Stocks Prediction Using RNN

1st 61247030S*
CSIE of National Taiwan Normal University
Taipei, Taiwan
61247030S@ntnu.edu.tw

***Abstract*—In predicting stock performance, neural networks offer a promising approach. My observations suggest that the closing price of a stock is correlated with its prices from the preceding days. Therefore, I utilized Recurrent Neural Networks (RNN) along with Fully Connected layers to forecast these prices. By back-testing on the 0050.TW dataset, I achieved favorable prediction outcomes. My code and pretrained model can be found on https://github.com/Luncheyea/stocks-prediction-using-rnn.git**

***Index Terms*—stocks prediction, Recurrent Neural Networks, deep learning**

I. Introduction

This paper presents a project for the Artificial Neural Network course offered by the Department of Computer Science and Information Engineering (CSIE) at National Taiwan Normal University. The primary goal is to employ neural networks to predict the closing prices of the 0050.TW stock index over the next five days based on historical data.

In stock prediction, various methodologies have been explored, such as Diffstock [1], Modeling stock price dynamics on the Ghana Stock Exchange [2], and Advanced Statistical Arbitrage with Reinforcement Learning [3], among others. These models have demonstrated impressive performance; however, their complexity poses a significant learning challenge for beginners in Artificial Neural Networks. Thus, I sought a method that is not only powerful but also accessible for novices, facilitating a more efficient learning process.

Considering that stock market forecasting is essentially a time-series analysis — where the price of a stock is influenced by its previous prices — I decided to utilize Recurrent Neural Networks (RNN) [4] and Fully Connected layers for my analysis. RNNs are particularly suited for this task due to their capability to remember input sequences, making them a fundamental architecture in Neural Network studies. To accommodate the output requirement (predicting the closing prices for the next 5 days), I incorporated a Fully Connected layer towards the end of my model.

The experiment results indicate that my model exhibits low loss during training, suggesting that it has achieved favorable prediction outcomes.

finally, I make the following contributions:

- **Simplified Yet Effective Architecture**: I have developed a straightforward but potent architecture for stock prediction. This approach not only demonstrates the power of simplicity in design but also ensures that beginners can understand and apply these concepts with ease.
- **Precision Indicated by Loss Metrics**: The loss metrics of my model serve as a strong indicator of its precision in predicting stock prices. These metrics suggest that the model can reliably forecast future prices, marking a significant achievement in stock market analysis.
- **Open-Source Contribution**: I have published my code on GitHub, making it accessible to peers and other interested parties. This move facilitates collaborative learning and enables others to explore, learn from, and possibly improve upon my work.

II. Methods

A. Dataset

The dataset consists of a sequence of daily closing prices. During each training epoch, a set of N consecutive closing prices is randomly selected from the dataset. This set is then divided into two parts: the first k prices are used for training, and the subsequent $N-k$ prices are reserved for testing. Formally, the process is described as follows: Given a dataset D containing sequences of closing prices, a subset S is selected such that $S \subset D$ and $|S| = N$. This subset S is randomly chosen to ensure variety in the training and testing phases. The division of S for training and testing purposes is defined by:

$$S_{\text{train}} = \{s_1, s_2, \ldots, s_k\} \tag{1}$$

$$S_{\text{test}} = \{s_{k+1}, s_{k+2}, \ldots, s_N\} \tag{2}$$

where s_i represents the closing price on day i. The size of the training and testing sets is such that $|S_{\text{train}}| + |S_{\text{test}}| = N$. This methodological approach facilitates a dynamic learning environment, aiming to enhance the model's generalization capabilities.

B. Model Architecture

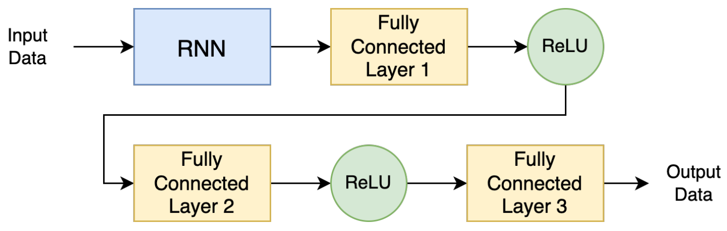

Fig. 1. The training and testing pipline of the model.

Given the correlation between a stock's closing prices and those of preceding days, it is pivotal to employ a neural network capable of memorizing sequential input data and identifying relationships within it. To simplify the architecture and facilitate learning, I have chosen Recurrent Neural Networks (RNN) as the primary structure for my model. This decision leverages the RNN's inherent ability to process sequences of data effectively.

Furthermore, to ensure stable output and align with the required number of output data points, a Fully Connected (FC) layer is incorporated at the end of the model. Between each layer, a Rectified Linear Unit (ReLU) activation function is applied to introduce non-linearity and aid in the learning process.

During training, the subset S_{train} serves as the input data for the model pipeline (Fig. 1). The model outputs O, where $|O| = N - k$, which are then compared against S_{test} to compute the loss function. For this purpose, the Mean Squared Error (MSE) [5] loss is utilized:

$$\text{MSE Loss} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{3}$$

where Y_i represents the actual values from S_{test} and $\hat{Y}_i$ denotes the predicted values from O.

In the testing phase, the model's weights are fixed to evaluate its performance on unseen data, ensuring an unbiased assessment of its predictive capabilities.

III. Experiment

A. Environment

My operating system is Ubuntu 20.04 LTS, which provides a stable and efficient platform for developing and running machine learning projects. For project management and code editing, Visual Studio Code (VSCode) is used as the primary editor due to its extensive support for development tools and extensions.

Considering the scale of the dataset and the model, the hardware requirements are not heavy. Therefore, a single GPU, specifically a GTX 1080 Ti, suffices for both training and testing phases. This setup demonstrates that effective machine learning projects can be conducted without extensive hardware resources, provided the problem domain is adequately understood and optimized.

The software environment is built on Python version 3.10.14, which offers a rich ecosystem of libraries and frameworks for data science and machine learning projects. There are 3 key packages used in this project include:

- **PyTorch (version 2.2.1)**: Utilized for constructing, training, and evaluating the neural network model. PyTorch's flexibility and intuitive design make it an excellent choice for implementing RNN architectures.
- **Matplotlib (version 3.8.3)**: Employed to visualize the training and testing data, as well as the loss over epochs. These visualizations are crucial for understanding the model's learning progress and making necessary adjustments.
- **Pandas (version 2.2.1)**: Used for parsing the dataset from CSV files. Pandas provide an efficient and user-friendly interface for data manipulation and analysis, making it indispensable for pre-processing tasks.

B. Dataset

For the course project in Artificial Neural Networks, the 0050.TW [6] stock index is asked to use as primary dataset. This dataset comprises daily closing prices of the 0050.TW index, spanning from May 1, 2009, to March 22, 2024, covering each trading day. Overall, the dataset contains 3,719 data points, providing a comprehensive view of the index's performance over nearly 15 years.

In the experimental setup, N is set to 30, and k is set to 25. This configuration means that for each epoch, the training set (S_{train}) comprises 25 data points, and the testing set (S_{test}) consists of the subsequent 5 data points. The choice of N and k values is strategic to simulate a realistic forecasting scenario where the model predicts the next week's closing prices based on the past month's data.

The visualization of data for each epoch, illustrating the division between training and testing sets, is presented in Fig. 2. This visual representation aids in understanding the temporal division of data and the model's performance across different epochs.

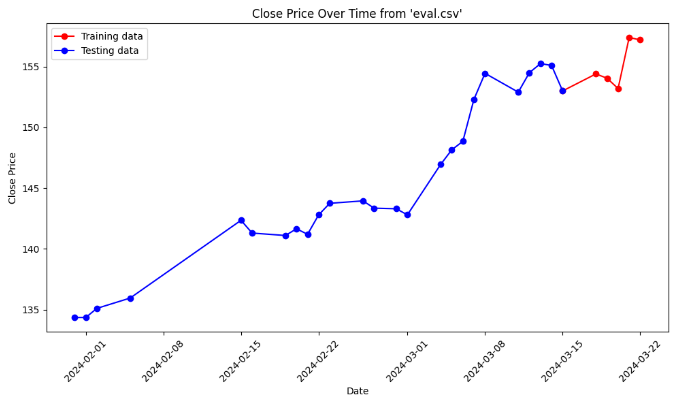

Fig. 2. Closing prices of 0050.TW in an epoch. The blue line is training data; The red line is testing data.

C. Model Setting

As depicted in Fig. 1, the model architecture is designed around a Recurrent Neural Network (RNN) core, characterized by specific dimensional settings to optimize for the task of stock price prediction. The RNN is configured with an input size of 1, reflecting the single-dimensional nature of the closing price data. It comprises 20 hidden units to capture the complexities within the time series data, and is structured with a single layer to maintain a balance between model simplicity and its ability to learn temporal dependencies.

Following the RNN, the architecture includes three Fully Connected (FC) layers. The first two FC layers (FC1 and FC2) are designed with both input and output sizes of 20, ensuring a consistent flow of information while allowing for nonlinear transformations. To achieve the final output, which predicts the closing prices for the next 5 days, the third FC layer (FC3) transitions from 20 input units to 5 output units.

The training process is fine-tuned with a batch size of 1, indicating that the model updates its weights after learning from each individual data point. This approach, while computationally intensive, allows for a more detailed learning process. The model is trained over 150 epochs to ensure adequate learning without overfitting, with a learning rate set at 0.0001 to provide a steady yet effective pace of optimization.

The progression of the model's learning is visually represented through a line chart of the loss metrics across epochs, as shown in Fig. 3. This visualization highlights the model's improvement and stabilization over time, offering insights into the effectiveness of the chosen configuration and training.

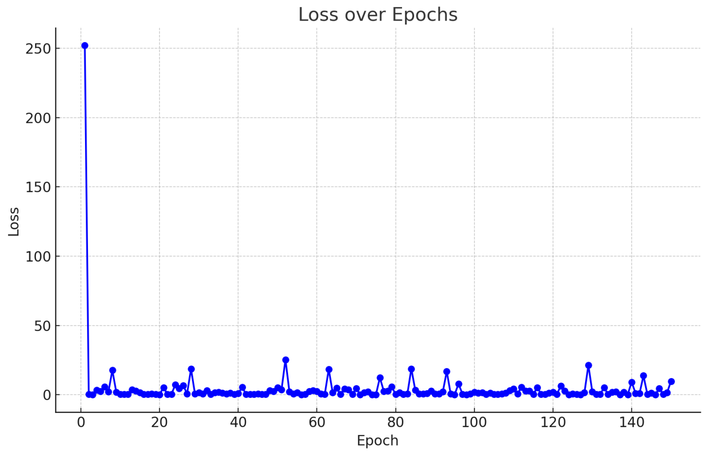

Fig. 3. The training loss over epochs.

D. Result

TABLE I
PREDICTED CLOSING PRICES

Closing prices from February 16, 2024, to March 22, 2024

141.300003	141.100006	141.649994	141.199997	142.800003
143.75	143.949997	143.350006	143.300003	142.800003
146.949997	148.149994	148.850006	152.300003	154.449997
152.899994	154.5	155.25	155.100006	153
154.399994	154.050003	153.199997	157.399994	157.199997

Predicted closing prices from March 25, 2024, to March 29, 2024

155.95952	156.02403	156.13509	156.20909	156.10287

Table. I presents the predicted closing prices from March 25, 2024, to March 29, 2024, as forecasted by my model.

IV. Conclusion

In this project, I have acquired valuable knowledge and skills in several key areas of machine learning and software development. The construction and application of a Recurrent Neural Network (RNN) paired with Fully Connected (FC) layers enabled the prediction of future closing prices for the 0050.TW stock index. This hands-on experience has not only strengthened my understanding of neural network architectures but also demonstrated the practical implications of such models in financial forecasting.

Moreover, I use LaTeX for academic writing, enhancing my ability to present research findings in a structured and formal manner. Publishing my work on GitHub further contributed to my learning, offering insights into version control and the importance of open-source contributions to the scientific community.

The low training loss observed during the model's development suggests that the predictions for the closing prices from March 25, 2024, to March 29, 2024, will be highly accurate.

Overall, this project has been immensely educational, providing a comprehensive introduction to neural network design, the application of machine learning techniques in finance, and the professional presentation of research work. The successful prediction results further motivate continued exploration and development within this field.

References

[1] Daiya, Divyanshu et al. "Diffstock: Probabilistic Relational Stock Market Predictions Using Diffusion Models." ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2024).
[2] Quayesam, Dennis Lartey et al. "Modeling stock price dynamics on the Ghana Stock Exchange: A Geometric Brownian Motion approach." (2024).arXiv:2403.13192
[3] Ning, Boming and Kiseop Lee. "Advanced Statistical Arbitrage with Reinforcement Learning." (2024). arXiv:2403.12180
[4] https://en.wikipedia.org/wiki/Recurrent_neural_network
[5] https://en.wikipedia.org/wiki/Mean_squared_error
[6] https://www.twse.com.tw/zh/ETFortune/etfInfo/0050