A study of stock price prediction models based on time series analysis: LSTM and CNN

Luoyuan Zhang

Faculty of Science, Xi’an University of Technology, Xi’an, Shaanxi, 710054, China

2220920020@stu.xaut.edu.cn

Abstract. The stock market is one of the most dynamic and complex systems and stock price prediction is essential for investors to make informed investment decisions and minimize risks. Over the years, various techniques have been developed for this topic. This paper studied two machine learning models for predicting the price of stock, leveraging time series analysis, namely Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN). This essay use time index as labels, and the data was standardized before training the two models with a step-size. A circulation was created for predicting the daily price of stock with the data of the last three days. RMSE and MAE were two indicators used to assess the models in this task. The results of this study indicated that both of the two models performed well in this topic, and the CNN model showed a better performance than LSTM. As a suggestion, investors should consider other factors such as market trends and risk management strategies when relying on these models to ensure a higher accurate result.

Keywords: LSTM, CNN, stock price prediction, time series analysis.

1. Introduction

The stock market is renowned for the unpredictability, nonlinearity, and active essence. Since stock prices are influenced by a number of factors, including the political situation, the global economic environment and more, it can be challenging to predict them. Therefore, predicting future stock prices accurately is critical for investors to make informed investment decisions and minimize risks [1-3]. An effective model that can find the hidden mode and complex associations in an enormous dataset is required to deal with the variety of data. Compared to preceding methods, machine learning techniques have shown a more efficient performance that increased the effectiveness by 60–86% in this field [4].

As an important branch of artificial intelligence (AI), Machine learning (ML) allows computers to learn and make decisions or choices based on data without having to be explicitly programmed. Traditional machine learning algorithms rely on a collection of manually created features and require extensive domain knowledge to create an effective model. These algorithms have been successfully used in a variety of applications and are efficient at solving issues with little data. In contrast, deep learning algorithms, such as recurrent neural networks (RNN) and convolutional neural networks (CNN), can learn representations of data directly from raw input without any domain knowledge. They consist of multiple layers of interconnected nodes that learn hierarchical representations of data, which enables them to extract complex features and patterns from data [5]. When compared to the conventional statistical method, the deep learning method performs significantly better. As one of the major causes, direct analysis can be used by deep learning to convert the raw data to a nonlinear model, which improves the fit of the multilayer neural network. In the area of financial applications, deep learning additionally benefits from self-selection. The majority of financial data is quite unreliable and noisy [6].

This project focused on comparing the performance of LSTM and CNN models in predicting stock prices based on time series analysis and making suggestions for practicing the two models to real market. The paper is ordered as follows. Methods for adding labels, principles of models and prediction are illustrated in Chapter II. The results of the two models are represented in Chapter III, while some practical suggestions are represented in Part IV. In the fifth part, a limitation analysis and some prospects of the models and the project are demonstrated.

2. Method

2.1. Dataset

The dataset used in this program was given by the Kaggle website, which contains data of stock prices from May 2013 to May 2019 for four companies, namely Amazon, Bitcoins, Domino’s Pizza and Netflix. In details, stock price values of every company were set as a column in the dataset as a time series, and the abbreviations of the four companies were set as the index of every column.

2.2. Data preprocessing

In this project, to build models based on time series analysis, the to_datetime() function in pandas was used to change the format of the data in the ‘Date’ column and then set them as an index, which is regarded as labels. Then the data transformed into the mode required by the input shape of the models. In terms of the method for data pre-processing, this paper chose standardization, and all of the data given were narrowed down between 0 and 1. This would have some positive impacts, such as improving comparability, improving accuracy, reducing overfitting and saving time and cost.

When building the models, a circulation was created to generate a dataset with time_steps in LSTM and look_back size in CNN, which operated until it reached the last date and was used to implement the prediction. The thought of prediction in this paper is demonstrated vividly by Figure 1. The circulation and the time_steps or look_back size made it feasible to analyse time series to predict price values. The dataset was divided into two parts, one as train dataset with 80% of the whole segment and the rest part to test the model. After training 300 times, the model predicted the stock price of the first new day with prices of the last 30 days in train segment. Then values of the second new day were predicted with that of the first new day and the last 29 days in train data.

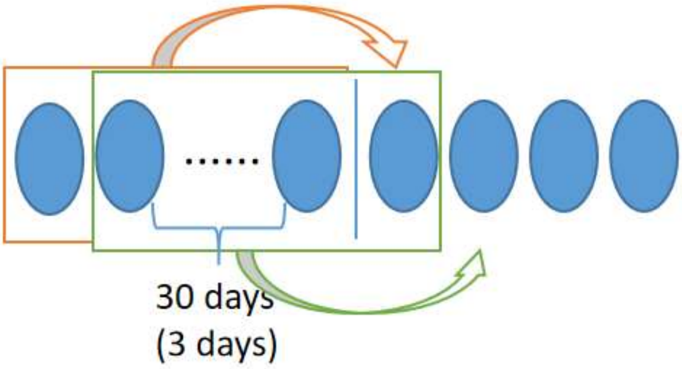

Figure 1. Logistic of circulation and principle of prediction.

2.3. Principle of LSTM

The vanishing gradient problem, which is a frequent difficulty in conventional RNNs, is addressed by the LSTM architecture [7,8]. The fundamental components of an LSTM cell contain three gates, including input, forget and output one. Together, the information flow inside the LSTM cell is regulated by these gates, which enable it to selectively remember or forget data from earlier time steps. At each time step, an LSTM cell receives an input vector and a hidden state vector from the previous one. The input vector is multiplied by a weight matrix, and each member of the input vector is given a value between 0 and 1 after being passed through a sigmoid activation function. The amount of the relevant input element that should be permitted to flow through the input gate is represented by this value. The forget gate chooses which data to omit from the previous hidden state after taking into account the input vector and hidden state. Each component of the previous hidden state is given a forget vector between 0 and 1 after being multiplied by a weight matrix and then put through a sigmoid activation function. This forget vector is then used to remove irrelevant information from the previous hidden state. The output gate combines the input vector and the current hidden state to generate an output vector. This output vector is multiplied by a weight matrix, then run through a sigmoid activation function to give each member a value between 0 and 1. How much of the appropriate output element should be included in the output is indicated by this value. The finishing hidden state is the combination of the input vector, previous hidden state, and the current input that has been allowed to pass through the input gate and the output that has been selected by the output gate. The concealed state is then sent as the previous hidden state to the following time step.

The equations for updating the cell state are:

$$\mathrm{i_t} = \sigma(\mathrm{W_{ix}x_t} + \mathrm{W_{ih}h_{t-1}} + \mathrm{b_i}) \quad (1)$$

$$\mathrm{f_t} = \sigma(\mathrm{W_{fx}x_t} + \mathrm{W_{fh}h_{t-1}} + \mathrm{b_f}) \quad (2)$$

$$\mathrm{o_t} = \sigma(\mathrm{W_{ox}x_t} + \mathrm{W_{oh}h_{t-1}} + \mathrm{b_o}) \quad (3)$$

$$\widetilde{\mathrm{C_t}} = \tanh\,(\mathrm{W_{cx}x_t} + \mathrm{W_{ch}h_{t-1}} + \mathrm{b_c}) \quad (4)$$

$$\mathrm{C_t} = \mathrm{f_t} \odot \mathrm{C_{t-1}} + \mathrm{i_t} \odot \widetilde{\mathrm{C_t}} \quad (5)$$

where $\mathrm{x_t}$ denotes the input at time step t, $\mathrm{h_{t-1}}$ denotes the hidden state, W and b are the learned weight parameters and biases, σ is the sigmoid function, $\odot$ denotes element-wise multiplication, and $\widetilde{\mathrm{C_t}}$ represents the candidate cell state. Equation (1), (2) and (3) are for input, forget, and output gate namely. Equation (4) is for cell state update and equation (5) is for new cell state.

The equation for calculating the hidden state is:

$$\mathrm{h_t} = \mathrm{o_t} \odot \tanh\,(\mathrm{C_t}) \quad (6)$$

where $\odot$ denotes element-wise multiplication.

In addition, there is also a method for LSTM called Backpropagation Through Time. In order to train an LSTM model, a loss function must be minimized with regard to the inputs of the model. This is done using the backpropagation through time (BPTT) algorithm, which is a type of backpropagation that considers the sequential nature of the LSTM model. The equations for BPTT are:

$$\delta_{\mathrm{o,t}} = \frac{\partial \mathrm{L}}{\partial \mathrm{o_t}} \odot \sigma^{'}\ (\mathrm{W_{ox}x_t} + \mathrm{W_{oh}h_{t-1}} + \mathrm{b_o}) \quad (7)$$

$$\delta_{\mathrm{C,t}} = \delta_{\mathrm{h,t}} \odot \mathrm{o_t} \odot (1 - \tanh^2(\mathrm{C_t})) \quad (8)$$

$$\delta_{\mathrm{h,t}} = \frac{\partial \mathrm{L}}{\partial \mathrm{h_t}} + \delta_{\mathrm{o,t}}\mathrm{W_{oh}} + \delta_{\mathrm{i,t}}\mathrm{W_{ih}} + \delta_{\mathrm{f,t}}\mathrm{W_{fh}} \quad (9)$$

where L is the loss function, $\delta_{\mathrm{o,t}}$ is the error in the output gate, $\delta_{\mathrm{C,t}}$ is the error in the cell state, $\delta_{\mathrm{h,t}}$ is the error in the hidden state, $\sigma^{'}$ is the derivative of the sigmoid function, and $\delta_{\mathrm{i,t}}$ and $\delta_{\mathrm{f,t}}$ are the errors in the input and forget gates, respectively. Equation (7), (8) and (9) are for output error, cell state error and hidden state error respectively.

LSTM has several advantages for stock price prediction. First of all, because LSTM models have long-term memories, they can retain patterns and trends in time series data across extended stretches of time. This is especially helpful for predicting stock prices because historical trends often foretell future ones. Second, LSTM models have the ability to simulate non-linear correlations between input and output, which can be advantageous for stock price prediction in situations where there may be intricate interactions between several inputs. Third, LSTM models have the ability to be taught to foresee several steps in advance, which can be helpful for forecasting stock prices.

In summary, LSTM is a powerful neural network architecture that can effectively handle the vanishing gradient problem in RNNs.

2.4. Principle of CNN

CNN could be employed to predict stock prices [9,10]. It is designed to recognize patterns in visual data by using convolutional layers that learn spatial features from the input image. Several layers, including the input, convolutional, pooling, and fully connected layers, make up the fundamental structure of CNN. The convolutional layer is where a collection of filters is applied to achieve spatial information from the input data after the original data was sent to the input layer. The fully connected one is leveraged for classification, and the minimization of the dimensionality of the output was implemented by the pooling layer.

The operating principle of CNN is built upon the feature extraction and classification. For predicting stock prices, time series data is first converted into a 2D image where time and price is the x- and y-axis respectively. After that, a succession of convolutional and pooling layers is applied to this image in order to extract spatial characteristics that represent potential changes in stock price. The characteristics gathered are then sent to one or more fully connected layers for classification or regression. Each filter in the convolutional layer extracts a particular characteristic from the input data by applying it to the data. The filter moves over the input image, conducting dot products between the filter and the matching input image pixel values at each place. Non-linearity is then introduced into the output by a non-linear activation function when the result of this dot product is added up to generate a single output value. By performing a pooling operation, such as max pooling or average pooling, on a tiny portion of the output, the pooling operation is leveraged for lowering the dimensionality of the convolutional output. This reduces the size of the output and prompts the model more robust to small variations in the input dataset. The fully connected layer maps the pooling output to class labels. This is done by the dot product between the output from the pooling layer and a weight matrix, and then applying a softmax function to the result. The predicted output is determined by taking the class with the highest probability out of the available output classes generated by the softmax function.

The equation for the convolution operation is:

$$y_{i,j} = \sigma(\sum_{k,l} x_{i+k,j+l} w_{k,l} + b) \tag{10}$$

where x denotes the data, w denotes the weights of filter, b denotes bias term, σ denotes the activation function, and y is the feature map that output from the model. The parameters k and l represent the spatial dimensions of the filter.

The equation for the max pooling operation is:

$$y_{i,j} = \max_{k,l} x_{i+k,j+l} \tag{11}$$

where x and y denote input and output feature maps respectively.

The equation for Softmax function is:

$$\mathrm{y_k} = \frac{\mathrm{e^{z_k}}}{\sum_{\mathrm{j=1}}^{\mathrm{K}} \mathrm{e^{z_j}}} \tag{12}$$

where $\mathrm{z_k}$ is the output score of the last layer for class k, K denotes the number of classes overall, and $\mathrm{y_k}$ represents the probability of the input belonging to class k.

The equation for the cross-entropy loss function is:

$$\mathrm{L(y,\hat{y})} = -\sum_{\mathrm{k=1}}^{\mathrm{K}} \mathrm{y_k} \log \mathrm{\hat{y}_k} \tag{13}$$

where y is the true label distribution, $\hat{y}$ is the predicted probability distribution, K is the entire number of classes, $\mathrm{y_k}$ represents the annotated probability of class k, and $\mathrm{\hat{y}_k}$ denotes the predicted one.

CNNs also have a number of benefits for predicting stock prices. Firstly, CNNs have the ability to learn features automatically from the input data, which can eliminate the requirement for manual feature engineering. When predicting stock prices, where the data may be complicated and multidimensional, this is especially helpful. Second, CNNs can manage variable-length sequences, which is crucial for stock price prediction because time series data might have a range of lengths. Thirdly, CNNs may be taught effectively on GPUs, which can shorten the training process. This is especially helpful for stock price forecasting, where training may call for a sizable amount of data.

2.5. Evaluation metrics

In machine learning and statistics, two often used metrics for assessing the precision of prediction models are Root Mean Square Error (RMSE) and Mean Absolute Error (MAE). These two indexes are used to validate the discrepancy between a variable's predicted and actual values.

The mean value of the squared discrepancies between the expected and real values is calculated using the widely used statistic known as the root mean square error, or RMSE. Since it increases the impact of large errors, this statistic is beneficial in situations where greater errors are more important. The equation for RMSE is:

$$\mathrm{RMSE} = \sqrt{\frac{1}{\mathrm{n}} \sum_{\mathrm{i=1}}^{\mathrm{n}} (\mathrm{y_i} - \widehat{y_\imath})^2} \tag{14}$$

where $\mathrm{y_i}$ represents the ground truth, $\widehat{y_\imath}$ denotes the predicted one, while n represents the number of observations overall.

Nevertheless, MAE determines the mean value of the absolute disparities between the expected and real values. It is useful in situations where all errors are equally essential. The equation for MAE is:

$$\mathrm{MAE} = \frac{1}{\mathrm{n}} \sum_{\mathrm{i=1}}^{\mathrm{n}} |\mathrm{y_i} - \widehat{y_\imath}| \tag{15}$$

where $\mathrm{y_i}$, $\widehat{y_\imath}$, and n are same as that for RMSE.

In summary, RMSE and MAE are two widely used metrics for evaluating the accuracy of predictive models. While RMSE amplifies the impact of larger errors, MAE treats all errors equally. The decision between these two metrics is based on the particular requirements of the current challenge.

3. Result

3.1. Visualization result of improved LSTM

The prediction values were saved to a csv, and this paper compared it with the dataset.csv that given by Kaggle. From Figure 2 and Figure 3, it could be seen that the result of LSTM model can successfully predict the stock price.

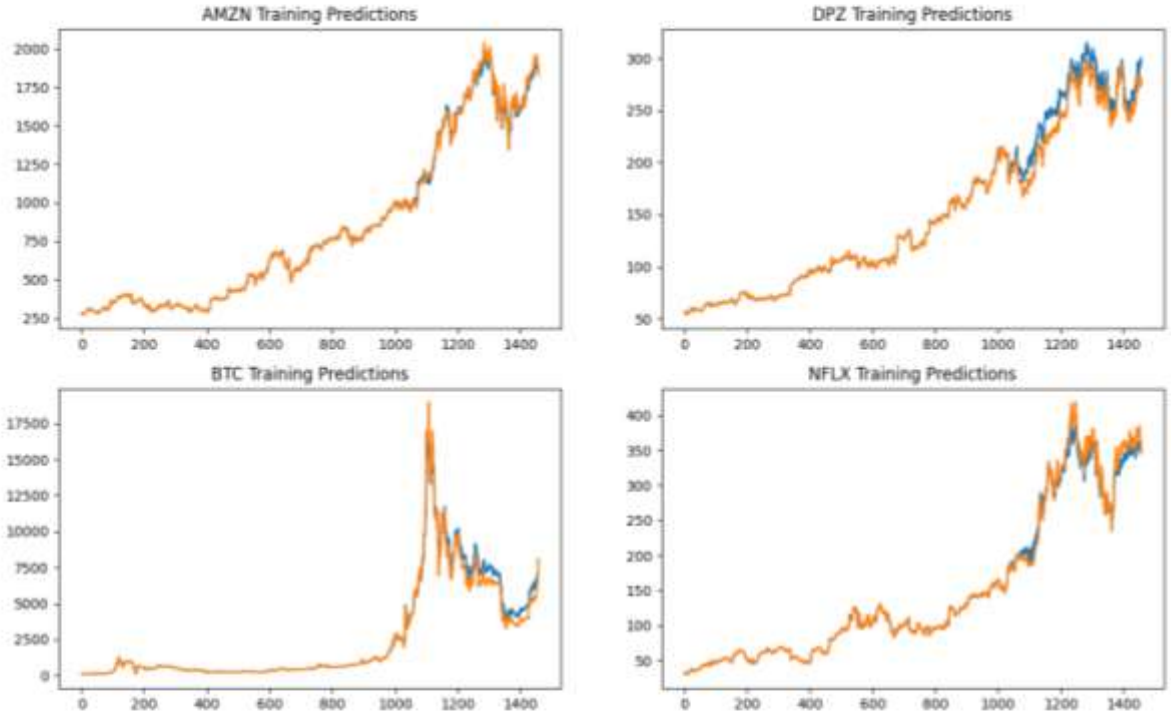

Figure 2. Visualization result of LSTM, with train_size=0.7, trained 100 times, and time_steps=3.

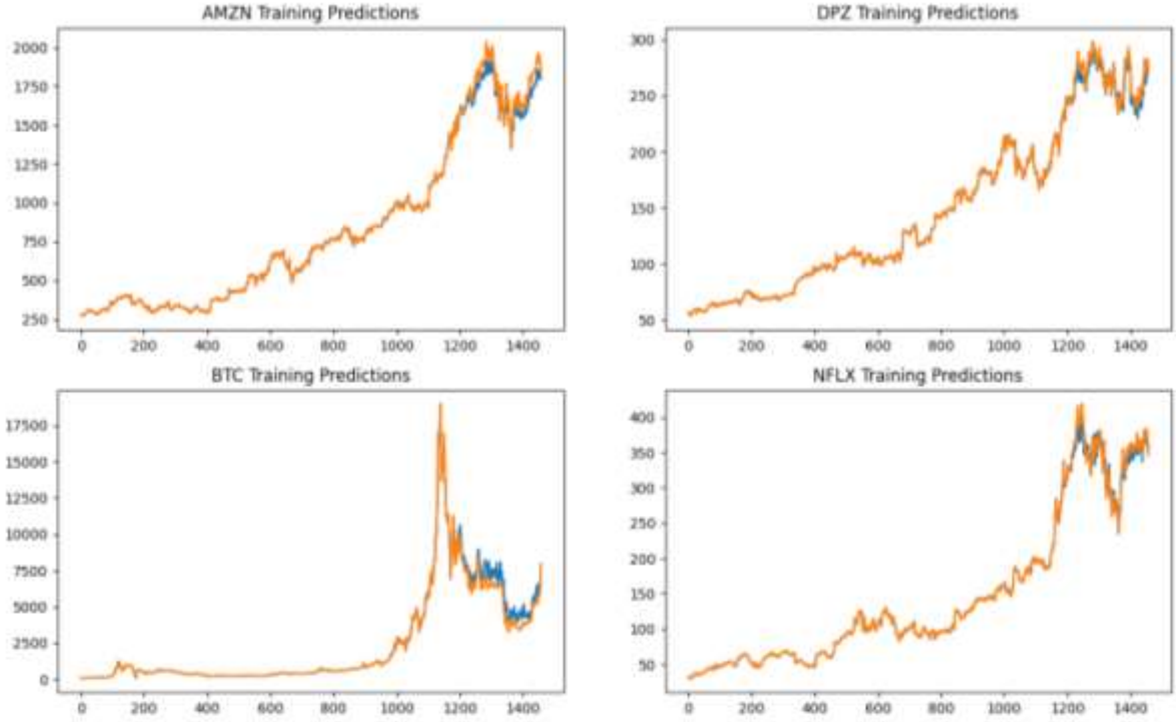

Figure 3. Visualization result of LSTM, train_size=0.8, trained 300 times, and time_steps=3.

It is clear that when the train_size and training times were increased, the performance of the model could be more accurate.

3.2. Visualization result of CNN

The prediction values and actual values were all saved to a csv file at the same time, and this latter compared them. It could be seen that the result of CNN model can accurately predict the stock price with a small error.

As the following visualizations illustrated in Figure 4, the effect of CNN is better than LSTM in this project.

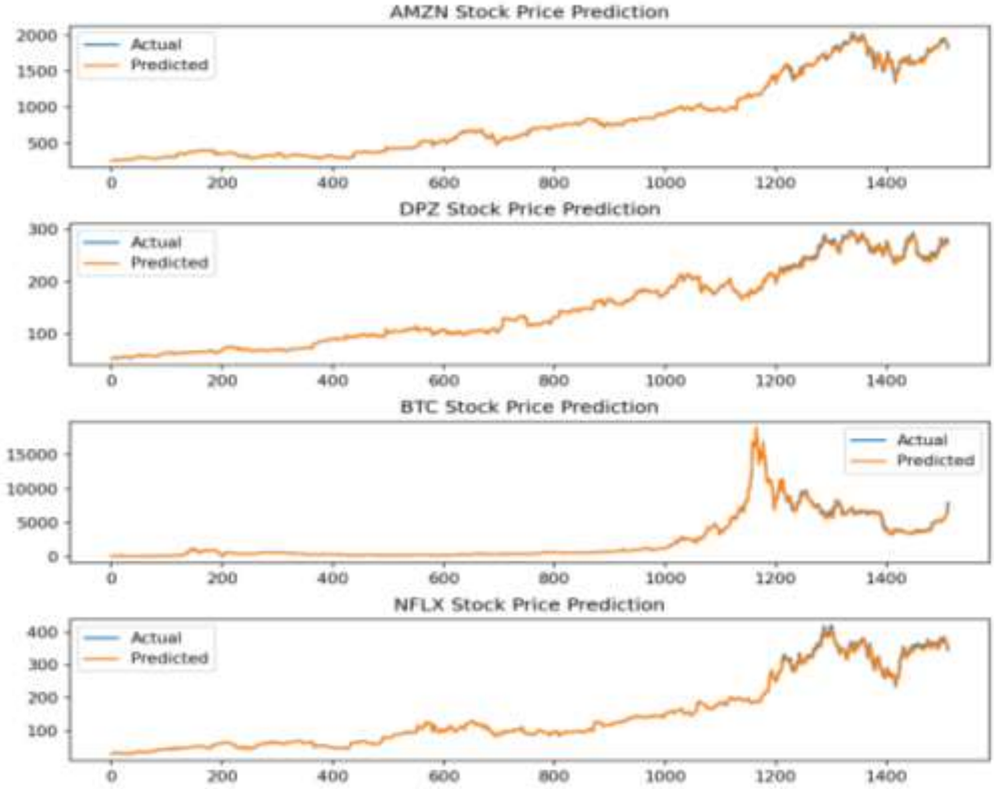

Figure 4. Visualization result of CNN, with train_size=0.8, trained 300 times, and look_back=3.

3.3. Comparison of CNN and LSTM

For LSTM, it was thought to be good and successful if meeting the overall hypothesis of the research, which means that the trend of the predicted price could nearly fit the actual value. However, as for CNN, a higher level of accuracy of prediction than LSTM model was expected.

In this project, the same conditions were set in implement the codes of the two models. Firstly, measures of adding labels, data pre-processing, inversing standardize and setting dataset with step_size are all the same. Secondly, both train_size accounted for 80 percent of the dataset given, which hold 1216 data and both models were trained 300 times. Thirdly, the soul of predicting is the same one and the two models have the same step_size (time_steps in LSTM is equal to the look_back in CNN).

Under these conditions, the visualization of the results displayed those performances of CNN was superior than that of LSTM in this project. As shown in Table 1, the same conclusion could also be made through computing the RMSE and MAE of the two models respectively.

Table 1. RMSE and MAE for LSTM and CNN.

	LSTM	CNN
RMSE	204.808427	174.523154
MAE	99.188623	78.778815

4. Discussion

4.1. Result explanations

There are several reasons for the phenomenon that CNN performed better than LSTM in this project. For instance, the series in the dataset given was not long enough to reflect the superiority of LSTM, which is better at modeling long-term dependencies. Besides, LSTM models may suffer from gradient disappearance, gradient explosion or over fitting, resulting in a result with a few more error than CNN and unstable training. Last but not least, CNN is better at capturing local features and can handle high-dimensional input.

There is also a point from another aspect of advantages and shortages of LSTM and CNN. Initially, LSTM can be slow to train. LSTM models require more computations and memory compared to other models, which can make them slower to train. However, CNN can be computationally efficient because it require fewer computations and memory compared to LSTM models, making them faster to train and evaluate.

4.2. Suggestions

Based on the results of these project, some suggestions could be made for the real market and investors to care about when using these models.

Firstly, using predictive models as a tool for decision-making. LSTM and CNN models can provide useful insights for stock price movements. However, they should be used as a tool for decision-making rather than as the sole basis for investment decisions. Investors should consider other factors such as company fundamentals, market trends, and risk management strategies. Secondly, be careful when relying on predictive models. While LSTM and CNN models can provide valuable insights, they should not be relied upon solely for investment decisions. These models are based on historical data and may not capture unexpected events or market shifts. Thirdly, stay informed about the latest developments. The stock market can be affected by external factors, including news, economic indicators, and political events. Keeping abreast of the latest developments and trends can help investors make informed decisions.

4.3. Limitation Analysis

Some limitations are still worth being considered for furthering this research in the future. Initially, the models are based solely on historical data and do not consider any objective factors or market fundamentals that may impact stock prices. This means that unexpected events or shifts in the market may not be captured by the models. Another limitation is that the models assume that past stock price trends will continue in the future. This may not always be the case, and stock prices can be affected by a range of factors. In addition, the models may suffer from overfitting, where the models may perform well during training but not generalize well to new data. This can result in misleading predictions and investment decisions. Transaction costs is also a valuable factor that should be improved, such as brokerage fees or taxes, which can impact investment returns. Furthermore, the performance of the models may not be well during periods of high volatility or sudden market shifts, as these situations may not be well-represented in the historical data.

Generally, investors should consider other factors such as market trends, risk management strategies, and objective market fundamentals when making investment decisions.

5. Conclusion

In conclusion, what was discussed in this paper is really a hot topic in the real life. There would be more thinking than thoughts mentioned above, including some other models, some optimizing methods and even consideration of objective factors like investor sentiment. It is valuable to further the research in the future to make the models more practicable. As for the results about prediction of stock prices, both LSTM and CNN model can be used in stock price prediction to gain returns as they performed well as the assumptions ahead of the project. Apart from that, this paper also compared the advantages and shortages about the two models so that investors could take it as a reference.

In terms of the prospects for the future research, it may set up with two aspects: one is testing more other models, and another is how to optimize the performance of the models. Here are plans for the two areas. Hybrid CNN-LSTM model would be the first one to compared with the results in this paper, and then may be ARIMA, Bayesian and SVM models. For optimization, several measures and objective factors would be considered, such as adding convolutional layers, changing step-size, other ways of label adding, and consideration of the investor's sentiment and marketing analysis. In that case, a more complete consequence of the stock price prediction topic would be generated.

References

[1] Vijh, M., Chandola, D., Tikkiwal, V. A., & Kumar, A. (2020). Stock closing price prediction using machine learning techniques. Procedia computer science, 167, 599-606.

[2] Masoud, N. M. (2013). The impact of stock market performance upon economic growth. International Journal of Economics and Financial Issues, 3(4), 788-798.

[3] Murkute, A., & Sarode, T. (2015). Forecasting market price of stock using artificial neural network. International Journal of Computer Applications, 124(12), 11-15.

[4] Li, L., Wu, Y., Ou, Y., Li, Q., Zhou, Y., & Chen, D. (2017). Research on machine learning algorithms and feature extraction for time series. In 2017 IEEE 28th annual international symposium on personal, indoor, and mobile radio communications (PIMRC), 1-5.

[5] Mehtab, S., & Sen, J. (2020). Stock price prediction using convolutional neural networks on a multivariate timeseries. arXiv preprint arXiv:2001.09769.

[6] Liang, X., Ge, Z., Sun, L., He, M., & Chen, H. (2019). LSTM with wavelet transform based data preprocessing for stock price prediction. Mathematical Problems in Engineering, 1-8.

[7] Smagulova, K., & James, A. P. (2019). A survey on LSTM memristive neural network architectures and applications. The European Physical Journal Special Topics, 228(10), 2313-2324.

[8] Yu, Y., Si, X., Hu, C., & Zhang, J. (2019). A review of recurrent neural networks: LSTM cells and network architectures. Neural computation, 31(7), 1235-1270.

[9] Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., et al. (2018). Recent advances in convolutional neural networks. Pattern recognition, 77, 354-377.

[10] Ajit, A., Acharya, K., & Samanta, A. (2020). A review of convolutional neural networks. In 2020 international conference on emerging trends in information technology and engineering (ic-ETITE), 1-5.