Recurrent Neural Network (RNN) for Google Stock Price Prediction

Adithya Sundararajan Iyer
Department of Computer Science
University of Texas at Dallas
Richardson, TX, USA
asi200000@utdallas.edu

Siddhant Suresh Medar
Department of Computer Science
University of Texas at Dallas
Richardson, TX, USA
ssm200002@utdallas.edu

Shreyans Patel
Department of Computer Science
University of Texas at Dallas
Richardson, TX, USA
ssp210009@utdallas.edu

Pranitha Sreethar
Department of Computer Science
University of Texas at Dallas
Richardson, TX, USA
pxs200095@utdallas.edu

***Abstract*—This paper discusses our approach at building a Recurrent Neural Network (RNN) model from scratch that implements a time-series stock prediction based on Google's stock price data of the past one year. We use the coefficient of determination, R-squared, to determine the accuracy of our model since it is a reliable measure of the proportion of variation that is explained by regression models.**

Keywords—RNN, time-series, stock prediction, forecasting, deep learning

I. Introduction

Prediction of the stock market has been in the study for the past several years. With the advent of Machine Learning, the complexity of stock market prediction has significantly been reduced. Applying different machine learning techniques has helped both stock buyers and sellers alike to carefully consider their choices and earn profit.

Our aim here is to accurately forecast the stock value based on its past performance.This forecasting will be time-series based and can also be used for applications such as monitoring sales, predicting the weather, recognizing speech, audio, video, and much more.

The Recurrent Neural Network (RNN) algorithm is a state-of-the-art algorithm that has been used by multiple tech giants for their products. It is a proven algorithm for a model that includes time-series data, as it remembers its input due to an internal memory [1].

In this report, we shall see the implementation of a time-series deep learning model using RNN that can accurately predict the future stock price based on the past year's data.

II. Background Work

Numerous events influence the stock market, and the impact of such events is hard to estimate. Multiple research works have been carried out on stock price prediction techniques. Many models were devised to predict the stock value as accurately as possible, but they didn't come close. Neural Networks came into the picture because of their ability to solve problems that statistical and traditional models found hard to solve.

The Neural Network based Deep Learning techniques can study the characteristics of the data that are not structured and identify the patterns in the fluctuating price of different stocks. RNN is known to make accurate predictions involving time-series data.

III. Theoretical and Conceptual Study

A. Time Series

Time is an essential component of data that can be observed. In today's world, millions of applications emit data every second. Monitoring and analyzing such data can help in predicting future values.

Time series prediction evaluates the previous trends from the dataset and assumes that the future values are similar to the previous trends. It gives us such a prediction by training the model against time. Time series can be used not only for forecasting but also for detection, clustering, and classification.

Time Series Analysis (TSA) is the central pillar for prediction. TSA provides meaningful insights into the data. It helps identify the events that influence specific attributes in different time intervals and also helps identify the trend, irregularity, and seasonality.

Fig. 1 shows the time series plot of the stock price value of Alphabet Inc. (GOOG) from April 2021 to April 2022.

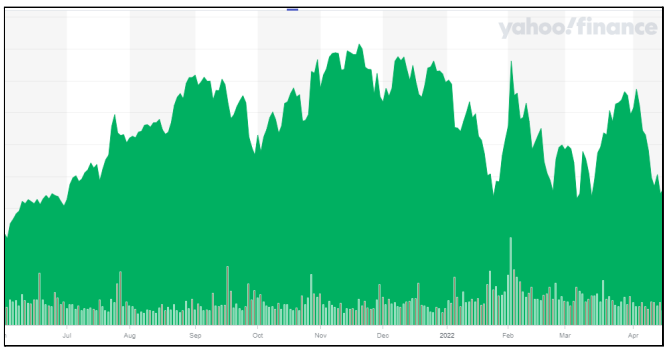

Fig. 1. Example of Time Series Data for Stock Price [2]

Time series data can be divided into two types, namely regular and irregular. In regular time series data, data is recorded at regular intervals, and in irregular time series data, the data is recorded at irregular intervals. Time series data can also be classified as linear and nonlinear. In linear time series data, past and future have a linear relationship, and in nonlinear, past and future data have a non-linear relation. Although many varieties of time series data exist, there are specific rules that all of the time series data need to follow. [3] The three main rules are as follows.

1. The data is immutable. It cannot be changed, and it needs to be added only as a new entry.
2. The newly added data is newer than the existing data based on its timestamp.
3. The attribute in the primary axis should be time.

The fact that the data is mutually dependent in a time series is what distinguishes it from other types of data, and also challenges us in finding the generality, since a slight change may affect the entire dataset.

B. Recurrent Neural Networks

RNN is a class under ANN (Artificial Neural Networks) [4]. RNN uses either time-series or sequential data. The concept of Feed-Forward Neural Network (FNN) paved the way for RNN. RNN consists of multiple nodes, inputs and outputs flow between these nodes or neurons. The main difference between RNN and other neural networks is the direction of the flow. In FNN, the flow is unidirectional. The flow moves from the input layer through the hidden layers to the output layer. In the case of RNN, the flow is bi-directional. It has the ability to loop over the same node.

The main advantage of RNN is the ability to remember all of its inputs. It generates outputs, and later the same outputs are put inside the network again. Fig. 2 shows the difference between RNN and FNN. Different types of RNNs include - one-to-one, one-to-many, many-to-one, and many-to-many.

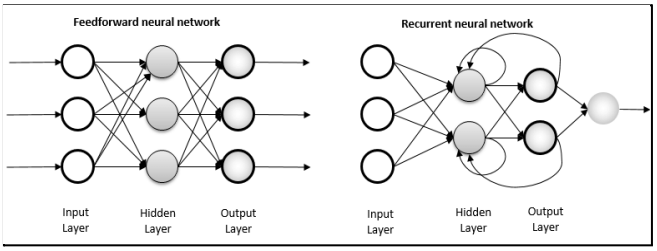

Fig. 2. RNN vs FNN [5]

In RNN, we first pass the input through nodes in the input, hidden, and output layers to get the final output. This flow is referred to as forward-propagation. After this step, our flow is the reverse of the forward-propagation, i.e., we pass from the output layer to hidden layers to the input layer. This step is referred to as back-propagation. In this propagation, we find the partial derivative of the error. Optimization algorithms like gradient descent then use these derivatives with respect to the weights to minimize the error. Fig. 3 helps us visualize the steps involved in forward and backward propagation.

RNN works well with time series data compared to just ANN as it can accurately predict the pattern in the data sequence. ANN is not capable of understanding the sequence in the data. When RNN is used with Convolutional Neural Networks (CNN), it can be used in solving a complex myriad of applications.

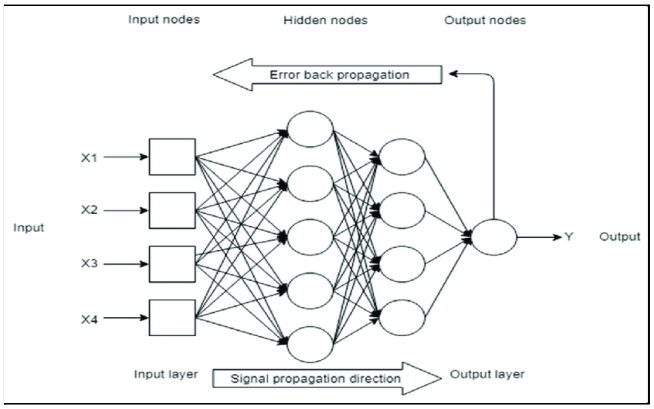

Fig. 3. Neural Networks: Forward and back Propagation [6]

RNN does have inevitable shortcomings. RNN can get complicated when there is an extensive network to train. When the network gets deeper or the dimensions increase, the gradient might get close to zero and stop learning new weights. This is called the vanishing gradient problem. In order to overcome this problem, two variants of RNN are used, namely Long Short Term Memory (LSTM) and Gated Recurrent Units (GRU). Another shortcoming of RNN is that it does not use any of the future values to make decisions. To overcome this, another variant of RNN is Bidirectional Recurrent Neural Networks (BRNN). BRNN can connect layers of two opposite directions, thereby focusing on the values obtained in the next step.

IV. Model Implementation

The following section consists of an explanation of the dataset used, code implementation of RNN algorithm, and the accuracy evaluation of the model built from scratch.

A. Dataset Used

We will utilize a trustworthy data set for training from the Historical Data tab of Yahoo Finance stock quotes [7]. The stock price data is obtained and transformed into a data frame via the Pandas package. The data available is from the dates April 23, 2021 to April 21, 2022.

The format of the data will be seen from Fig. 4 and the data columns are as follows:

- *Date* - date for which the corresponding GOOG stock price information is being displayed
- *Open* - the price at which the GOOG stock started trading at the time of the opening bell ringing
- *High* - the highest price at which the GOOG stock traded during the entirety of that date (also called intraday high)
- *Low* - the lowest price at which the GOOG stock traded during the entirety of that date (also called intraday low)

- *Close* - the trade price of the GOOG stock when the stock market closed shop on that particular date, adjusted for splits
- *Adj Close* - the close price after adjusting for splits, dividend and/or capital gain distributions
- *Volume* - the total quantity of GOOG shares traded throughout that particular date

All stock prices shown are in USD currency. Notice that the data shown has dates missing. This is due to the fact that the stock market is only open Monday through Friday from 9:30 a.m. Eastern Time to 4:00 p.m. Eastern Time. It is also not open on many federal holidays such as Good Friday, Labor Day [8].

Date	Open	High	Low	Close	Adj Close	Volume
2021-04-23	2283.469971	2325.820068	2278.209961	2315.300049	2315.300049	1433500
2021-04-26	2319.929932	2341.26001	2313.840088	2326.73999	2326.73999	1041700
2021-04-27	2336	2337.449951	2304.27002	2307.120117	2307.120117	1598600
2021-04-28	2407.14502	2452.37793	2374.850098	2379.909912	2379.909912	2986400
2021-04-29	2410.330078	2436.52002	2402.280029	2429.889893	2429.889893	1977700
2021-04-30	2404.48999	2427.139893	2402.159912	2410.120117	2410.120117	1957100
2021-05-03	2402.719971	2419.699951	2384.5	2395.169922	2395.169922	1689400
2021-05-04	2369.73999	2379.26001	2311.699951	2354.25	2354.25	1756000
2021-05-05	2368.419922	2382.199951	2351.409912	2356.73999	2356.73999	1090300
2021-05-06	2350.639893	2382.709961	2342.337891	2381.350098	2381.350098	1030900
2021-05-07	2400	2416.409912	2390	2398.689941	2398.689941	1163600
2021-05-10	2374.889893	2378	2334.72998	2341.659912	2341.659912	1300300
2021-05-11	2291.860107	2322	2283	2308.76001	2308.76001	1605500
2021-05-12	2261.709961	2285.370117	2230.050049	2239.080078	2239.080078	1746700

Fig. 4. Format of data used for GOOG Stock Price Information

B. Code Implementation

Our implementation from scratch of the RNN Model involves the use of the numpy and pandas libraries in Python. Upon visual inspection, the dataset seemed to contain no null values or categorical variables. Nevertheless, we checked for null entries, duplicate values and categorical data. Correlation matrix is generated to show correlation between the attributes. The result of this will be shown in Section V of this report.

The major data pre-processing portion of the logic involved only normalization. An important reason for normalization or feature scaling is so that gradient descent in our RNN algorithm converges faster. [9] One of the most prevalent methods for normalizing data is min-max normalization. For each feature, the lowest value is converted to a 0, the highest value is converted to a 1, and all other values are converted to a floating point value ranging from 0 to 1. After using min-man normalization to normalize our data, we start tuning our hyperparameters.

We use multiple train/test splits of 70/30, 75/25, 80/20, 85/15 on our normalized dataset. After finding the trends in performance, the optimum train/test split ratio is chosen. The training and testing data is reshaped in order to change its dimensions as desired. Our RNN model takes in the input sequence and has one output node.

The number of neurons in the hidden layer is also a hyperparameter. This is tuned by comparing the results between the trials using 10 and 12 hidden layer neurons. Further, the learning rate hyperparameter is tuned between 0.001 to 0.15 (α = 0.001, 0.01, 0.1, 0.15). We use three activation functions namely RELU, Sigmoid and Tanh to compare which one gives a better estimation.

Finally, the number of epochs that the training cycle must go through is also determined by varying the number of iterations in steps of 50 and ranging from 50 to 200. Each epoch consists of a forward pass and a backpropagation through time. In each iteration, the weights are updated by gradient descent during the forward pass. During the backpropagation through time, the output of the hidden layer is compared with the target output and the error obtained is used to update original weights using the gradients calculated by iterating backwards through each timestamp.

Once the RNN model is trained on the training data with the given hyperparameters and the parameters are optimized, it is used on the testing data and the results are evaluated.

C. Evaluation

We have used various metrics to evaluate the performance of our model. [10] They have been described below and the corresponding equations show how they are defined over n samples given that y_i is the i^{th} target value, $\hat{y}_i$ is the i^{th} predicted value, and $\bar{y}$ is the mean of all expected values.

1. Mean Squared Error (MSE) - it is a risk metric that represents the expected value of the quadratic loss or the squared error.

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

2. Root Mean Squared Error (RMSE) - it is simply the square root of the mean squared error.

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

3. Mean Absolute Error (MAE) - it is a risk statistic that represents the expected value of the L-1 norm loss or the absolute error.

$$MAE = \frac{1}{n}\sum_{1}^{n}\left|y_i - \hat{y}_i\right| \quad (3)$$

4. R-squared (R^2) score - also referred to as the coefficient of determination, it represents the proportions of the variance explained by the model to the overall variance present in the data. It is defined mathematically as the ratio of the difference between the total sum of squares and the residual sum of squares over the total sum of squares.

$$R^2 = 1 - \frac{\sum_{1}^{n}\left(y_i - \hat{y}_i\right)^2}{\sum_{1}^{n}\left(y_i - \bar{y}\right)^2} \quad (4)$$

V. Results and Analysis

The correlation heatmap shown in Fig. 5 tells us that all the attributes that represent a price are highly correlated with correlation values almost equal to 1. The only attribute that is not correlated with the rest is the volume of shares traded, which is not of concern to our use case for determining stock price. Hence, we drop all attributes except for one of them as that will be sufficient to proceed. The attribute chosen for this purpose is the Adjusted Close Price.

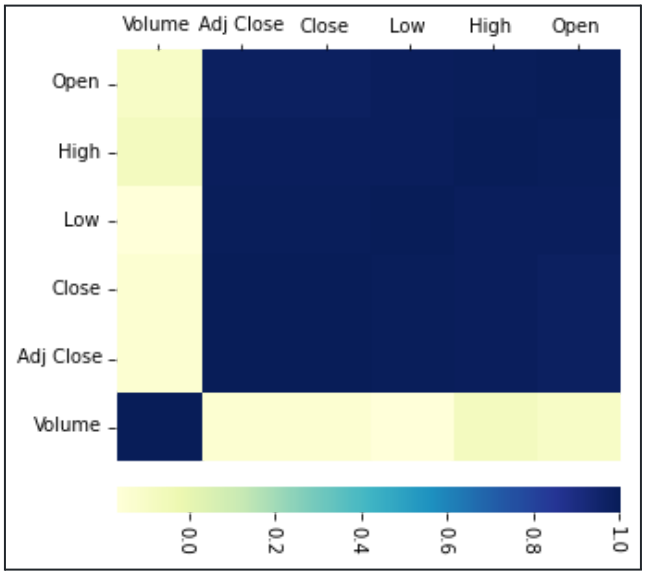

Fig. 5. Correlation Heatmap for the attributes

By generating logs with the different trials for the hyperparameter tuning, we arrive at the following values for each of the hyperparameters.

- Train-test Data Split = 80/20
- Epochs = 100
- Learning Rate $\alpha = 0.1$
- Activation function = Tanh
- Number of hidden layer neurons = 10

Fig. 6 shows the graphical display of the prediction of the GOOG stock and how the model fared against the target or expected outcome. It is a visual representation of the training as well as testing error for the model. Table I shows the MSE, RMSE, R^2 score and MAE values determined during training and testing time. These error values are indicative of the performance of our RNN model. The R^2 value of 60.15 shows during testing that 60.15% of the variance in the data can be explained using our predicted model.

TABLE I. Evaluation Metrics for Performance Analysis

	Training Error	**Testing Error**
MSE	2059.6770852633076	3990.2596966217848
RMSE	45.383665401367786	63.168502409205374
MAE	33.67456100269091	52.28069669725958
R^2-score	0.9465383515137054	0.6014550094807489

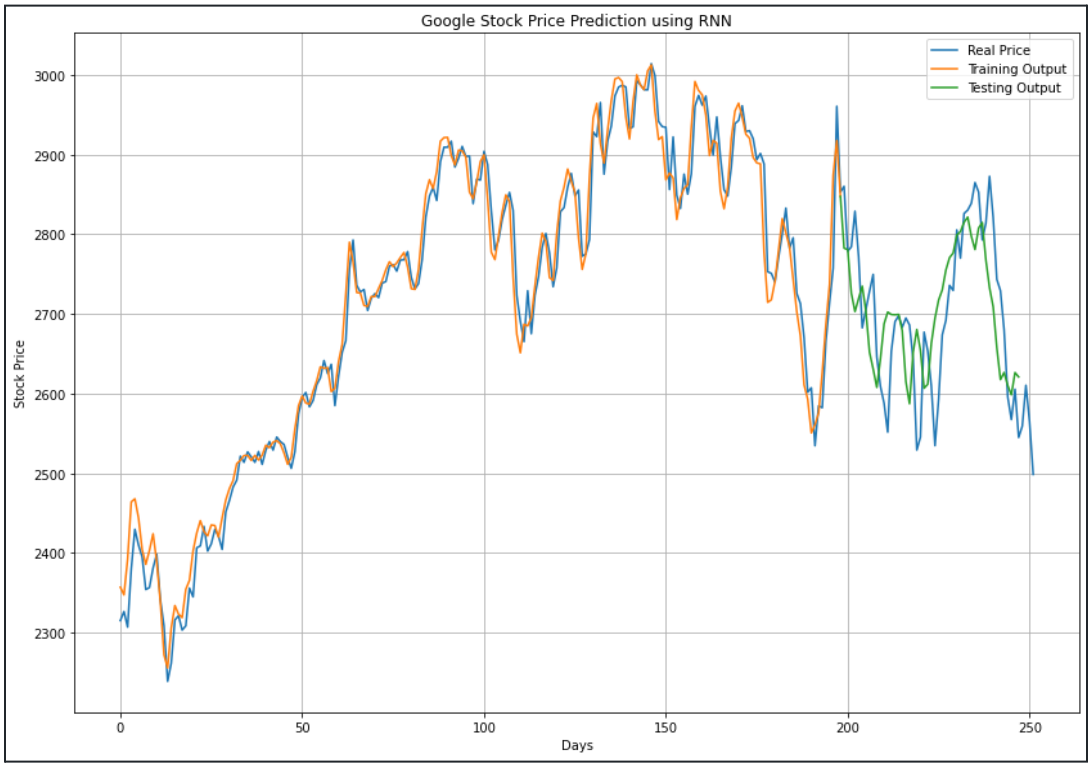

Fig. 6. Google Stock Price Prediction using RNN Model built from scratch

VI. Conclusion and Future Work

In this project, we have predicted the stock price variation of Alphabet Inc (GOOG) from the historical data using our RNN Model. We can see from the results that the predicted values aren't much different from the target (real) values.

Overall, we were able to build a model from scratch and train it such that at least 60% of the variation occurring in the real data could be explainable by the RNN algorithm we implemented, and this is an acceptable accuracy in this scenario.

Our project can be extended to even predict stock price variations of multiple stock prices. There could also exist a correlation in the stock prices of similar companies which would improve prediction accuracy of individual models as well.

This prediction could be further improved by considering other features that determine the stock price of Google or any company, which includes factors like current affairs, exchange rates and so on.

Acknowledgment

We thank Prof Anurag Nagar, our instructor for the course Machine Learning (CS6375), and Yufei Li, TA for the course. We are grateful for this learning opportunity to implement the knowledge gained into practical and real-world application. We hope our work is satisfactory and meets expectations.

References

[1] https://builtin.com/data-science/recurrent-neural-networks-and-lstm

[2] https://finance.yahoo.com/quote/GOOG/chart

[3] https://www.timescale.com/blog/what-the-heck-is-time-series-data-and-why-do-i-need-a-time-series-database-dcf3b1b18563/

[4] S. Dupond, "A thorough review on the current advance of neural network structures", Annual Reviews in Control, no. 14, pp. 200–230, 2019.

[5] E. Pekel, S. S. Kara, "A Comprehensive Review for Artificial Neural Network Application to Public Transportation", Sigma Journal of Engineering and Natural Sciences, no. 35, pp. 157–179, 2017.

[6] M. Azlah, L. S. Chua, F. Rahmad, F. Abdullah, και S. R. Wan Alwi, 'Review on Techniques for Plant Leaf Classification and Recognition', Computers, no. 8, p. 77, 10 2019.

[7] https://finance.yahoo.com/quote/GOOG/

[8] https://thefinancialgeek.com/blog/stock-market-close-weekends/

[9] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," arXiv.org, 02-Mar-2015. [Online]. Available: https://arxiv.org/abs/1502.03167v3. [Accessed: 24-Apr-2022].

[10] https://scikit-learn.org/stable/modules/model_evaluation.html