Research on Deep Learning and Its Application in Stock Price Prediction

Hoang Van Hai[1], Dang Thi Thu Hien[2]
[1]Thai Nguyen University of Economics and Business Administration, Tan Thinh ward, Thai Nguyen city, Vietnam
[2]Thuy Loi University, 175 Tay Son, Dong Da, Hanoi, Vietnam
Correspondence: Hoang Van Hai (hoanghai@tueba.edu.vn)
Communication: received 20 Nov 2022, revised 16 Feb 2023, accepted 31 Mar 2023
Digital Object Identifier: 10.32913/mic-ict-research.v2023.n1.1136

Abstract: The article studies the problem of forecasting the closing price of a stock based on historical data of a previous day. The paper uses and compares algorithms based on deep learning such as LSTM, BiLSTM, and CNN. The dataset includes data on price, trading volume and some technical indicators related to VCB, MSN, and HPG shares. The results show that CNN performs better for predicting the next day's closing price than the other architectures.

Keywords: *deep learning, stock price prediction, LSTM, BiLSTM, CNN.*

I. Introduction

The stock market has emerged since the 17th century and it is one of the most mercantile entities in the world. The mercantilist aspect of the stock market is the unpredictability of the companies 'stock prices because these prices are determined only by people who are willing to pay for shares of these companies. For attracting people to invest in a company by buying its stock, that company needs to have a good reputation in the public if there is any change in the way people perceive the company, it could positively or negatively affect its stock prices. Because of the general interest in making profits, researchers have tried for decades to predict price movements in the stock market.

Researches on stock market prediction mainly focuses on proposing methods of predicting stock trends effectively. Previous studies can be divided into three categories: feature-oriented methods, model- oriented methods and integration - oriented methods. Feature-oriented methods mainly use statistical-based techniques such as principal component analysis (PCA), independent components analysis (ICA), and information gains (IG) to select features efficiency [10][17] [21]. The performance of a given model will be improved if low relevant features in the input are removed. The model-driven approach focuses on improving the fitting ability of the model. Support vector machine (SVM) has been demonstrated to be very effective for stock market prediction [9]. In recent years, deep learning models have been shown many successful applications in computer vision and natural language processing (NLP) because of their powerful feature extraction capabilities. Therefore, deep learning has also been applied in stock market prediction [1][2][6][7][13][22][24]. Integration-oriented methods often combine various artificial intelligence models or statistical-based techniques to predict the stock market [11][18].

The article applies three deep learning methods, namely Long Short-Term Memory (LSTM), Bidirectional Long Short-Term Memory (BiLSTM), and Convolutional Neural Network (CNN) to predict the stock price of Joint Stock Commercial Bank for Foreign Trade of Vietnam (VCB), Masan Group Corporation (MSN) and Hoa Phat Group Joint Stock Company (HPG). Besides stock data including opening price, closing price, highest price, lowest price, and volume, the article added technical indicators to the analysis to be able to simulate cases of sharp price drops or price spikes due to the influence of exogenous shocks (such as epidemics, wars...). Data is downloaded from investing.com.

The article is organized as follows: part II briefly presents related works, part III presents the design of predictive models, part IV presents data processing and experimental results.

II. Related Works

Althelaya et al in [2] used some variations of LSTM, Gated Recurrent Unit (GRU) to forecast the daily closing price of the S&P500 index based on historical data of the previous 10 days. Specifically, the authors used a bidirectional LSTM (BiLSTM), a bidirectional GRU (BiGRU),

LSTM and GRU with 2 layers stacked to form S-LSTM and S-GRU architectures respectively. The data was taken from Yahoo Finance for the period from 01/01/ 2010 to 30/11/ 2017, and includes five input variables: closing price, opening price, lowest price, highest price, and volume. Data was preprocessed to remove the trend and seasonality by differencing and then normalized between 0 and 1 using min-max normalization. The authors then compared and evaluated the performance of these models. As a result, the S-LSTM architecture demonstrated the highest predictive performance. However, stabilizing time series data removes useful dynamic trends and relationships, and thus we may exclude valuable information. In addition, the study does not include technical indicators in the analysis while they are often referred to by investors to make decisions in buying/selling stocks.

Hoseinzade and Haratizadeh in [7] proposed two CNN-based methods to predict the next day's movement for the S&P 500, NASDAQ, DJI, NYSE and RUSSELL index using information from the previous 60 days. The first method, 2D-CNNpred, is based on the assumption that the actual mapping function from history to the future is an exact function for many markets. Therefore, a single model can predict the future of the market based on its own history. However, to extract the desired mapping function, that model needs to be trained by samples from different markets. In 2D-CNNpred, the input data was aggregated and fed to a specially designed CNN as a two-dimensional tensor consisting of the historical data and features dimension. In the second method, 3D-CNNpred, instead of training a single predictive model that can predict the future of each market based on its own historical data, the research used features from historical information of multiple markets to train a separate prediction model for each market. In 3D-CNNpred, the input data was aggregated and fed to a specially designed CNN as a three-dimensional tensor consisting of historical data, markets, and features. The dataset includes 82 variables for the period from January 2010 to October 2017. This set of variables can be classified into eight different groups, which are fundamental variables, technical indicators, world stock market indices, the exchange rate of the US dollar to other currencies, commodities, data of big US companies, future contracts, and other useful variables. The evaluation shows a significant improvement in the performance of the proposed models compared to baseline algorithms incorporated in the study. Since the research used commodity indices of the stock market of the United States, there will be elements in common among these indices. Therefore, using 2D-CNNpred with a single model that can predict the future of different market based on their own history or 3D-CNNpred with shared hyperparameters among different models for different markets are reasonable. The proof is that these two proposed models both offer better predictive results than the baseline algorithms included in the research. However, if the indexes are replaced by stocks of companies operating in different sectors of an economy, finding the optimal architecture for each stock may be the most effective method to improve prediction accuracy.

Sethia and Raut in [16] used LSTM, GRU, Artificial Neural Network (ANN), and SVM to predict stock prices for t+ 5th day based on historical data of the five previous trading days and thus provide a buying/selling strategy for the Standard's and Poor's 500 index. Data for the period from 03/01/2000 to 30/10/ 2017 taken from Yahoo finance includes opening price, highest price, lowest price, closing price, volume and a number of technical indicators. Independent Components Analysis (ICA) was used to reduce the size of this dataset. A performance comparison between the LSTM, GRU, ANN and SVM models was performed, and the LSTM and GRU models outperformed the remaining models. This result may come from the fact that the authors used ICA in data preprocessing, but because deep learning is very powerful in feature extraction, it is possible that this ICA technique does not contribute much to the results. The key may be that both LSTM and GRU are variants of Recurrent Neural Network (RNN) that is better suited for time series inputs than the other algorithms.

The commonalities of the aforementioned studies are assesments that made on indices, so a same model may not have the same performance if applied to a particular stock. In addition, the choice of window size in the studies has been neither mentioned nor supported by a statistical basis.

III. PROPOSED MODELS

The relationship between variables related to the stock is nonlinear. However, machine learning algorithms are not capable of learning all functions. This limits the problems that these algorithms can solve regarding the complex relationships between variables so stock prediction is a challenge with traditional machine learning methods [3]. Deep learning which is very powerful in feature selection and capable of learning any nonlinear function seems to be a promising approach to this challenge [3]. In deep learning, Artificial Neural Network (ANN), due to having no feedback loop, is only suitable for input data where the data points are independent of each other. Thus, it cannot capture sequential information in the input where a single data point depends on previous observations Therefore, it is difficult for ANN to give an accurate forecast as working with time series [3]. Recurrent Neural Network

(RNN) has a concept of memory which helps them to store the state or information of previous inputs to produce the next output of the sequence, thus more suitable for working with time series in general and stock data in particular [3]. Although RNN works well on time series data, it is difficult to avoid the long-term dependency matter due to the vanishing/exploding problems. Long short-term memory (LSTM) and Bidirectional long short-term memory (BiLSTM) are among the variants of RNN. Furthermore, LSTM was introduced as an efficient way to solve the vanishing/exploding gradient problem by having memory cells to retain time-related information [14].

The article uses an optimizer of stochastic gradient descent (SGD) to train LSTM and BiLSTM model with the expectation that higher performance and faster training can be achieved for related problems.

The formula of SGD as follows [19]:

$$\theta = \theta - \eta \nabla_\theta J\left(\theta; x^{(i:i+n)}; y^{(i:i+n)}\right) \tag{1}$$

Where θ is a vector whose components are the weights to be estimated; η is the learning rate; x^i, y^i is a single training data point and its corresponding label, respectively; n is batch size; and $\nabla_\theta j(\theta; x^{(i:i+n)}; y^{(i:i+n)})$ is the derivative of the loss function at θ. The article chooses n equal to 32.

However, one of the challenges faced by SGD is that when the objective function is not convex, it almost certainly converges to a local minimum. To overcome this limitation of the algorithm, the article uses SGD with momentum.

The formula of momentum as follows [19]:

$$v_{t+1} = \mu v_t - \varepsilon \nabla J(\theta_t) \tag{2}$$

$$\theta_{t+1} = \theta_t + v_{t+1} \tag{3}$$

Where v is velocity or momentum, ε is the learning rate, $\mu \in [0, 1]$ is the momentum coefficient with common values of 0.9 and 0.99, and $\nabla J(\theta_t)$ is the derivative at θ_t . The article chooses μ equal to 0.9.

Momentum along with the learning rate can improve the ability to detect a better set of weights in fewer training epochs. In practice, it is necessary to gradually reduce the learning rate over time. This is because SGD introduces a noise source that does not disappear even when we reach the minimum. Therefore, the paper uses a descending learning rate based on the number of epochs.

The formula for calculating the decreasing learning rate [5]:

$$learningrate = learningrate * 1/(1 + daycay * \#epoch) \tag{4}$$

$$decay = learningrate/epoch \tag{5}$$

The article chooses epoch equal to 200. The initial value of the learning rate will be determined through experiment.

The SGD algorithm with momentum solves the problem that gradient descent does not reach the global minimum, but only stops at the local minimum. However, when approaching the target, it still takes a long time to oscillate back and forth before coming to a complete stop, which is explained by the presence of momentum [20].

To overcome this drawback, depending on the characteristics of the data of each stock, Nesterov's Accelerated Gradient (NAG) can be used in the hope that it will help the algorithm become smarter to know where it proceeds to and slows down when approaching the global minimum. Whether or not to use NAG will be determined through experiment.

The Formula for calculating NAG [19]:

$$v_{t+1} = \mu v_t - \varepsilon \nabla J(\theta_t + \mu v_t) \tag{6}$$

$$\theta_{t+1} = \theta_t + v_{t+1} \tag{7}$$

NAG changes v in a faster and more responsible way, allowing it to behave more consistently than momentum in many situations, especially for higher μ values.

Early Stopping is used to help stop training when there is no improvement in the loss function value after a number of epochs.

Dropout can be used after each hidden layer. This is a fine-tuning method that can effectively reduce model overfitting and improve the model's prediction performance. Whether or not to use this technique and, if used, what the dropout rate is will be determined through experiment.

Convolutional neural network (CNN) is well suited to data with spatial relationships among observations [3], and thus can be suitable for time-relational data such as time series. In addition, CNN with the parameter sharing characteristic can learn from both the current timestep and previous timesteps, allowing them to capture both short-term and long-term patterns in the data. This makes this algorithm promising with time series data as input. Last but not least, CNN with suitable activation functions such as relu will become less sensitive to the vanishing/exploding gradient problem, which prevents deep models from being trained effectively. The article chooses a Multichannel CNN model, whereby each one-dimensional time series as input will be provided to the model as a separate channel. The model is compiled using the Adam optimizer with the learning rate determined through experiment and the loss function used is mean squared error.

IV. EXPERIMENTAL RESULTS

1. Experimental data

The article selects stocks of three companies representing three different industries in the economy according to the Global Industry Classification Standard: developed by Standard & Poor's and MSCI Barra. Russel Global Sectors, and Industry Classification Benchmark: co-developed by Dow Jones and FTSE. Specifically:

1. Joint Stock Commercial Bank for Foreign Trade of Vietnam (VCB)

2. Masan Group Corporation (MSN)

3. Hoa Phat Group Joint Stock Company (HPG)

VCB is in the financial sector, MSN is in the field of essential goods, and HPG is in the production of basic materials. All three companies were included in the VN30 index at the date of 11/03/2022.

Data to determine the performance of proposed methods is downloaded from investing.com. The original data includes opening price, closing price, highest price, lowest price, volume, VN-index, and the yield on 10-year government bond. Some technical indicators are added into the analysis since they are frequently used by traders to better understand the supply and demand of securities and market sentiment. Together, these indicators form the basis of technical analysis providing investors with buying and selling signals. Data of these features is built by Excel based on the original data aforementioned.

Thus, the dataset of each stock includes 21 features as follows:

Original data

1. open: Opening stock price of a day

2. close: Closing stock price of a day

3. high: Highest stock price in a day

4. low: Lowest stock price in a day

5. volume: Number of shares traded in a day

6. VNINDEX: VN-index

7. Y10_T: Yield on 10-year government bondTechnical Indicators

8. rsi: Relative Strength Indicator

$$RSI = 100 - \left[\frac{100}{1+\frac{Average\_gain}{Average\_loss}}\right].$$

- Average_gain: Average gains, Average_loss: Average losses.

- The average gains or losses used in this calculation is the average percentage of gains or losses over the lookback periods, which is 14 days in the article.

9. MACD: Moving Average Convergence Divergence;

$$MACD = EMA(12) - EMA(26)$$

EMA (12): Exponential moving average of 12 days;

EMA (26): Exponential moving average of 26 days.

10. ADX: Average Directional Index (see in [8])

11. % R: Williams Percent Range

$$\%R = \frac{Highest\_High - Close}{Highest\_High - Lowest\_Low} \times (-100);$$

Highest_High: The highest price during the lookback period, which is 14 days in the article.

Close: Most recent closing price.

Lowest_Low : The lowest price in the lookback period, which is 14 days in the article.

12. cci (14): Commodity Channel Index

$$cci(14) = \frac{Typical\_Price - MA}{0.015 \times Mean\_Deviation}$$

$$Typical\_Price = \sum_{i=1}^{P} ((High + Low + Close) \div 3)$$

$$p = 14$$

$$MA = (\sum_{i=1}^{P} Typical\_Price) \div P$$

$$Mean\_Deviation = (\sum_{i=1}^{P} |Typical\_Price - MA|) \div P$$

13. MA (5): Moving average of 5 days

14. MA (10): Moving average of 10 days

15. MA (20): Moving average of 20 days

16. ATR: Average True Range

$$ATR = \frac{1}{n}\sum_{i=1}^{n} TR_i$$

$$TR = Max\left[(H - L), \left|H - C_p\right|, \left|L - C_p\right|\right]$$

H: Today's highest price

L: Today's lowest price

C_p: Yesterday's closing price

$n = 14$

17. Ult Osc: Ultimate Oscillator

$$UltOsc = \left[\frac{A_7 \times 4 + A_{14} \times 2 + A_{28}}{4+2+1}\right] \times 100$$

A: Average

BP (Buying pressure) = Close – Min (Low, PC)

TR (True Range) = Max (High, PC) – Min (Low, PC)

PC: Prior Close

High: Today's highest price

Low: Today's lowest price

$$Average_7 = \frac{\sum_{p=1}^{7} BP}{\sum_{p=1}^{7} TR}$$

$$Average_{14} = \frac{\sum_{p=1}^{14} BP}{\sum_{p=1}^{14} TR}$$

$$Average_{28} = \frac{\sum_{p=1}^{18} BP}{\sum_{p=1}^{28} TR}$$

18. ROC: Price Rate of Change

$$ROC = \left(\frac{Close\_price_p - Close\_price_{p-n}}{Close\_price_{p-n}}\right) \times 100$$

Close_pricep: Closing price of most recent period

Close_pricep-n: Closing price n periods before most recent period

n = 13

19. STOCH (9,6): Stochastic oscillator

$\%K = \left(\frac{C-L9}{H9-L9}\right) \times 100$

C: The most recent closing price

L9: The lowest price traded of the 9 previous trading sessions

H9: The highest price traded during the same 9-day period %K: The current value of the stochastic indicator

STOCH (9,6): 6-period moving average of %K.

20. STOCHRSI (14):

$STOCHRSI = \frac{RSI - \min[RSI]}{\max[RSI] - \min[RSI]}$

RSI: Current RSI reading

min [RSI]: Lowest RSI reading over the last 14 periods

max [RSI]: Highest RSI reading over the last 14 periods

21. Bull/Bear: Bull/Bear Ratio (see in [12])

One problem that the article finds with the original data is that the sampling time is not the same. For example, we have the first date that is the first day of April (2015–04–01), the second one that is the second day of April (2015–04–02) and the third one that is the sixth day (2015–04–06).). The paper solves this problem in two steps based on the technique presented in [15]. Specifically, step 1 is to create a continuous time space from the start date to the end date. For example, with the aforementioned example, time space will include dates in the order 2015–04–01, 2015–04–02, 2015–04–03, 2015–04–04, 2015–04–05, 2015 –04–06. Step 2, at each time series, the missing value of a certain date will be filled with the value of the closest day available.

After using Excel to build technical indicators, with all three datasets, there are cells with no data in the first observations. To deal with this problem, with the dataset of VCB, observations from 12/03/2012 to 08/04/2012 are removed, with dataset of MSN, observations from 29/07/2013 to 25/08/2013 are not included, and with the HPG dataset, observations from 05/07/2013 to 01/08/2013 are excluded. The data processing results in the dataset of VCB including 3624 days during the period from 09/04/2012 to 11/03/2022, dataset of MSN including 3120 days for the period from 26/08/2013 to 11/03/2022, and the dataset of HPG including 3144 days for the period from 02/08/2013 to 11/03/2022.

Since data type of all features is numeric, the paper uses the Pearson correlation coefficient matrix in feature selection. If the absolute value of the Pearson correlation coefficient between a feature and the target variable of close is less than 0.05, that feature is not included in the proposed models. For VCB stock, Figure 4.1 shows that the correlation coefficient between close and STOCHRSI (14), ROC, and Bull_Bear is 0.011, 0.043, and -0.0032, respectively. Therefore, the three variables STOCHRSI (14), ROC, and Bull_Bear are not selected as inputs. For MSN stock, Figure 4.2 shows that the correlation coefficients between close and STOCHRSI (14), ADX, Ult Osc, and Bull_Bear are 0.0093, 0.034, -0.026, and -0.012, respectively. Therefore, the four variables STOCHRSI (14), ADX, Ult Osc, and Bull_Bear were not selected as inputs. For HPG stock, Figure 4.3 shows that the correlation coefficient between close and STOCHRSI (14), ADX, and Bull_Bear is -0.0066, -0.021, and -0.039, respectively. Therefore, the three variables STOCHRSI (14), ADX, and Bull_Bear are not chosen as inputs.

Thus, applying this technique to the data of three types of stocks, the article in turn obtains the following results:

- With VCB, the features included in the proposed models are close, open, high, low, volume, rsi, STOCH, MACD, ADX, %R, cci (14), MA (5), MA (10), MA (20), ATR, Ult Osc, VNINDEX, and Y10_T
- With MSN, the features included in the proposed models are: close, open, high, low, volume, rsi, STOCH, MACD, %R, cci (14), MA (5), MA (10), MA (20), ATR, ROC, VNINDEX, and Y10_T
- With HPG, the features included in the proposed models are close, open, high, low, volume, rsi, STOCH, MACD, %R, cci (14), MA (5), MA (10), MA (20). , ATR, Ult Osc, ROC, VNINDEX, and Y10_T.

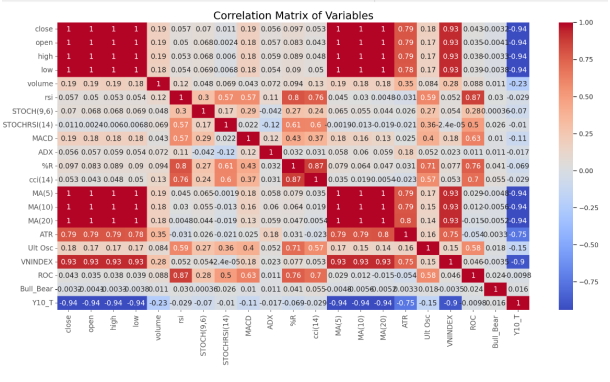

Figure 1. Correlation coefficients Matrix of VCB

To select the window size, the close time series is tested for autocorrelation. The article uses *stattools.arma_order_select_ic technique* with parameters *max_ar = 20, max_ma = 0, ic = ['bic'], trend = 'c'* for close time series of all 3 stocks. The results show that all three series in question have first order autocorrelation and so the window size is chosen to be 1. Thus, the length of

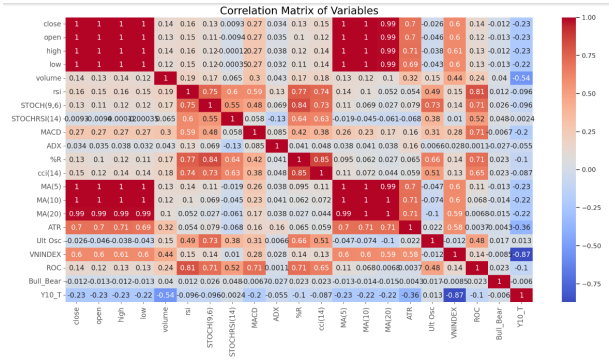

Figure 2. Correlation coefficients Matrix of MSN

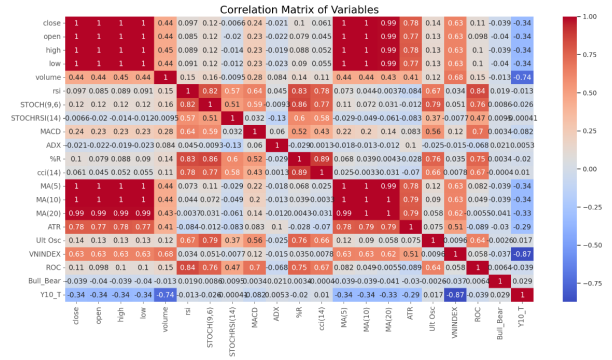

Figure 3. Correlation coefficients Matrix of HPG

the history is 1 day. In other words, for each prediction of the next day's closing price, the models use information from the previous day including the value of this variable (*close*) itself.

Variables measured at different scales do not contribute equally to the learned function of the model and may eventually produce bias. Therefore, to solve this potential problem, the z-score standardization method is applied to normalize the input data.

The input data is normalized according to the formula:

$$y_i = \frac{x_i - \overline{x}}{s} \tag{8}$$

where y_i is the normalized value, x_i is the input data, $\overline{x}$ is the mean value of the input data, and s is the standard deviation of the input data.

Covid-19 began to affect the Vietnamese stock market from the end of January 2020. Thus, in order for the training set to include observations during the period when the market is affected by this epidemic, it must contain pieces of data after January 2020. The goal of the paper is the training set to include as many records as possible during the Covid-19 period while still ensuring a large enough number of observations in the testing set in the hope of increasing the predictive performance of the algorithms. Derving from the data size of each stock (3624 days in the VCB's dataset, 3120 days in the MSN's dataset, and 3144 days in the HPG's dataset) and the last observation in the data of all three stocks is 11/03/2022, the paper takes first 90% of the dataset as the training set and the remaining 10

2. Evaluating methodology

We used two performance measures to assess which forecasting model accurately represents the actual data. The measures include Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). The mathematical equations of the performance measures are defined as follows:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|y_{true}^{i} - y_{pred}^{i}\right| \tag{9}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(y_{true}^{i} - y_{pred}^{i}\right)^2} \tag{10}$$

where N is the sample size, y_{true}^{i} is the observed value, and y_{pred}^{i} is the predicted value.

All methods are implemented using Python and Keras, an open-source learning library based on TensorFlow. All experiments are performed in the operating environment of Intel i5-1035G1 1.19 GHz, 8GBs of RAM, and Windows 10.

3. Experiment settings

The architectures proposed in the article are the best predictive performance ones among the experimented architectures.

a) Long short-term memory

With VCB, the architecture of LSTM includes: an LSTM hidden layer with 18 neurons, input shape = (1,18) where 1 is the window size and 18 is the number of features. Dropout with the parameter of 0.5 is used. In other words, 50% of the number of neurons in the hidden layer along with all its incoming and outgoing connections will be temporarily removed from the network after each iteration in the training process to deal with the overfitting problem. Finally, there is an output layer with 1 neuron to predict the closing price.

With MSN, the architecture of LSTM includes: an LSTM hidden layer with 16 neurons, input shape = (1,17) where 1 is the window size and 17 is the number of features. Dropout with the parameter of 0.1 is used. In other words, 10% of the number of neurons in the hidden layer along

```
model = Sequential()
model.add(LSTM(10, input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dropout(0.5))
model.add(Dense(1))
epochs = 200
learning_rate = 0.1
decay_rate = learning_rate / epochs
mo= 0.9
sgd = SGD(learning_rate=learning_rate, momentum=mo, decay=decay_rate, nesterov=True)
model.compile(loss='mse', optimizer=sgd)
monitor = EarlyStopping(monitor='val_loss', min_delta=1e-3, patience=5,verbose=1, mode='auto', restore_best_weights=True)
model.fit(X_train, y_train, epochs=epochs, batch_size=32, validation_data=(X_test, y_test),  callbacks=[monitor], verbose=2, shuffle=False)
```

Figure 4. LSTM architecture corresponding to VCB

with all its incoming and outgoing connections will be temporarily removed from the network after each iteration in the training process to deal with the overfitting problem. Finally, there is an output layer with 1 neuron to predict the closing price.

With HPG, the architecture of LSTM includes: an LSTM hidden layer with 10 neurons, input shape = (1,18) where 1 is the window size and 18 is the number of features. Dropout with the parameter of 0.1 is used. In other words, 10% of the number of neurons in the hidden layer along with all its incoming and outgoing connections will be temporarily removed from the network after each iteration in the training process to deal with the overfitting problem. Finally, there is an output layer with 1 neuron to predict the closing price.

b) Bidirectional Long short-term memory

With VCB, the architecture of BiLSTM includes: two BiLSTM hidden layers with 21 neurons in each layer, input shape = (1,18) where 1 is the window size and 18 is the number of features. Dropout with the parameter of 0.1 is used between the two hidden layers and between the final hidden layer and the output layer. In other words, 10% of the number of neurons in each hidden layer along with all its incoming and outgoing connections will be temporarily removed from the network after each iteration in the training process to deal with the overfitting problem. Finally, there is an output layer with 1 neuron to predict the closing price.

With MSN, the architecture of BiLSTM includes: two BiLSTM hidden layers with 9 neurons in each layer, input shape = (1,17) where 1 is the window size and 17 is the number of features. Dropout with the parameter of 0.1 is used between the two hidden layers and between the final hidden layer and the output layer. In other words, 10% of the number of neurons in each hidden layer along with all its incoming and outgoing connections will be temporarily removed from the network after each iteration in the training process to deal with the overfitting problem. Finally, there is an output layer with 1 neuron to predict the closing price.

With HPG, the architecture of BiLSTM includes: two BiLSTM hidden layers with 11 neurons in each layer, input shape = (1,18) where 1 is the window size and 18 is the number of features. Dropout with the parameter of 0.5 is used between the two hidden layers and between the final hidden layer and the output layer. In other words, 50% of the number of neurons in each hidden layer along with all its incoming and outgoing connections will be temporarily removed from the network after each iteration in the training process to deal with the overfitting problem. Finally, there is an output layer with 1 neuron to predict the closing price.

c) Convolutional neural network

With VCB the architecture of CNN includes: A 1D convolution layer applying 72 filters with kernel size of 1, activation function is relu, input shape = (1,18) where 1 is window size and 18 is the number of features. Following the 1D convolutional layer is the max pooling layer with pool_size = 1 and the fully connected layer with 35 neurons and the activation function of relu. Finally, there is an output layer with 1 neuron to predict the closing price.

The model is compiled using the Adam optimizer with a learning rate of 0.0001 and the loss function used is the mean squared error.

With MSN the architecture of CNN includes: Two 1D convolution layer applying 16 filters with kernel size of 1, activation function is relu, input shape = (1,17) where 1 is window size and 17 is the number of features. Following the 1D convolutional layer is the max pooling layer with pool_size = 1 and the fully connected layer with 20 neurons and the activation function of relu. Finally, there is an output layer with 1 neuron to predict the closing price.

The model is compiled using the Adam optimizer with a learning rate of 0.01 and the loss function used is the mean squared error.

With HPG the architecture of CNN includes: A 1D convolution layer applying 56 filters with kernel size of 1, activation function is relu, input shape = (1,18) where 1 is

```
model = Sequential()
model.add(LSTM(16, input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dropout(0.1))
model.add(Dense(1))
epochs = 200
learning_rate = 0.001
decay_rate = learning_rate / epochs
mo= 0.9
sgd = SGD(learning_rate=learning_rate, momentum=mo, decay=decay_rate, nesterov=False)
model.compile(loss='mse', optimizer=sgd)
monitor = EarlyStopping(monitor='val_loss', min_delta=1e-3, patience=5, verbose=1, mode='auto', restore_best_weights=True)
model.fit(X_train, y_train, epochs=epochs, batch_size=32, validation_data=(X_test, y_test),  callbacks=[monitor], verbose=2, shuffle=False)
```

Figure 5. LSTM architecture corresponding to MSN

```
model = Sequential()
model.add(LSTM(10, input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dropout(0.1))
model.add(Dense(1))
epochs = 200
learning_rate = 0.1
decay_rate = learning_rate / epochs
mo= 0.9
sgd = SGD(learning_rate=learning_rate, momentum=mo, decay=decay_rate, nesterov=False)
model.compile(loss='mse', optimizer=sgd)
monitor = EarlyStopping(monitor='val_loss', min_delta=1e-3, patience=5,verbose=1, mode='auto', restore_best_weights=True)
model.fit(X_train, y_train, epochs=epochs, batch_size=32, validation_data=(X_test, y_test),  callbacks=[monitor], verbose=2, shuffle=False)
```

Figure 6. LSTM architecture corresponding to HPG

```
model = Sequential()
model.add(Bidirectional(LSTM(21, return_sequences = True,input_shape=(X_train.shape[1], X_train.shape[2]))))
model.add(Dropout(0.1))
model.add(Bidirectional(LSTM(21)))
model.add(Dropout(0.1))
model.add(Dense(1))
epochs = 200
learning_rate = 0.1
decay_rate = learning_rate / epochs
mo= 0.9
sgd = SGD(learning_rate=learning_rate, momentum=mo, decay=decay_rate, nesterov=False)
model.compile(loss='mse', optimizer=sgd)
monitor = EarlyStopping(monitor='val_loss', min_delta=1e-3, patience=5, verbose=1, mode='auto', restore_best_weights=True)
model.fit(X_train, y_train, epochs=epochs, batch_size=32, validation_data=(X_test, y_test),  callbacks=[monitor], verbose=2, shuffle=False)
```

Figure 7. BiLSTM architecture corresponding to VCB

```
model = Sequential()
model.add(Bidirectional(LSTM(9, return_sequences = True,input_shape=(X_train.shape[1], X_train.shape[2]))))
model.add(Dropout(0.1))
model.add(Bidirectional(LSTM(9)))
model.add(Dropout(0.1))
model.add(Dense(1))
epochs = 200
learning_rate = 0.1
decay_rate = learning_rate / epochs
mo= 0.9
sgd = SGD(learning_rate=learning_rate, momentum=mo, decay=decay_rate, nesterov=False)
model.compile(loss='mse', optimizer=sgd)
monitor = EarlyStopping(monitor='val_loss', min_delta=1e-3, patience=5, verbose=1, mode='auto', restore_best_weights=True)
history = model.fit(X_train, y_train, epochs=epochs, batch_size=32, validation_data=(X_test, y_test),  callbacks=[monitor], verbose=2, shuffle=False)
```

Figure 8. BiLSTM architecture corresponding to MSN

window size and 18 is the number of features. Following the 1D convolutional layer is the max pooling layer with pool size = 1 and the fully connected layer with 40 neurons and the activation function of relu. Finally, there is an output layer with 1 neuron to predict the closing price

The model is compiled using the Adam optimizer with a learning rate of 0.01 and the loss function used is the mean squared error.

```
model = Sequential()
model.add(Bidirectional(LSTM(11, return_sequences = True,input_shape=(X_train.shape[1], X_train.shape[2]))))
model.add(Dropout(0.5))
model.add(Bidirectional(LSTM(11)))
model.add(Dropout(0.5))
model.add(Dense(1))
epochs = 200
learning_rate = 0.1
decay_rate = learning_rate / epochs
mo= 0.9
sgd = SGD(learning_rate=learning_rate, momentum=mo, decay=decay_rate, nesterov=True)
model.compile(loss='mse', optimizer=sgd)
monitor = EarlyStopping(monitor='val_loss', min_delta=1e-3, patience=5, verbose=1, mode='auto', restore_best_weights=True)
history = model.fit(X_train, y_train, epochs=epochs, batch_size=32, validation_data=(X_test, y_test),  callbacks=[monitor], verbose=2, shuffle=False)
```

Figure 9. BiLSTM architecture corresponding to HPG

```
model = Sequential()
model.add(Conv1D(filters=72, kernel_size=1, activation='relu', input_shape=(n_timesteps,n_features)))
model.add(MaxPooling1D(pool_size=1))
model.add(Flatten())
model.add(Dense(35, activation='relu'))
model.add(Dense(n_outputs))
model.compile(loss='mse', optimizer= keras.optimizers.Adam(0.0001))
model.fit(train_x, train_y, epochs=200, batch_size=32, verbose=verbose)
```

Figure 10. CNN architecture corresponding to VCB

```
model = Sequential()
model.add(Conv1D(filters = 16, kernel_size=1, activation='relu', input_shape=(n_timesteps,n_features)))
model.add(MaxPooling1D(pool_size=1))
model.add(Conv1D(filters=16, kernel_size=1, activation='relu'))
model.add(MaxPooling1D(pool_size=1))
model.add(Flatten())
model.add(Dense(20, activation='relu'))
model.add(Dense(n_outputs))
model.compile(loss='mse', optimizer= keras.optimizers.Adam(0.01))
model.fit(train_x, train_y, epochs=200, batch_size=32, verbose=verbose)
```

Figure 11. CNN architecture corresponding to MSN

```
model = Sequential()
model.add(Conv1D(filters=56, kernel_size=1, activation='relu', input_shape=(n_timesteps,n_features)))
model.add(MaxPooling1D(pool_size=1))
model.add(Flatten())
model.add(Dense(40, activation='relu'))
model.add(Dense(n_outputs))
model.compile(loss='mse', optimizer= keras.optimizers.Adam(0.01))
model.fit(train_x, train_y, epochs=200, batch_size=32, verbose=verbose)
```

Figure 12. CNN architecture corresponding to HPG

d) Baseline algorithms

To demonstrate the effectiveness of the proposed methods, these methods are compared with ARIMA, a popular and widely used statistical method for time series forecasting, using the same training and testing data in the same operating environment. As a univariate model, the article runs ARIMA only on the closing price.

A standard notation used of ARIMA(p,d,q) where parameters are replaced with integer values to quickly indicate the particular ARIMA model in use. The parameters of the ARIMA model are determined as follows:

- p: A number of lag observations included in the

model, also called the lag order. Since there is an autocorrelation of 1 in the close series of all stocks, p is chosen to be 1 .

- d: A number of times that the raw observations are differenced, also called the degree of differencing to make the time series stationary. In the article, d equals 1 is enough to make the close time series in all stocks stationary.
- q: The size of the moving average window, also called the order of the moving average. The value of q is determined through experiment.

After conducting experiments with the data of the aforementioned three stocks, the paper finds the optimal ARIMA models corresponding to VCB, MSN, and HPG, are ARIMA(1,1,0) and ARIMA(1,1,1) respectively.

V. Resulting comparison

For each stock, all methods use the same training and testing set. All methods are implemented using Python and Keras, an open-source learning library based on TensorFlow. All experiments are performed in the operating environment of Intel i5-1035G1 1.19 GHz, 8GBs of RAM, and Windows 10.

Each prediction algorithm is executed multiple times. Averages of the results of all the experiments conducted per each technique are compared. Models that generate the minimum performance measures on the testing data are also selected and compared.

Tables I and II show the averages of RMSE and MAE for each stock's architecture. Comparisons between models generating the minimum RMSE and MAE of each stock are shown in Tables III and IV, respectively. The results **in bold** are the best results in the algorithms surveyed in the article. It is clear that in predicting the next day's closing price, the CNN models beat the rest including the ARIMA baseline models for all three stocks. This is proof that CNN is an effective and promising tool in stock price forecasting

The average comparison of the results in Tables I and II shows that the ARIMA baseline algorithm has the second highest prediction accuracy in all surveyed stocks. LSTM with prediction accuracy ranks 3rd in VCB, and 4th in MSN and HPG. BILSTM has the 3rd highest prediction accuracy in HPG and MSN, and 4th in VCB.

According to the results in table III and IV, we see that the ARIMA baseline algorithm has the second highest prediction accuracy in all surveyed stocks. LSTM with prediction accuracy ranks 3rd in VCB, 4th in MSN and HPG. BILSTM has the 3rd highest prediction accuracy in stocks including MSN and HPG, and 4th in VCB stocks.

TABLE I
RMSE IN AVERAGE OF ALL MODELS FOR ALL STOCKS

Models	VCB	MSN	HPG
LSTM	4877.394	31870.690	8238.961
BiLSTM	8183.190	29020.730	4252.388
CNN	860.530	1568.830	655.909
ARIMA	1126.409	2212.868	868.072

TABLE II
MAE IN AVERAGE OF ALL MODELS FOR ALL STOCKS

Models	VCB	MSN	HPG
LSTM	3771.708	28278.280	7246.288
BiLSTM	6672.860	26050.260	3472.885
CNN	647.325	1267.096	527.298
ARIMA	687.585	1396.499	541.981

TABLE III
MINIMUM RMSE OF ALL MODELS FOR ALL STOCKS

LSTM	4452.224	30768.223	5074.592
BiLSTM	5176.616	11960.404	3757.165
CNN	**635.660**	**1188.001**	**607.661**
ARIMA	1126.409	2212.868	868.072

TABLE IV
MINIMUM MAE OF ALL MODELS FOR ALL STOCKS

LSTM	3598.402	26753.147	4262.372
BiLSTM	4254.300	9736.850	3035.655
CNN	**472.443**	**905.085**	**460.292**
ARIMA	687.585	1396.499	541.981

VI. Conclusion

The article studies the problem of time series forecasting in finance, specifically the problem of forecasting the closing price of stocks based on historical data of a previous day. Besides stock data including opening price, closing price, highest price, lowest price, and volume, the article added technical indicators to the analysis to be able to simulate cases of sharp price drops or price spikes due to the influence of exogenous shocks (such as epidemics, wars....). Moreover, the paper also uses and compares algorithms such as LSTM, BiLSTM, CNN, and ARIMA. The results show that CNN performs better for predicting the next day's closing price than the remaining models. This is a proof that CNN is an effective and promising tool in stock price forecasting.

References

[1] Achkar, R., Elias-Sleiman, F., Ezzidine, H., & Haidar, N. (2018), "Comparison of BPA-MLP and LSTM-RNN for Stocks Prediction", *2018 6th International Symposium on Computational and Business Intelligence* (ISCBI). doi:10.1109/iscbi.2018.00019

[2] Althelaya, K. A., El-Alfy, E.-S. M., & Mohammed, S. (2018), "Stock Market Forecast Using Multivariate Analysis with Bidirectional and Stacked (LSTM, GRU)", *2018 21st*

Saudi Computer Society National Computer Conference (NCC). doi:10.1109/ncg.2018.8593076

[3] Aravindpai, P.: CNN vs. RNN vs. ANN – Analyzing 3 Types of Neural Networks in Deep Learning. https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/

[4] Bengio, Y. (2012), "Practical Recommendations for Gradient-Based Training of Deep Architectures", *Neural Networks: Tricks of the Trade*, 437–478. doi:10.1007/978-3-642-35289-8_26

[5] Brownlee, J.:Using Learning Rate Schedules for Deep Learning Models in Python with Keras.https://machinelearningmastery.com/using-learning-rate-schedules- deep-learning-models-python-keras/

[6] Chen, W., Zhang, Y., Yeo, C. K., Lau, C. T., & Lee, B. S. (2017)," Stock market prediction using neural network through news on online social networks", *2017 International Smart Cities Conference (ISC2)*. doi:10.1109/isc2.2017.8090834

[7] Hoseinzade, E., & Haratizadeh, S. (2019), "CNNpred: CNN-based stock market prediction using a diverse set of variables", *Expert Systems with Applications*. doi:10.1016/j.eswa.2019.03.029

[8] Jay, A.: How to do Average Directional Index (ADX) in Excel: https://investsolver.com/how-to-do-average-directional-index-adx-in-excel/

[9] Kara, Y., Boyacioglu, M.A., Baykan, Ö.K.(2011), "Predicting direction of stock price index movement using artificial neural networks and support vector machines:The sample of the istanbul stock exchange", *Expert Syst. Appl*. 38 (5), 5311–5319.

[10] Lee, M.-C. (2009), "Using support vector machine with a hybrid feature selection method to the stock trend prediction", *Expert Syst. Appl*. 36 (8), 10896–10904.

[11] Luo, L., Chen, X. (2013), "Integrating piecewise linear representation and weighted support vector machine for stock trading signal prediction", *Appl. Soft Comput*.13 (2), 806–816.

[12] Mark, U.: How to Calculate the Elder Ray Technical Indicator using Excel: https://www.tradinformed.com/how-to-calculate-the-elder-ray-technical-indicator-in-excel/.

[13] Mehtab, S. & Sen, J. (2020), "Stock Price Prediction Using Convolutional Neural Networks on a Multivariate Timeseries", https://doi.org/10.48550/arXiv.2001.09769.

[14] Olah, C.: Understanding LSTM Networks http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[15] Paialunga, P.: Noise cancellation with Python and Fourier Transform https://towardsdatascience.com/noise-cancellation-with-python-and-fourier-transform-97303314aa71.

[16] Ruder, S. (2017), "An overview of gradient descent optimization algorithms", https://doi.org/10.48550/arXiv.1609.04747.

[17] Sethia, A., & Raut, P. (2018), "Application of LSTM, GRU and ICA for Stock Price Prediction", *Smart Innovation, Systems and Technologies*, 479–487.

[18] Sonkiya, P. , Bajpai, V. and Bansal, A. (2021), "Stock price prediction using BERT and GAN", *In Proceedings of ACM Conference* (Conference'17). ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/.

[19] Sutskever, I., Martens, J., Dahl, G. and Hinton, G. (2013), "On the importance of initialization and momentum in deep learning", *Proceedings of the 30th International Conference on Machine Learning*, PMLR 28(3):1139-1147, 2013.

[20] Tran .T. T.: Optimizer- https://viblo.asia/p/optimizer-hieu-sau-ve-cac-thuat-toan-toi-uu-gdsgdadam-Qbq5QQ9E5D8.

[21] Tsai, C.-F., & Hsiao, Y.-C. (2010), "Combining multiple feature selection methods for stock prediction: Union, intersection, and multi- intersection approaches", *Decision Support Systems*, 50(1), 258–269. doi:10.1016/j.dss.2010.08.028.

[22] Zhang, K., Zhong, G., Dong, J., Wang, S., & Wang, Y. (2019)," Stock Market Prediction Based on Generative Adversarial Network", *Procedia Computer Science*, 147, 400–406. doi:10.1016/j.procs.2019.01.256.

[23] Zhang, Q., Wang, R., Qi, Y. and Wen, F. (2022)," A watershed water quality prediction model based on attention mechanism and Bi-LSTM", *Environ Sci Pollut Res (2022)*. https://doi.org/10.1007/s11356-022-21115-y.

[24] Zhou, X., Pan, Z., Hu, G., Tang, S., & Zhao, C. (2018), "Stock Market Prediction on High-Frequency Data Using Generative Adversarial Nets", *Mathematical Problems in Engineering*, 2018, 111. doi:10.1155/2018/4907423.

Hoang Van Hai received a B.Sc. degree in Mathematics from Thai Nguyen University of Education in 2001, completed M.Sc. degree in Public policy at Antwerp University in 2012, and has completed M.Sc. degree in Data science at Hanoi University of Sciences in 2023. He has currently been working at Faculty of Economics, University of Economics and Business Administration, Thai Nguyen University. His research interests are Deep learning, Regression, Machine learning.
Email: hoanghai@tueba.edu.vn

Dang Thi Thu Hien has currently been working at Faculty of Computer Science and Engineering, Thuy Loi University. Her research interests are Artificial intelligence, Neural networks, Deep learning, Regression, Data mining, Machine learning. .
Email: Hiendt@tlu.edu.vn