Journal of Industrial Information Technology and Application *ISSN(Online): 2586-0852*
Vol.8 No. 4

Feasibility of the Neural Network Model for Stock Price Prediction

Joshua Buhr[1)], Even Nybo[1)], Nicklaus Campanella[1)], and Donghwoon Kwon[1,*)]

[1)] Dept. of Computer Science and Engineering, North Central College, Naperville, IL 60540, United States of America

Abstract. This study explores the feasibility of utilizing a neural network model to predict stock prices. The neural network model employed is a 1-dimensional convolutional layer-based model called 1D Convolutional Neural Network (1D-CNN). Historical stock data for Tesla and Disney, spanning three years from January 1st, 2018, to December 31st, 2020, is collected using the Yahoo Finance Application Programming Interface (API). The collected stock data establishes three cases for evaluating model performance. Model training is based on window sizes of 15, 30, and 60 days and random seeds range of 1 to 1,000 with 2,000 epochs and a learning rate of 1e-3. The experimental results with three cases reveal competitive performance for stock price prediction.

Keywords; stock prices; financial forecasting; deep learning; time series analysis; stock prediction

1. Introduction

Predicting stock prices accurately is a significant challenge in finance due to the high volatility and unpredictable nature of financial markets [1]. In particular, stock prices are greatly influenced by various factors such as economic events, corporate news, and global political and economic instability. Predicting the impact of these factors on stock prices is a crucial area of research in finance, and many individuals, stock market analysts, or researchers have attempted to predict accurate stock prices in conjunction with statistical, econometric, or neural network models [1]. Among them, this research

[*] Corresponding author: dkwon@noctrl.edu

Received: Oct. 31. 2024; Accepted: Nov. 30, 2024; Published: Dec. 31.2024

specifically focuses on investigating the feasibility of a neural network model for predicting stock prices and sharing observations from initial experimental results with the Convolutional Neural Network (CNN) model and the stock data of Tesla and Disney. Hence, the following three research questions are established:

- Would the proposed CNN model work well for predicting stock prices?
- Would the employed neural network model show consistent performance with different datasets of stock prices?
- If the employed neural network does not perform satisfactorily, what strategies could improve model performance?

To find answers to the research questions established above, we conducted a literature review and extensive experiments and observed a few interesting findings. One finding regarding a neural network model is that while the Recurrent Neural Network (RNN) model in conjunction with Long-Short Term Memory (LSTM) and Gated Recurrent Units (GRU) is well-known for predicting time-series data well [2-4], the CNN model also performs well at predicting stock prices based on time-series data.

This paper is organized as follows: Section 2 provides a brief overview of the literature review related to neural network models that have been proposed for predicting stock prices; Section 3 describes our research methodology related to dataset preprocessing and the CNN model structure; Section 4 discusses experimental results with three cases; and finally, Section 5 describes the conclusions and future studies.

2. Related Work

This section briefly describes an overview of the literature review on neural networks and performance metrics related to stock price prediction.

Research conducted by [3] examined the use of Deep Learning (DL) models and performance metrics for predicting stock prices. The author of the paper stated that numerous research papers have widely utilized Support Vector Machine (SVM) and neural network models, such as LSTM and GRU-based RNN, as well as hybrid neural network models, for predicting stock price trends and stock market volatility. Additionally, the author introduced four performance metrics: Mean Absolute Percentage Deviation (MAPD), Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Square Deviation (RMSD).

In [5], the authors utilized four Machine Learning (ML) models, i.e., Decision Tree (DT), Support Vector Regression (SVR), Random Forest (RF), and Artificial Neural

Network (ANN), for stock market prediction. The dataset used in this research was a 5-year dataset of American Airlines stock, splitting it into 70% for training and 30% for testing. Specifically, the training dataset covered the period from February 8th, 2016, to August 6th, 2016, and the testing dataset covered August 10th, 2016, to February 6th, 2018. A total of seven features, e.g., date, open, high, low, close, volume, and name, were taken, and the Min-Max scaler and standard scaler were used for dataset normalization. The Mean Absolute Performance Error (MAPE) was adopted to evaluate the models' performance, and the authors reported that the RF model outperformed the others with a MAPE value of 0.36.

The authors in [6] employed Multi-Layer Perceptron (MLP), CNN, and LSTM-based RNN models and conducted a comparison analysis of models' performance with the S&P500 historical time series dataset to forecast stock market price movements. In this research, the CNN model showed better performance than the other two models, with the lowest MSE of 0.2491. Note that the CNN model architecture utilized in this research is as follows:

- Two 1D convolutional layers, two 1D max-pooling layers, and two Fully Connected (FC) layers.
- Hyperparameters: 64 filters, Rectified Linear Unit (ReLU) activation function, and Adaptive Moment Estimation (Adam) optimizer.

Based on the literature review above, we decided to utilize the CNN model with MSE performance metrics but build a different CNN model architecture with different hyperparameters and factors.

3. Methodology

A. Dataset Preprocessing

The first step in this research is dataset preprocessing, which begins with obtaining the dataset. Stock data for two companies, Tesla and Disney, spanning 1,095 days over three years from January 1, 2018, to December 31, 2020, is collected using the Yahoo API. The adopted features are open, high, low, and volume for input and close for output, and the entire dataset is then split into training and testing sets based on an 80/20 rule, with 876 days used for training and 219 days for testing. A sliding window algorithm is employed for feature extraction of the dataset, and window sizes of 15, 30, and 60 days are initially considered to find the most optimal window size. With the preprocessed dataset, the following three possible cases are established:

- CASE I: Train and test the model with Tesla.

- CASE II: Train and test the model with Disney.
- CASE III: Train the model with Tesla and test it with Disney.

Note that the case of training the model with Disney and testing it with Tesla is intentionally skipped because we are interested in observing how volatile stocks predict less volatile stocks.

B. *CNN Model Structure*

CNN model is a well-known DL model used for image classification tasks, giving it powerful classification capability when applied to an image dataset [7]. The main component of the CNN architecture is a convolutional layer, and there are three types available: one-dimensional convolutional layer (Conv1D), two-dimensional convolutional layer (Conv2D), and three-dimensional convolutional layer (Conv3D). Note that Conv1D is for time series data, Conv2D is for images, and Conv3d is for videos [8]. Since the dataset in this research is historical stock data (time-series data), the Conv1D layer needs to be used to build the CNN structure. The Conv1D layer fundamentally utilizes a set of filters, also known as kernels, essential in producing a feature map by applying a convolutional operation to the input data [9].

The second important component of the CNN model structure is the activation function. Many activation functions, e.g., linear, sigmoid, Hyperbolic Tangent (Tanh), and Rectified Linear Unit (ReLU), have been proposed [10]. Among them, the ReLU activation is selected because it shows faster and more efficient training than other activation functions such as sigmoid or Tanh [7][11]. Furthermore, the ReLU activation function is suitable for solving problems with the sigmoid and tanh activation functions [12-13].

Fundamentally, the ReLU activation function is denoted as:

$$f(x) = \begin{cases} 0 \; if \; x \leq 0 \\ x \; if \; x > 0 \end{cases} \tag{1}$$

where *f* is the ReLU activation function, and *x* represents the input. Note that when the input *x* is less than or equal to 0, the output becomes 0. On the other hand, when the input *x* is greater than 0, the output is equal to the input *x*. The ReLU activation is crucial in stock prediction because it allows for better modeling of nonlinearities in the data, such as sudden price changes due to market news.

The next key component is the pooling layer, and two widely used pooling methods are average pooling and max pooling. Both methods reduce dimensionality by downsampling the feature representation. However, the difference between these two is whether to take the average or the maximum value in the pooling region [14]. One benefit of using the max pooling layer is improving computational efficiency [15]. For this reason, max pooling is employed in this research instead of average pooling. Note that the pooling layer can help extract more abstract features for stock prediction by focusing on the most prominent patterns in the feature maps and ignoring less relevant details.

The Fully Connected (FC) layers further process the features extracted by the convolutional and pooling layers to make final predictions, and they map the extracted features to the desired output size. The first FC layer (FC1) reduces the dimensionality of the flattened feature map, and the second FC layer (FC2) outputs the final prediction. These layers are crucial for integrating the learned features into a coherent prediction, such as predicting future stock prices. Moreover, they can learn complex relationships between the features and the target variable.

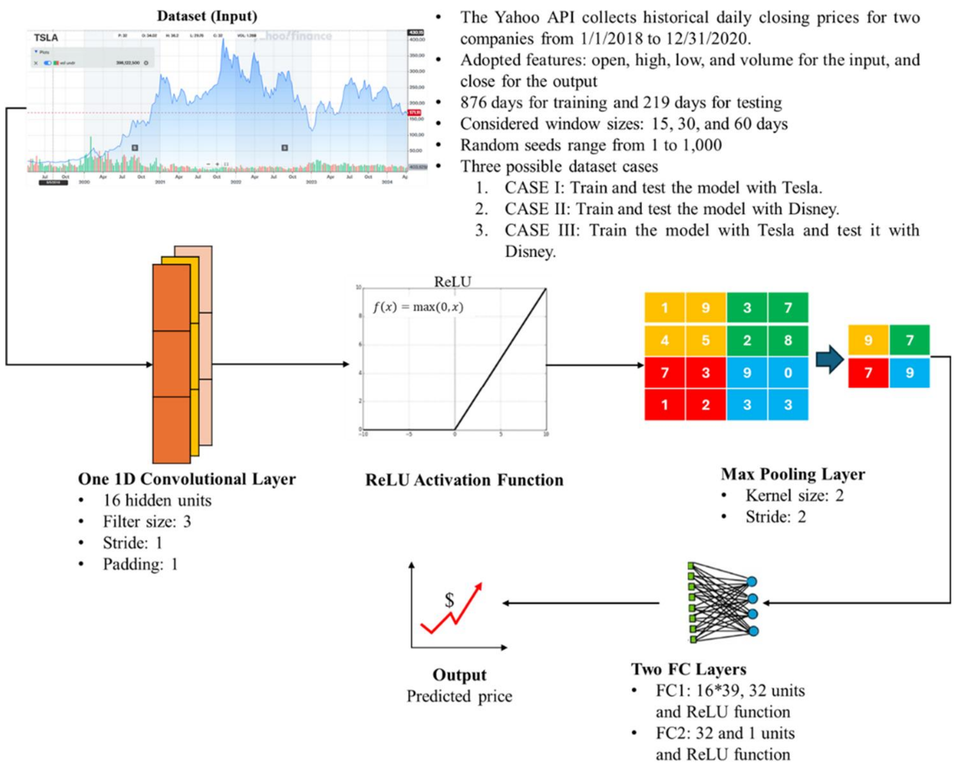
When training the model, a loss function, along with an optimizer, plays a significant role in optimizing the model's parameters to minimize the difference between predictions and actual output [16]. Since the CNN model used in this research is regression-based supervised learning, which predicts continuous output values (stock prices) based on multiple features, various loss functions such as Mean Squared Error (MSE), Mean Absolute Error (MAE), Huber Loss, Log-Cosh Loss, and others could be utilized [16]. Among them, the MSE loss function is adopted due to its simplicity characteristic and denoted as [16]:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2 \tag{2}$$

where N is the number of data, and Y_i and $\hat{Y}_i$ are the actual (observed) value and predicted value of the i^{th} data, respectively. The background of MSE is that it penalizes larger errors more heavily than smaller ones because the errors are squared. This is useful in stock prediction because it gives more significance to correcting predictions with larger deviations from the actual values while minimizing significant prediction errors. MSE provides a smooth and convex loss landscape, making it easier to find a minimum. It measures the squared dollar amount by which the predictions differ from the actual prices, offering a more intuitive interpretation of the errors.

Lastly, regarding the optimizer, we consider several optimizers, e.g., Stochastic Gradient Descent (SGD), AdaGrad, RMSProp, Adaptive Moment Estimation (Adam), and others. With consideration of the pros and cons and popularity of each optimizer, the Adam optimizer is the most widely used one in the field of DL. Note that it combines the advantages of AdaGrad and RMSProp and is based on adaptive learning rates, which involves adjusting the learning rate for each parameter individually based on estimates of the first and second moments of the gradients [17]. This adaptive learning rate is beneficial in handling the noisy and nonstationary nature of stock market data. In addition, in models where certain features might be sparsely represented commonly in financial data, Adam performs well as it inherently implements an adaptive form of momentum, which helps navigate through sparse gradients. Combining the MSE loss function and the Adam optimizer in stock prediction models offers a balance of simplicity, adaptability, and efficiency. MSE provides a straightforward, convex error surface, while Adam optimizes this surface effectively by adapting to the data's peculiarities. This synergy can lead to more accurate and robust models for predicting stock prices.

In summary, the overall CNN structure is illustrated in Figure 1 below.



- The Yahoo API collects historical daily closing prices for two companies from 1/1/2018 to 12/31/2020.
- Adopted features: open, high, low, and volume for the input, and close for the output
- 876 days for training and 219 days for testing
- Considered window sizes: 15, 30, and 60 days
- Random seeds range from 1 to 1,000
- Three possible dataset cases
  1. CASE I: Train and test the model with Tesla.
  2. CASE II: Train and test the model with Disney.
  3. CASE III: Train the model with Tesla and test it with Disney.

One 1D Convolutional Layer
- 16 hidden units
- Filter size: 3
- Stride: 1
- Padding: 1

ReLU Activation Function

Max Pooling Layer
- Kernel size: 2
- Stride: 2

Output
Predicted price

Two FC Layers
- FC1: 16*39, 32 units and ReLU function
- FC2: 32 and 1 units and ReLU function

Figure 1. CNN Model Structure

4. Experiments

We utilized the Google Colab platform and the PyTorch library to build a CNN model to conduct experiments and assess the model's performance. We opted for three different window sizes (15, 30, and 60 days), as previously mentioned, and random seeds ranging from 1 to 1,000. The 1D CNN model was trained for 2,000 epochs and used a learning rate of 1e-3 with the MSE loss function and the Adam optimizer. Note that the experiments were conducted for each case, and the window size and random seed showing the best MSE loss value may differ for each case. Table 1 and Figure 2 below illustrate and summarize the average MSE values of model training and testing.

Table I. AVERAGE MSE OF MODEL PERFORMANCE

	Train MSE	Test MSE	Random Seed	Window Size
CASE I	0.000099	0.011060	640	15
CASE II	0.001093	0.003359	97	30
CASE III	0.000099	0.011060	640	15

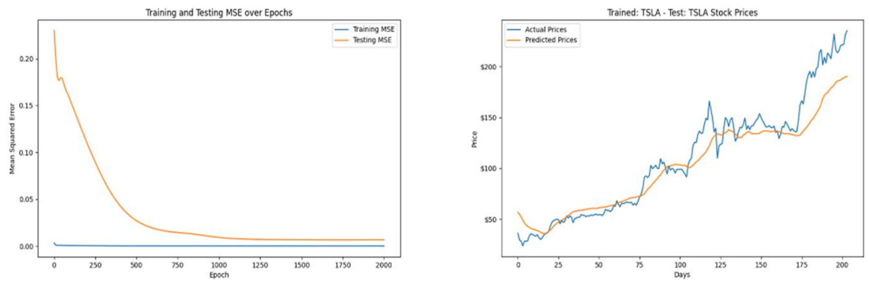


Figure 2. CASE I Model Performance

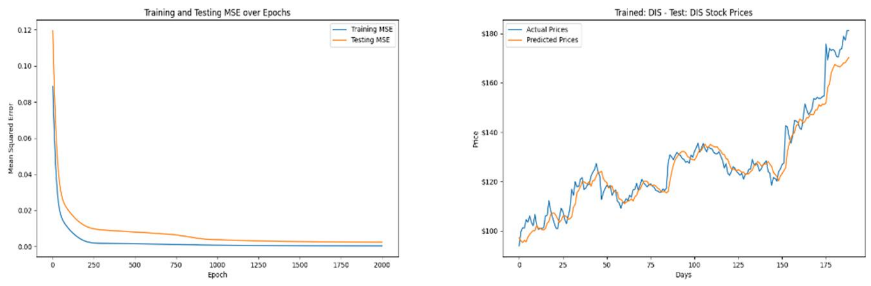


Figure 3. CASE II Model Performance

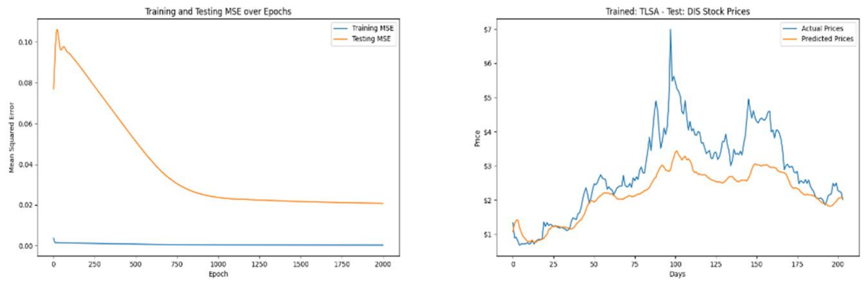


Figure 4. CASE III Model Performance

As shown in Table 1 and Figure 2 above, a window size of 15 days with a random seed of 640 showed the best performance for CASE I and III. On the other hand, a window size of 30 days with a random seed of 97 worked best for CASE II. From these experimental results, we observed that the optimal window size for predicting stock price trend lines varies depending on the stock price volatility. When the stock price volatility is significant, a small window size is more advantageous for predicting the trend line. Furthermore, our experimental results of CASE II and III indicated that when the stock price volatility was low, the CNN model predicted a trend line almost identical to the actual stock price. Even when trained and tested with two different stock prices, the trend line predicted by the CNN model closely matched the actual stock price trend line and showed very consistent model performance.

5. Conclusion and Future Work

This research presents a 1D CNN model that predicts the closing price of individual stocks based on historical data. The proposed CNN model is a feasible method for stock price prediction because it can process time series data and extract relevant features. The experimental results show that the proposed model can capture the general trend of stock prices well, depending on two main factors: window size and random seed. Additionally, the model shows consistent performance when even two different datasets are used for training and testing. However, further experiments are needed to optimize the model performance and evaluate its generalizability to other stocks and market conditions.

Future studies can explore the significance of qualitative factors in stock market analysis, such as the interpretation of news, earnings reports, market volatility, liquidity, and interest rates. One research plan is to develop hybrid neural networks or transformer models, such as CNN-transformer, LSTM-transformer, and RNN-transformer.

References

[1] Y. Song, Y. Zhou, and R. Han, “Neural networks for stock price prediction,” *arXiv preprint arXiv:1805.11317*, 2018.

[2] G. Yadav and R. Vasuja, “Analysis of Time Series Prediction using Recurrent Neural Networks,” International Journal of Computer Applications, Vol. 182, No. 48, pp. 34-40, 2019.

[3] X. Chen, “Stock Price Prediction Using Machine Learning Strategies,” International Conference on Company Management, Accounting and Marketing, Vol. 12, pp. 68-73, 2022.

[4] A. Moghar and M. Hamiche, “Stock Market Prediction Using LSTM Recurrent Neural Network,” Procedia Computer Science 170, pp. 1168-1173, 2020.

[5] J. Hota, S. Chakravarty, B. K. Paikaray, and H. Bhoyar, “Stock Market Prediction Using Machine Learning Techniques,” Workshop on Advances in Computation Intelligence, its Concepts and Applications (ACI 22), pp. 163-171, 2020.

[6] L. D. Persio, O. Honchar, “Artificial Neural Networks architectures for stock price prediction: comparisons and applications,” International journal of circuits, systems and signal processing, Vol. 10, pp. 410-143, 2016.

[7] F. Sultana, A. Sufian, P. Dutta, “Advancements in Image Classification using Convolutional Neural Network,” 2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN), IEEE, pp. 122-129, 2018.

[8] B. A. Krohling and R. A. Krohling, “1D Convolutional neural networks and machine learning algorithms for spectral data classification with a case study for Covid-19,” arXiv preprint arXiv:2301.10746, 2023.

[9] M. Saini, U. Satija, M. E. Upadhayay, “Light-Weight 1-D Convolutional Neural Network Architecture for Mental Task Identification and Classification Based on Single-Channel EEG,” arXiv preprint arXiv:2012.06782, 2020.

[10] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, “Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark,” Neurocomputing, Vol. 503, pp. 92-108, 2022.

[11] T. Szandała, “Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks,” Bio-inspired neurocomputing, pp. 203-224, 2021.

[12] Y. Bai, “RELU-Function and Derived Function Review,” SHS Web of Conferences, Vol. 144, 2022.

[13] K. He, X. Zhang, S. Ren, and J. Sun, “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification,” Proceedings of the IEEE international conference on computer vision, pp. 1026-1034, 2015.

[14] H. Gholamalinezhad and H. Khosravi, “Pooling Methods in Deep Neural Networks, a Review,” arXiv preprint arXiv:2009.07485, 2020.

[15] L. Zhao and Z. Zhang, “A improved pooling method for convolutional neural networks,” Scientific Reports 14.1, Article number. 1589, 2024.

[16] J. R. Terven, D. M. Cordova-Esparza, and A. Ramirez-Pedraza, "Loss functions and metrics in deep learning. A review," arXiv preprint arXiv:2307.02694, 2023.

[17] D. Soydaner, "A Comparison of Optimization Algorithms for Deep Learning," International Journal of Recognition and Artificial Intelligence, Vol. 34, No. 13, 2020.