will be combined according to their respective parameter matrices to calculate h_t until the model reaches the final value of t.

The basic formula of the regression network to calculate the hidden signal vectors h_t is:

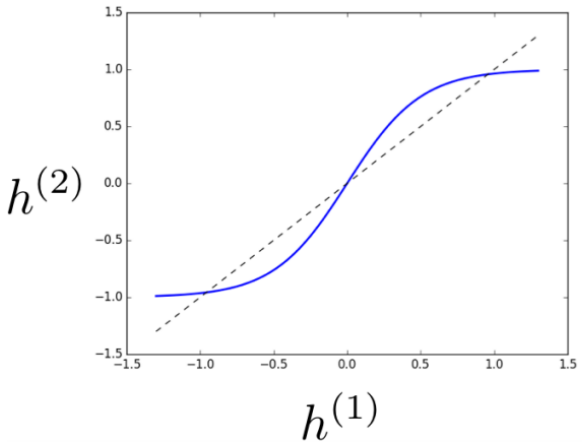
$$h_t = \varphi\,(WX_t + Uh_{t-1})$$

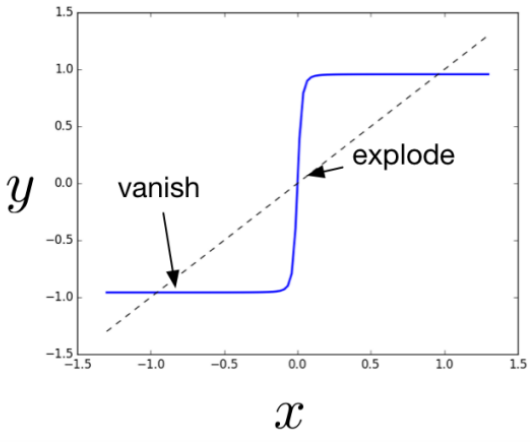
with W being equivalent to the weight matrix like in basic artificial neural networks and U representing the weights of historical result signals. Finally, the function φ will be used to "squeeze" each value of the vector h_t into a logarithmic value (such as from -1 $->$ 1, so that the derivative can be calculated).

a. The drawbacks of RNN

The distinctive feature of RNN compared to other neural networks is the concept of connecting previous information to conduct the next prediction. But if the data is too long or the amount of information is too large, the RNN will not be able to "remember" the information from too long ago, resulting in a phenomenon called “vanishing gradient”.

This phenomenon occurs since the activation function (tanh or sigmoid) of the model will result in the output in the range [−1, 1] (for sigmoid it is [0, 1]), which means that the derivative value of it will also be in the range [0, 1] (for sigmoid it is [0, 0.25]). Above, we used the series rule to calculate the derivative, but when the derivative is 0, the corresponding network node will be saturated. When this happens, the front nodes will also be saturated. So, when we perform matrix multiplication for small values in the matrix, vanishing gradient will occur on the corresponding derivative, meaning that the derivative will vanish after only a few multiplications. Thus, it is impossible for the RNN to learn far-distance dependencies.





In addition to the above problem, we also see another phenomenon called “exploding gradient”. Depending on the activation function and the parameters of the network, this problem occurs when the values of the matrix are large (greater than 1) and leads to extremely large weights during training. However, vanishing gradients are mentioned more often for two reasons:

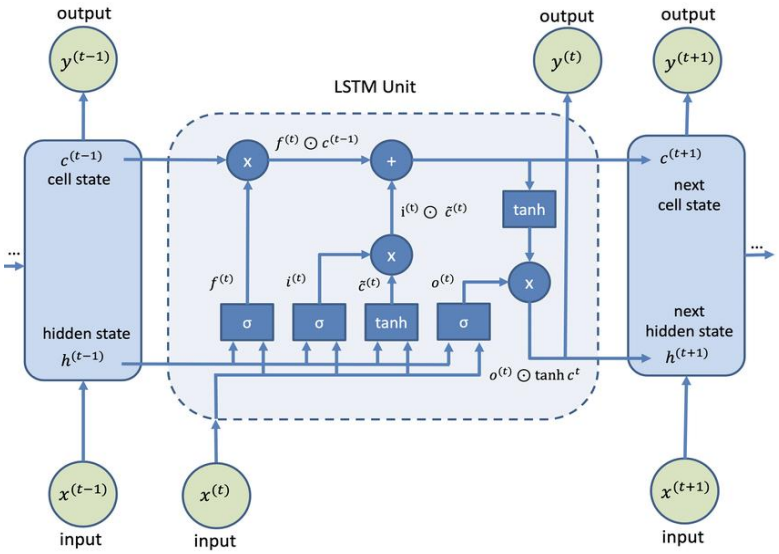
- Exploding gradients are traceable because when the derivative explodes, we get a non-numeric value of NaN that causes our program to crash.

• Exploding gradient can be prevented when we set a threshold value using the Gradient Clipping technique, but we cannot do that for vanishing gradient.

Nevertheless, there have been methods to tackle the vanishing gradient phenomenon. One of the popular ways is to use the activation function ReLU (or its variants) because the derivative of ReLu is either 0 or 1, so we can partially control the problem of the loss of derivatives. Because RNN is unable to filter unnecessary information, other variants of RNN were born with architectures that give the network more long-term memory. Among these, the most popular networks are LSTM (Long - Short Term Memory) and GRU (Gated Recurrent Unit).

3. Long - Short Term Memory – LSTM

Long Short Memory Network (LSTM) is a variant of the RNN that is capable of learning distant dependencies. It has been designed so that the vanishing gradient problem is almost completely removed, while the training model is left unaltered. Long time lags in certain problems are also bridged using LSTMs where they also handle noise, distributed representations, and continuous values. With LSTMs, there is no need to keep a finite number of states beforehand, as required in the hidden Markov model (HMM). Moreover, LSTMs provide us with a large range of parameters such as learning rates, and input and output biases. Hence, no need for fine adjustments.



The basic difference between the architectures of RNNs and LSTMs is that the hidden layer of an LSTM is a gated unit or gated cell. It consists of four layers that interact with one another in a way to produce the output of that cell along with the cell state. These two things are then passed on to the next hidden layer. Unlike RNNs, which have only a single neural net layer of tanh, LSTM comprises three logistic sigmoid gates and one tanh layer.

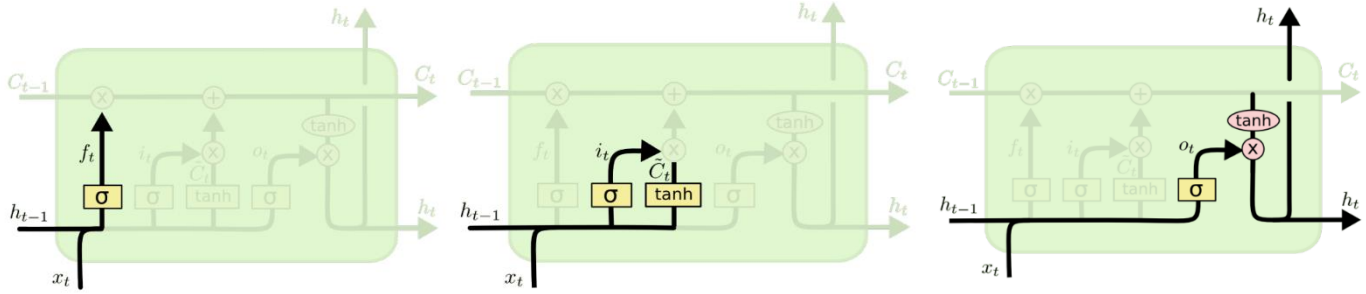
The main concept of the LSTM network is that: for each time t, we will have a corresponding state of the LSTM cell (1 cell). In the figure, we can see a straight line running across from C_{t-1} to C_t, corresponding to the fact that we will transfer the result from the previous state to the next state. Interacting with the C_{t-1} values will be different gates and matrix operations $(\times, +)$. Gates are introduced in order to limit the amount of information that is passed through the cell. They determine which part of the information will be needed by the next cell and which part is to be discarded.

a. Forget gate

The forget gate layer is used to decide how much of the old C_{t-1} information is kept. This gate will use the previously hidden result value (h_{t-1}) and the current input signal x_t as input. When pooled together,

that result is "reactivated" via the gate's sigmoid function. This new value will range from 0 −> 1, representing “forget all” and “remember all”:

$$f_t = \sigma(W_{fXt} + U_f h_{t-1} + b_f)$$



b. Update gate

The update gate layer is used to decide how much new information will be entered in $\widetilde{C_t}$. That new information is generated based on the previously hidden result (h_{t-1}) and the current input signal x_t, and is activated via the $tanh$ function, generating values from -1 −> 1. The actual value of the update gate will also be activated, similarly to the forget gate, through the sigmoid function, showing the level from "forget all" to "remember all" of the new information:

$$f_t = \sigma(Wx_f x_t + Wh_f h_{t-1} + Wc_f c_{t-1} + b_f)$$
$$i_t = \sigma(Wx_i x_t + Uh_i h_{t-1} + b_i)$$
$$\widetilde{C_t} = tanh(W_C x_t + U_C h_{t-1} + b_c)$$

Here we need to combine the forget gate and the update gate to produce the actual C_t state:

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C_t}$$

As mentioned, the forget gate decides whether the old information is kept or not, and the update gate determines whether the new information is used or not.

c. Output gate

The output gate further filters the input value to figure out what information of the cell state should be output. Like in the previous two gates, we use the sigmoid function to “squeeze” the combined value of h_{t-1} and x_t to make this decision. The value of the current state C_t is squeezed into the range from -1 −> 1 by using $tanh$ before multiplying with the output gate to obtain the desired output value:

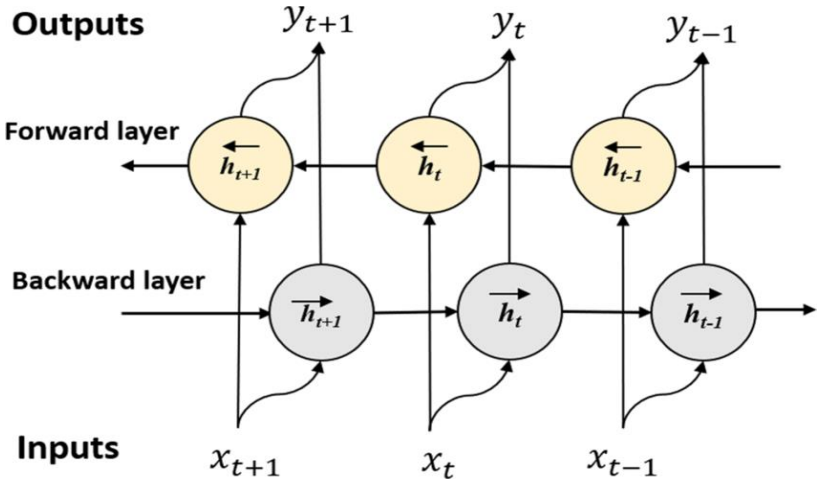
$$o_t = \sigma(W_o x_t + U_{oi} h_{t-1} + b_o)$$
$$h_t = o_t * tanh(C_t)$$

4. Bidirectional Long - Short Term Memory – BiLSTM

The Bidirectional Recurrent Neural Network (BiLSTM) model is a connection of two LSTMs in opposite directions. This structure allows the network to have both backward (from the past) and forward (from the future) information about the sequence at every time step. Invented in 1997 by Schuster and Paliwal,

BiLSTM was introduced to increase the amount of input information available to the network. For example, multi-layer perceptrons (MLPs) and time-delayed neural networks (TDNNs) have limitations in terms of input flexibility as they require their input to be corrected. Standard recurrent neural networks (RNNs) also have limitations because they cannot access future input from the current state. In contrast, Bi-LSTMs do not require their input data to be fixed. Furthermore, their future inputs are accessible from the current state.

Given a time step t, the minibatch input $X_t \in R^{n\times d}$ (n is the number of samples, d is the number of input) and the activation function in hidden layers is ϕ. In the two-way architecture, we assume that the forward and backward latent states of this time step are respectively $\overrightarrow{H_t} \in R^{n\times h}$ and $\overleftarrow{H_t} \in R^{n\times h}$, with h being the number of hidden nodes. We compute the forward and backward updating of the hidden state as follows:



$$\overrightarrow{H_t} = \phi(X_t W_{xh}^{(f)} + \overrightarrow{H_{t-1}} W_{hh}^{(f)} + b_h^{(f)})$$

$$\overleftarrow{H_t} = \phi(X_t W_{xh}^{(b)} + \overrightarrow{H_{t+1}} W_{hh}^{(b)} + b_h^{(b)})$$

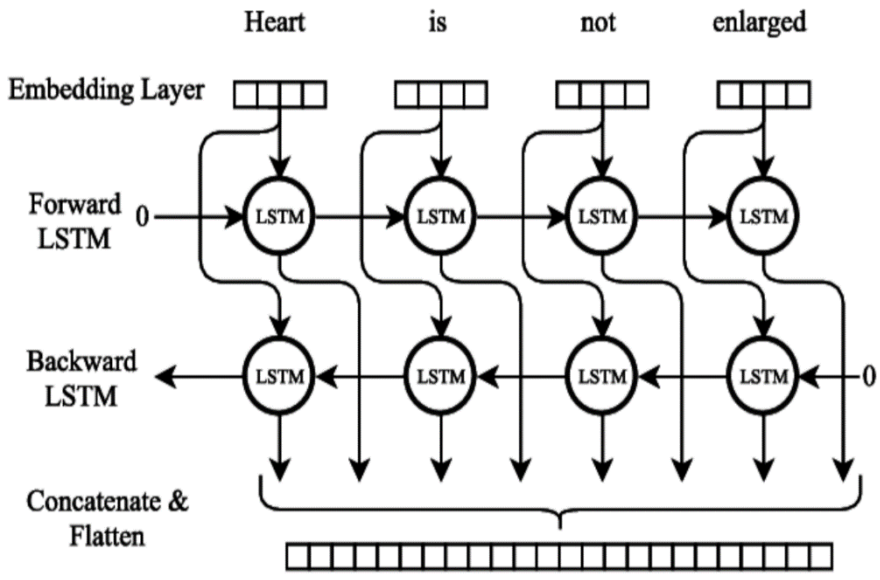
Here, the weights $W_{xh}^{(f)} \in R^{d\times h}, W_{hh}^{(f)} \in R^{h\times h}, W_{xh}^{(b)} \in R^{d\times h}, W_{hh}^{(b)} \in R^{h\times h}$ and bias $b_h^{(f)} \in R^{1\times h}, b_h^{(b)} \in R^{1\times h}$ are all parameters of the model.

Then we connect the forward and backward hidden states $(\overrightarrow{H_t}, \overleftarrow{H_t})$ to obtain the hidden state $H_t \in R^{n\times 2h}$ and transmit it to the output layer. In bidirectional recurrent neural networks, information is transmitted as input to the next bidirectional layer. Finally, the output layer calculates the output $H_t \in R^{n\times 2h}$ as follows (q is the number of output):

$$O_t = H_t W_{hq} + b_q$$

The weight $W_{hq} \in R^{2h\times q}$ and bias $b_q \in R^{1\times q}$ are model parameters of the output layer. The forward and backward directions can have different number of hidden nodes.

In essence, the principle of BiLSTM is to divide the neurons of an ordinary RNN into two directions, one for positive time direction (forward state) and the other for negative time direction (backward



state). The outputs of those two states are not connected to the inputs of the states in the opposite direction. By using two-time directions, input information from the past and future of the current time frame can be used unlike a standard RNN, which requires a delay to include future information.

BiLSTM can be trained using similar algorithms to RNNs, since the two directional neurons do not have any interaction. However, when back-propagation is applied, additional procedures are needed since it is not possible to update the input and output layers at the same time. The general procedure for training is as follows: for forwarding, the forward and backward states are switched first, and then the output neurons are forwarded; for backpropagation, the output neurons are forwarded first, then the forward and backward states are forwarded. After performing the forward and backward passes, the weights are updated.

PROPOSED MODEL

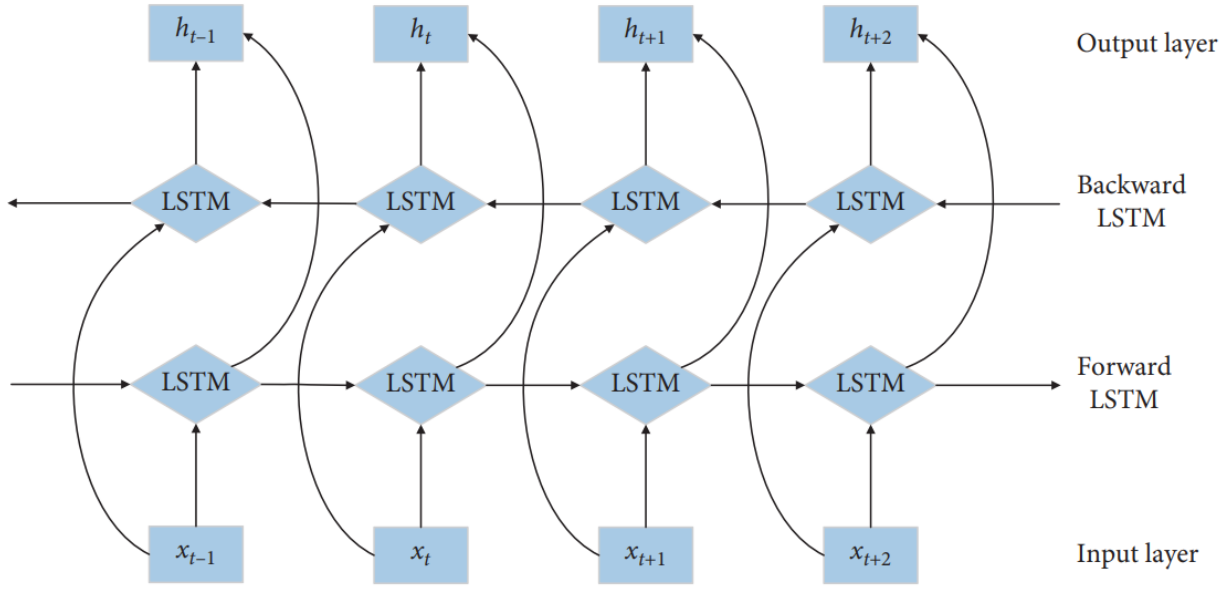
In order to extract features efficiently and improve prediction accuracy, we combine CNN and BiLSTM network into a unified framework and propose a new time series prediction network model named as CNN-BiLSTM. Our proposed model can automatically learn and extract local features and long memory features in the time series by making full use of data information to minimize the complexity of the model.

1. The structure of the hybrid CNN-BiLSTM model

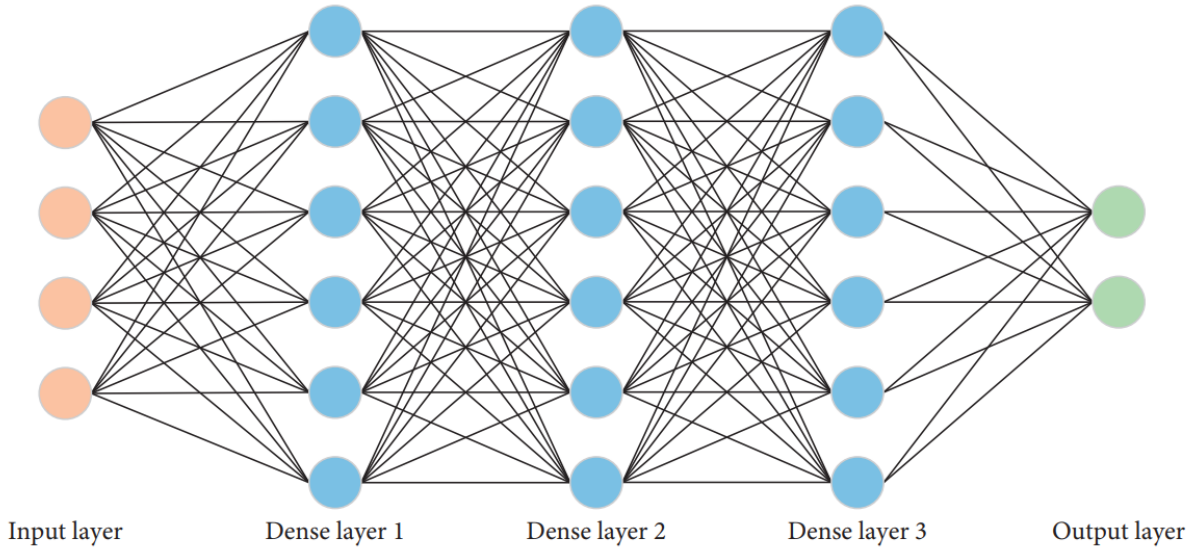
In this model, firstly, the CNN model is utilized to extract deep feature vectors from the input origin time series data. Then, the BiLSTM model is employed to learn the temporal features from the new time series data constructed by the deep feature vectors. Finally, the Dense model consisting of several fully connected layers is employed to perform the prediction task.

• *Convolutional 1D layer:* Performing multiple time-scale learning through different numbers of layers of CNN, combined with daily base prices to reflect stock price changes in the short, medium, and long term.

• *BiLSTM layer:* In order to build a more accurate prediction model, the bidirectional LSTM (BiLSTM) network is employed, which acts as forward and backward LSTM networks for each training sequence. The two LSTM networks are connected to the same output layer to provide complete context information to each sequence point.



• *Dense Model layer:* . The purpose of Dense model is first to exploit full connection layer to extract the correlation among these features changed by nonlinear mapping and then maps them to the output space. In the proposed network structure model, three layers of full connection layer are added to solve the nonlinear problem well, which can achieve accurate prediction.



This model has 2 main subparts: CNN and LSTM. For CNN, the layers are created with sizes 64,128,64 with kernel size = 3. In every layer, a Time Distributed function is added to track the features for every temporal slice of data with respect to time. In between, Max Pooling layers are added. After that, it is passed to Bi-LSTM layers.

2. Methods of evaluation

In order to analyze the performance of the proposed prediction model intuitively and quantitatively, three evaluation criteria including Mean Square Error (MSE), Root Mean Square Error (RMSE), and Mean Absolute Error (MAE) are utilized to evaluate the prediction results comprehensively.

a. MSE

The Mean Square Error (MSE) represents the mean of the sum squares of the errors between the predicted data and the original data, which is defined as follows:

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

b. RMSE

The Root Mean Square Error (RMSE) defines the expected value of the square of the error caused by the predicted value and the true value, and its range is [0, +∞). When the predicted value is completely consistent with the true value, the value of RMSE is equal to 0, which is a perfect model. On the contrary, when the prediction error is larger, the value of the RMSE will be larger. The calculation formula is as follows:

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

c. MAE

The mean absolute error (MAE) represents the average of the absolute value of the deviation of all single observations, which can avoid the problem of mutual cancellation of errors and accurately reflect the size of the actual forecast error as below:

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

d. MAPE

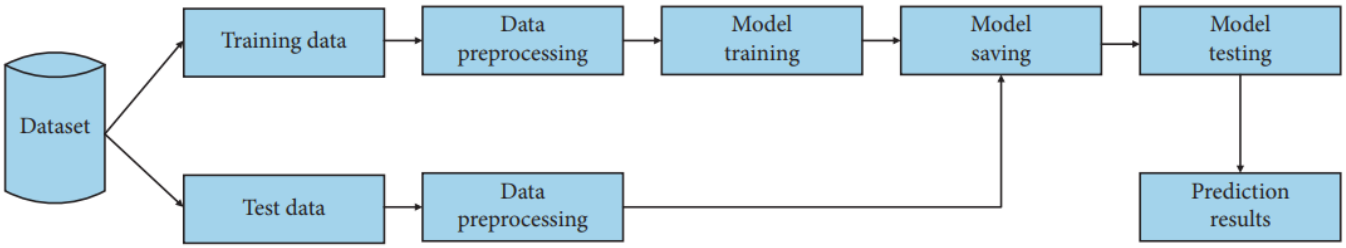
Mean Absolute Percentage Error (MAPE) is one of the most commonly used validation methods for regression problems. MAPE measures model performance by calculating absolute error percentage, with the formula defined as below:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| * 100$$

EXPERIMENTING

1. Experimental Process

The experiment consists of six parts: data colleting, data preprocessing, model training, model saving, model testing, and prediction results.



2. Data Description and Preprocessing

Data is a core indispensable component for any machine learning problem. Usually, in experiments on machine learning, the data is divided into 3 main components for testing and evaluation, including the *training* set, the *validation* set and the *test* set. The dataset used consists of data from 4 giant steel JSCs in Vietnam: Đại Thiên Lộc Corporation, Vietnam – Italy Steel, Hòa Phát Group and Hoa Sen Group.

The dataset includes historical data of the corresponding stock tickers - DTL, VIS, HPG and HSG - from 2010 till now, including fields of: Opening price (Open), closing price (Close), high price (High), low price (Low) and Volume from Phạm Đình Khánh's *vnquant* Python library. The dataset includes:

Table 5.1. Vietnam steel JSCs' stock dataset (2010-now)

Number of stock tickers	Total records
4	9362

Each stock data includes the closing price, the highest price, the lowest price, the opening price and closing price. From these tables, the original dataset obtained from the network cannot be directly used for model training and testing, so the data preprocessing is required.

	date	open	high	low	close	volume	symbol
0	1/4/2010	60.00	61.00	60.00	61.00	653120	HPG
1	1/5/2010	64.00	64.00	61.00	64.00	1203080	HPG
2	1/6/2010	63.50	66.00	63.00	66.00	1771660	HPG
3	1/7/2010	67.00	69.00	66.00	68.00	1535800	HPG
4	1/8/2010	69.50	69.50	65.00	65.00	909010	HPG
...	...	...	...	...	...	...	...
9357	8/22/2022	26.50	26.50	26.20	26.35	1200	DTL
9358	8/23/2022	26.35	26.35	26.35	26.35	400	DTL
9359	8/24/2022	26.35	26.50	26.35	26.50	700	DTL
9360	8/25/2022	26.50	27.00	26.50	27.00	500	DTL
9361	8/26/2022	27.00	27.00	27.00	27.00	500	DTL

9362 rows × 7 columns

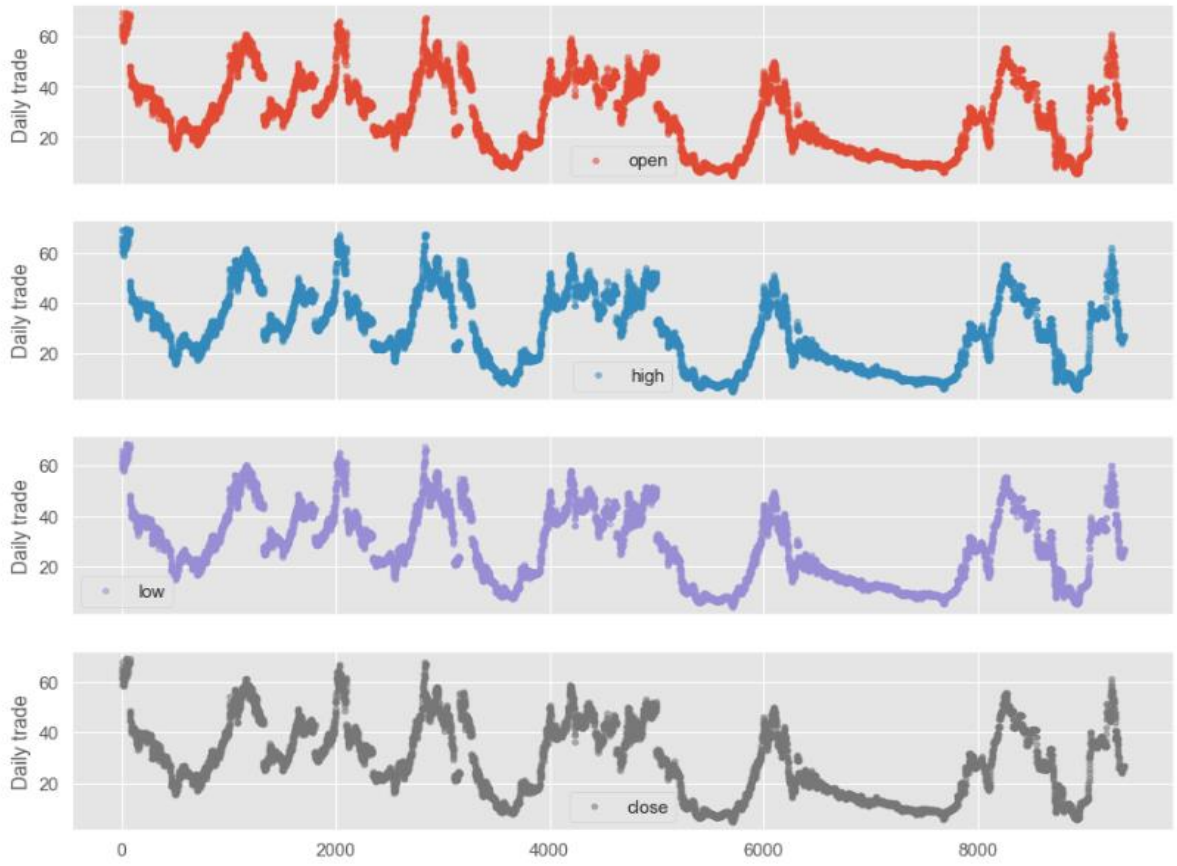
In the missing of null or disordered attribute values and inconsistency in the original dataset, there's no need to employ interpolation and data normalization to deal with. Finally, an effective dataset can be constructed for the experiment. Data normalization is to scale the data according to a certain scale and transform it into a specific interval. In this experiment, the value of data is converted to [0, 1], which improves the convergence speed and accuracy of the model.

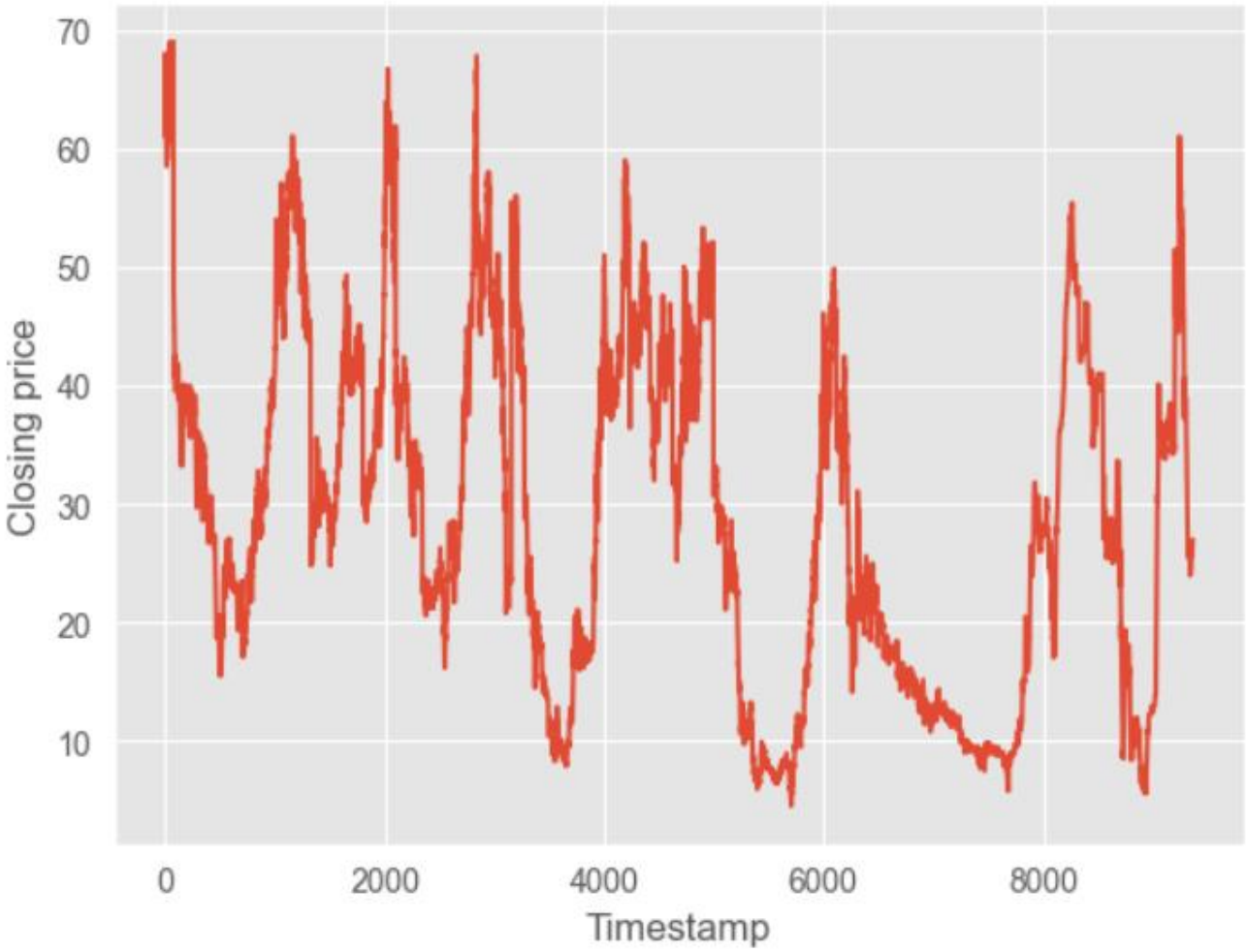
- ***Fetch & load dataset:***

Before data preprocessing, the above-mentioned *vnquant* library is employed to fetch stock data from cafef.vn/. Descriptive statistics:

	open	**high**	**low**	**close**	**volume**
count	9362.000000	9362.000000	9362.000000	9362.000000	9.362000e+03
mean	28.848578	29.205023	28.500911	28.842038	3.158201e+06
std	14.587222	14.753716	14.423069	14.588116	6.510987e+06
min	4.500000	4.800000	4.360000	4.550000	0.000000e+00
25%	15.900000	16.000000	15.600000	15.850000	1.000000e+04
50%	28.100000	28.450000	27.900000	28.125000	3.938750e+05
75%	40.500000	41.000000	40.100000	40.600000	3.119210e+06
max	69.500000	70.000000	68.500000	69.000000	7.784400e+07

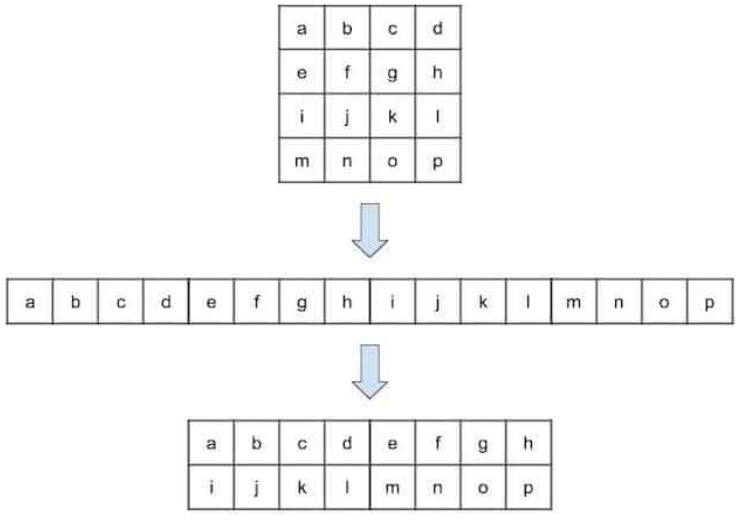
- ***Visualize data for understanding:***





- ***Split & reshape dataset***

After normalization and processing, the data set is divided into three sets: training set, the validation dataset, and the test set. The training set is used to train and refine the model parameters. The test set aims to select good models as well as prevent the model from overfitting. The test set is used to compare the predictability and check the generalizability of the model.



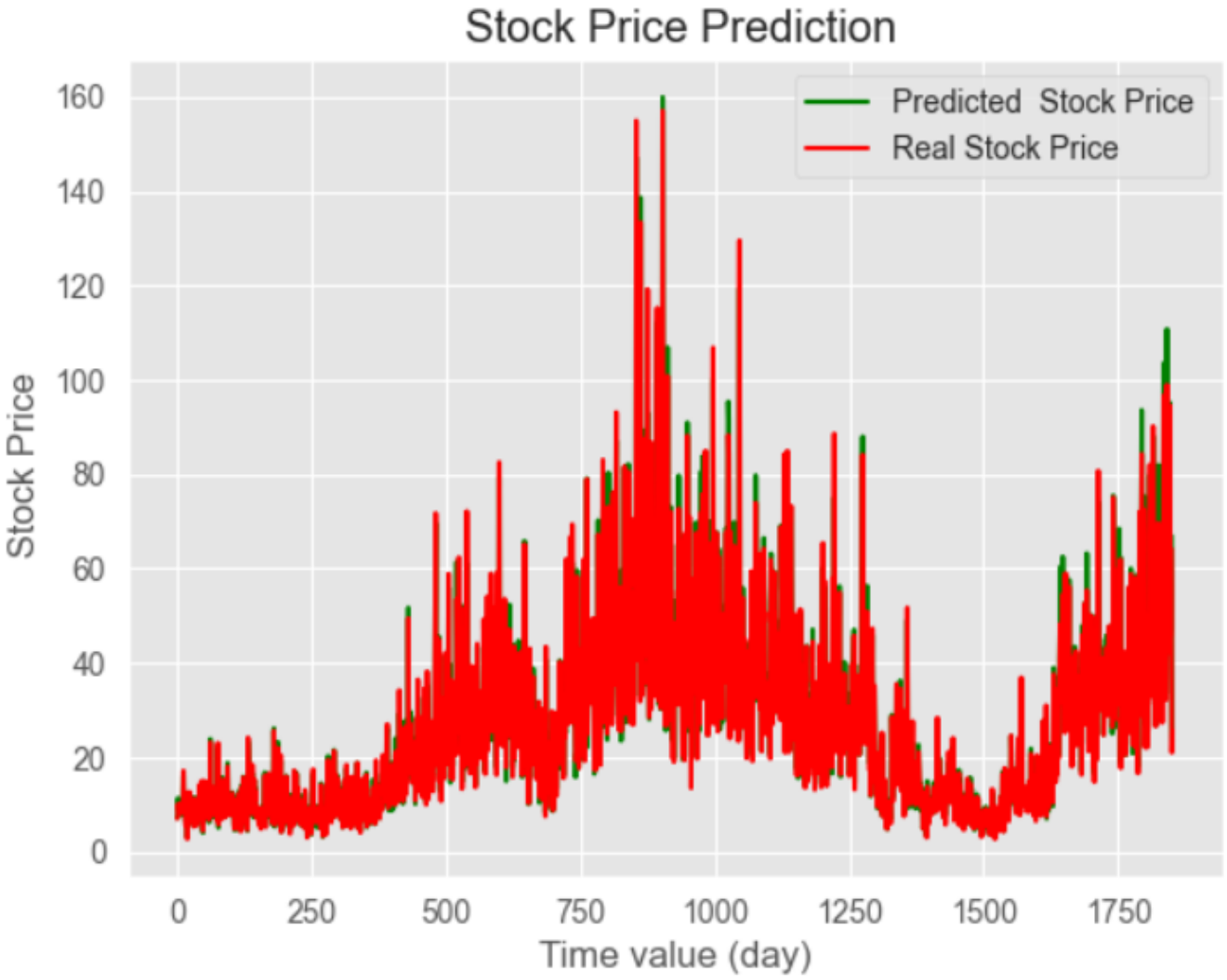
The data has been analyzed, but it must be converted into a [100,1] matrix to make it easier for CNN to train on. Else necessary features won't be selected, and the model will fail.

3. Experiment Results and Analysis

In order to verify the effectiveness of the proposed model, different models including CNN, LSTM, BiLSTM, CNN-LSTM, and CNN-BiLSTM are compared on the stock dataset collected.

Model	MSE	MAE	MAPE
CNN	0.037	0.134	62180.092
LSTM	0.036	0.128	58094.153
CNN-LSTM	0.030	0.1259	50917.610
CNN-BiLSTM	0.0247	0.1247	40050.6094

As shown in these figure, the green curve is the predicted value of the closing price of the stock, and the red curve is the true value of the closing price of the stock. Meanwhile, the x-axis is the time, and the y-axis is the normalization value of the stock price.



The prediction errors of different methods are listed in the table below. As shown in this table, firstly, the CNN model has the low prediction performance among all the compared approaches. Since BiLSTM uses bidirectional information, BiLSTM model can further improve more prediction ability than LSTM. Then, owing to the deep feature extraction of CNN, CNN-LSTM and CNN-BiLSTM model have higher predictive power. In summary, the experiment results verify the feasibility and effectiveness of the proposed network model.

CONCLUSION

For the stock market, learning the future price is very important for making investment decisions. This paper first proposes a new stock price time series forecasting network model (CNN-BiLSTM), which takes the stock closing price, the highest price, the lowest price, the opening price as the input to predict the stock closing price. The proposed network model combines CNN and BiLSTM network models. Firstly, CNN is used to extract the deep features of the input data effectively. Secondly, feature vectors are constructed in time series as inputs to the BILSTM network for learning and prediction. Thus, the useless features are reduced and the accuracy of the model is further improved. The proposed model is compared with CNN, LSTM, BiLSTM, CNN-LSTM, CNN-BiLSTM on the same dataset. The experimental results show that the proposed model has the highest prediction accuracy and the best performance. The MSE, RMSE, and MAE of CNN-BiLSTM are the smallest among all methods, with the values of ... respectively. Compared with single LSTM model, the reduction was ... respectively. It can be seen that it is difficult to achieve high prediction accuracy by using only a single network, but the complexity of the network can improve its prediction accuracy. The CNN-BiLSTM model proposed can effectively predict stock prices and provide relevant references for investors to maximize investment returns.

The existing problem is that the input characteristic parameters of the model selection are relatively single. Therefore, it can be considered to increase the training features of the model, such as the emotional or sentiment features of news events or social media, so as to improve the prediction performance of the model from the perspective of feature selection. And then, we further apply our model on more application fields, such as gold price prediction, oil price prediction, foreign exchange price prediction, novelty detection, and optic disc detection. In addition, a graph-based embedding technology will be introduced to solve the problem of time series prediction.

Data Availability

The network code and data are available upon request here.

References

- Nguyen, M., 2020. DỰ ĐOÁN GIÁ CỔ PHIẾU BẰNG PHƯƠNG PHÁP HỌC SÂU KHÔNG GIÁM SÁT GENERATIVE ADVERSARIAL NETWORK (GAN).
- Duong, T. and Nguy, T., 2022. XÂY DỰNG MÔ HÌNH HỌC SÂU DỰ ĐOÁN XU HƯỚNG GIÁ CHỨNG KHOÁN.
- Ji, L., Zou, Y., He, K. and Zhu, B., 2019. Carbon futures price forecasting based with ARIMA-CNN-LSTM model. Procedia Computer Science, 162, pp.33-38.