Stock market price prediction using Recurrent Neural Network and LSTM

Sahil Sanjay Landge
Mumbai, India
sahillandge10@gmail.com

***Abstract*— The stock market or equity market refers to the markets where shares or stocks are traded. The stock market is a volatile market and a great source that can generate wealth. It is very important to predict future trends of stocks to gain profits. Predicting a trend in stock prices requires advanced algorithms of machine learning. The recurrent neural network (RNN) algorithm is one of the most powerful algorithms for sequential data. LSTM is the acronym for long short-term memory. They are the units of the recurrent neural network. The role of LSTM is to remember information for a long duration. We will predict daily stock prices of NASDAQ Stock Exchange listed companies based on Recurrent Neural Network (RNN). In this work, we have trained input historical data of stock market price of various firms like Tesla, Apple and Facebook past ten years. It will show the results of Predicted stock values vs Real ground truth values**.

Keywords—Stock Prediction, Neural Networks, RNN, LSTM..

I. INTRODUCTION

RNN Architecture has been proved successful in forecasting stock prices. The future event or events can be predicted by analysing the historical data. This process is known as forecasting. The forecasting is can be further divided into short-term forecasting, medium-term forecasting, and, long-term forecasting. The prediction done for dataset less than a year is known as short-term forecasting. Prediction done for dataset between 2-3 years is medium-term forecasting and prediction done for dataset more than 3 years is long-term forecasting. A time series data can be defined as a chronological sequence of observations for a selected variable. The variable is stock price, it can either be univariate or multivariate. The univariate data includes data of only one stock whereas multivariate data includes stock prices of more than one company for various instances of time. The analysis of time series data helps in identifying patterns, trends and periods or cycles existing in the data. Also, the analysis of patterns helps in identifying the best-performing companies for a specified period. Forecasting and time series analysis is important for predicting stock prices. Many researchers have been given their idea how to forecast the stock market price to make gain using different techniques, such as technical analysis, statistical analysis, with different methods. [1] The aim of this work is to see if the solution will be suitable for predicting prices in stock markets, which are one of the most difficult time series to predict. Basing on time series data from NASDAQ we will try to predict next bid or ask value. If there exists any long- or short-term dependency with historical data, my LSTM model should outperform basic perceptron which will be used for comparison.

II. THEORETICAL FRAMEWORKS

A. Feedforward Neural Networks:-

Feedforward neural network is the simplest type of neural network which moves on only one direction. It does not consist of loops. It consist of three layers, viz. input layer, hidden layer, and the output layer. In a feedforward network, the information moves from the input nodes, through the hidden nodes, and to the output nodes.

Input layer:

It consists of artificial neurons which brings input data from outside environment and feed that data into the network which is used for further training. No computation is performed at the input layer, it just passes on the information to the hidden layer.

Hidden layer:

The layer present between the input and output layer is known as hidden layer. It consists of a collection of neurons which take information from the input layer and produces an output and then sends an output to the output layer. In this layer, computation is performed.

Output layer:

It consists of output neurons which gather the information from the hidden layer and gives output to the outside world.

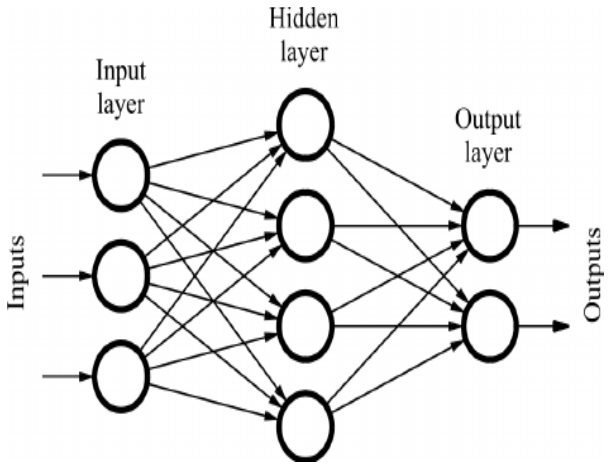

Fig.1. Feedforward neural network

B. Recurrent Neural Networks:-

Feedforward neural networks need a fixed size input and they give fixed size output. They do not capture sequences or time series data. This makes feedforward network unsuitable for the lot of tasks that involves time series data. The Recurrent neural network is designed for capturing time series or sequential data and they have a backward connection between hidden layers. They can take variable type inputs and give variable type outputs. They have an internal memory because of which they can remember important things about the input they received.

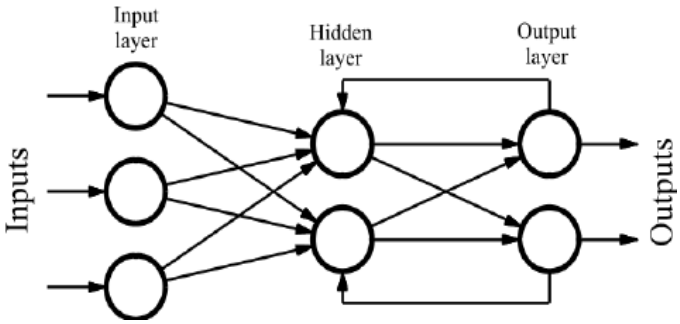

Fig.2. Recurrent Neural Network

Recurrent neural network work on the recursive formula: -

$S(t) = F(r)\ (S(t-1), X(t))$
where,
$X(t)$ = Input at time t
$S(t)$ = State at time t
$F(r)$= Recursive function (tanh)
When a RNN model takes any decision, it considers current input and also what it has learned in past.

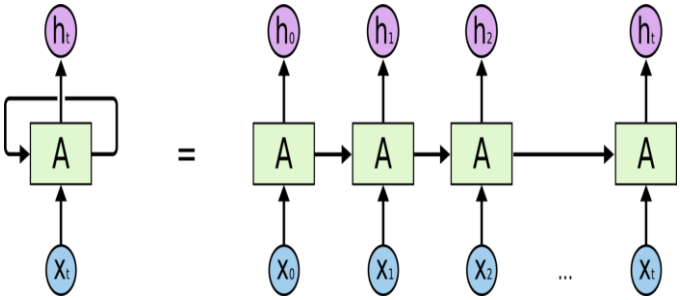

Fig.3. Unrolled Recurrent Neural Network

As you can see in the above-unrolled diagram of RNN, x (0) is the first input which gives output as h (0). Now h (0) and x (1) are given as input which gives output as h (1). In the same way for the next iteration h (1) and x (2) are given as input. RNN uses current input and previous outputs while computing the new output.

C. Long Short Term Memory:-

Gradient vanishing problem arises in RNN wherein it is difficult to train the neural network. To overcome this problem, we use LSTM. An input gate, cell state, forget gate, and output gate are present in a LSTM cell. The LSTM cell also consists of the sigmoid layer, tanh layer and, pointwise multiplication operation. The input which will be used for further computations is present in input gate. The cell state adds or removes information with the help of gates. The forget gate decides the fraction of the information to be allowed or not. The output generated by the LSTM is present in the output gate. Sigmoid layer describes how much of each component should be let through by generating numbers between zero and one, Tanh layer creates new vector value that is added to state. In a normal basic version, the hidden layer A is just single sigmoid or tanh layer. Since for learning such an architecture uses Backpropagation Through Time (BPTT) it is usually very difficult to train such networks. The problem is known as vanishing and exploding gradient which makes it impossible for the network to learn long-term dependencies. It is solved by using Long Short-Term Memory Networks (LSTM) that handles layer A differently. In that model A is more complex structure containing several tanh and sigmoid layers and +, * operators. LSTM handles and passes something called the cell state. It can add or remove information to/from the cell and in that way, he stores historical data. Mathematical definition of that architecture is shown below, where xt is our input, Ht−1 is the previous output and Wm are the weights in layer m:

$\alpha t = \sigma\ (W1 * xt + W1 * Ht-1 + b1)$
$\beta t = \sigma\ (W2 * xt + W2 * Ht-1 + b2)$
$\gamma t = \tanh\ (W3 * xt + W3 * Ht-1 + b3)$
$\theta t = \sigma\ (W4 * xt + W4 * Ht-1 + b4)$
then the new cell state and output are:
$Ct = \alpha t * Ct-1 + \beta t * \gamma t$
$Ht = \theta t * \tanh\ (Ct)$

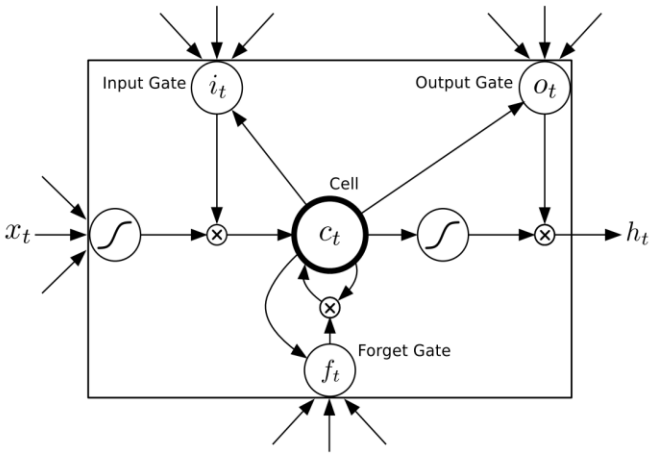

Fig.4. LSTM Architecture

III. METHODOLOGY

A. Selection of Data:-

From NASDAQ listed companies, we selected three companies for this study. The companies Tesla, Apple, and Facebook were selected for this study. The companies were selected based on

the performance in the market and stability of their stock prices. The last ten-year historical data from these companies were selected for model building.

B. Selection of Variables:-

The close, open, high and low prices of the past two days were selected as input variables for each company for model building. Following are the datasets of all three firms:

	close	open	high	low
0	299.0401	297.70	300.58	295.37
1	298.3300	303.56	305.98	293.33
2	299.0200	280.51	300.00	280.50
3	284.9600	296.69	302.64	275.50
4	294.8400	290.04	300.87	288.13

	close	open	high	low
count	2074.000000	2074.000000	2074.000000	2074.000000
mean	167.661437	167.668956	170.536822	164.639093
std	113.562685	113.581859	115.249418	111.773707
min	15.800000	16.140000	16.630000	14.980000
25%	32.320000	32.275000	32.947125	31.612500
50%	199.970000	199.675000	203.245000	196.330000
75%	251.010000	251.115000	254.590000	247.507500
max	385.000000	386.690000	389.610000	379.345000

Fig.5. Dataset of Tesla

	close	open	high	low
0	164.46	168.33	168.79	162.56
1	168.84	167.55	171.77	167.21
2	166.95	164.30	169.30	164.21
3	164.91	161.99	165.59	161.15
4	165.41	161.03	165.70	160.88

	close	open	high	low
count	1602.000000	1602.000000	1602.000000	1602.000000
mean	96.866534	96.842632	97.841803	95.790570
std	52.929155	52.859195	53.232531	52.492824
min	17.729000	18.080000	18.270000	17.550000
25%	54.476750	54.627500	55.157500	53.790000
50%	89.605000	89.180000	90.140000	88.095000
75%	136.337500	136.325000	137.089700	134.952500
max	217.500000	215.715000	218.620000	214.270000

Fig.6. Dataset of Facebook

	close	open	high	low
0	217.66	220.78	221.36	217.29
1	220.03	220.24	222.28	219.15
2	218.37	218.50	219.62	215.30
3	218.24	217.79	221.85	217.12
4	217.88	222.15	222.95	217.27

	close	open	high	low
count	2519.000000	2519.000000	2519.000000	2519.000000
mean	86.579827	86.578965	87.335720	85.785268
std	48.550794	48.527375	48.872773	48.211787
min	11.171400	11.341400	11.714300	11.171400
25%	48.513550	48.644300	49.052100	48.065000
50%	81.642800	81.657100	82.162200	80.915700
75%	115.510000	115.800000	116.540000	114.660000
max	228.360000	228.990000	229.670000	226.630000

Fig.7. Dataset of Apple

C. Data Preprocessing:-

It is the process of converting raw data into an understandable format. It will be beneficial to normalize your training data before you feed data into your model. Having different features with widely different scales fed to your model will lead the network to weight unequally. This leads to an incorrect prioritization of some features over the others.

D. Splitting data in X_train, y_train, X_test, and Y_test:-

To train a model we use training set where X-train is the training dataset and y-train are labelled to the data in X-train. The test set is used to test your model after the model has gone through initial vetting by validation set were x-test is the test dataset and y-test are labelled to the data in x-test.

E. Building the RNN model with 2 LSTM Layers:

RNN model with 2 LSTM layers was built. To stack one LSTM layer on another we must set return sequence as TRUE. The process is as follows:
LSTM --> Dropout --> LSTM --> Dropout --> Fully-Connected (Dense)

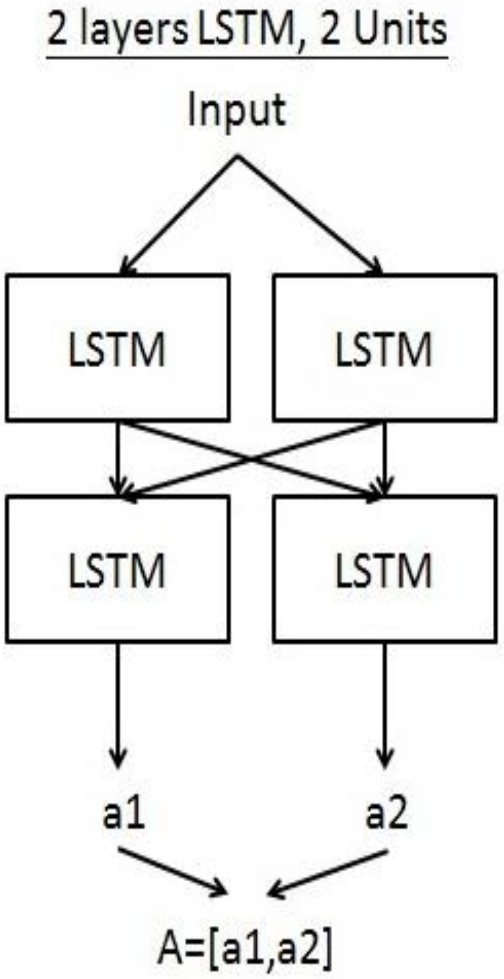

Fig.8. 2-Layer LSTM

F. Train the network:-

After building the RNN model. It is important to train the data. It can take a lot of time for training because of a large dataset. The training time depends on your GPU. I have set batch size as 5 and epoch as 15 for all three predictions. Once the data is trained we can find RMSE. Root Mean Square Error (RMSE) is the way to predict errors.

The formula is:

$$RMSE = \sqrt{(f - o)^2}$$

where:

f = expected or forecast values,
o = observed or known values.

Batch size:
The total number of training examples present in the single batch. You cannot pass entire dataset into a neural network at once. So, we must divide the dataset into batches [2].
Epoch:
One epoch is when an entire dataset is passed forward and backward through the neural network only once. Since one epoch is too big to feed to the computer at once, so we divide it into several smaller batches. As the number of epochs increases, a greater number of times the weight is changed in the neural network and the curve goes from underfitting to optimal curve [2].
Iterations:
Iterations are the number of batches needed to complete one epoch [2].
e.g. If we have a dataset of 2500 training examples and we gave the batch size of 5. So, it will take 500 iterations to complete one epoch.

G. Visualizing the prediction:-

Using Matplotlib we scaled the results and found out the graph of predicted stock price value and real stock price value.

IV. Results of the Study

This experiment was done with RNN and LSTM. Close prices of Tesla, Facebook, and Apple were successfully predicted. All the three-prediction got >0.5 RMSE value. The results of the study are as follows: -

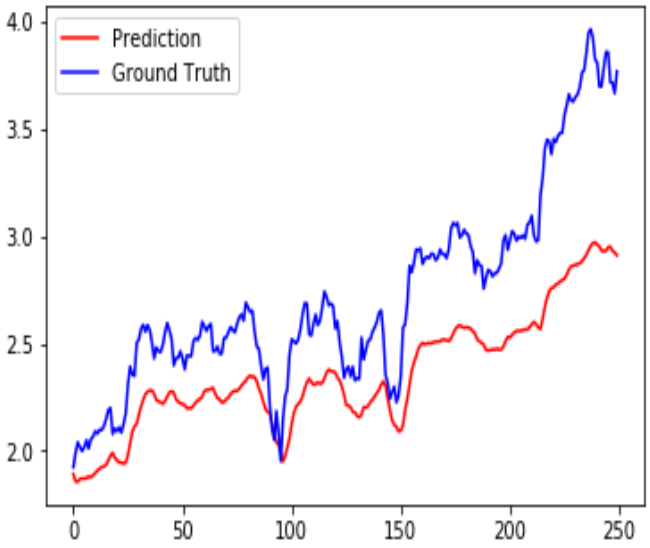

Fig.9. Prediction Vs Ground truth of Apple

Train score: 0.01 MSE (0.08 RMSE)

Test Score: 0.20 MSE (0.42 RMSE)

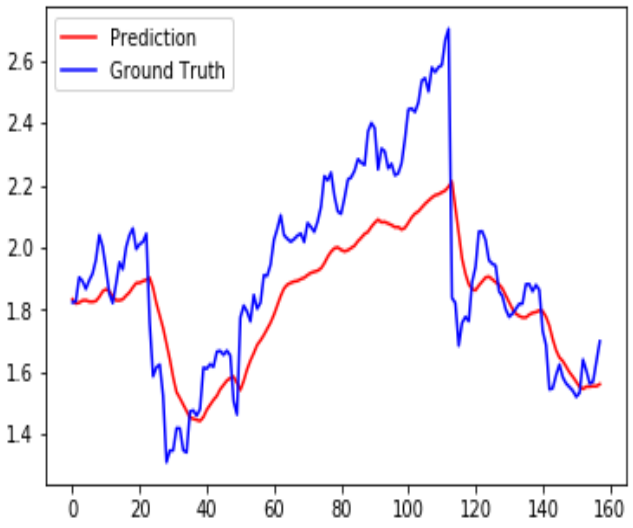

Fig.10. Prediction Vs Ground truth of Facebook

Train Score: 0.01 MSE (0.08 RMSE)

Test Score: 0.04 MSE (0.19 RMSE)

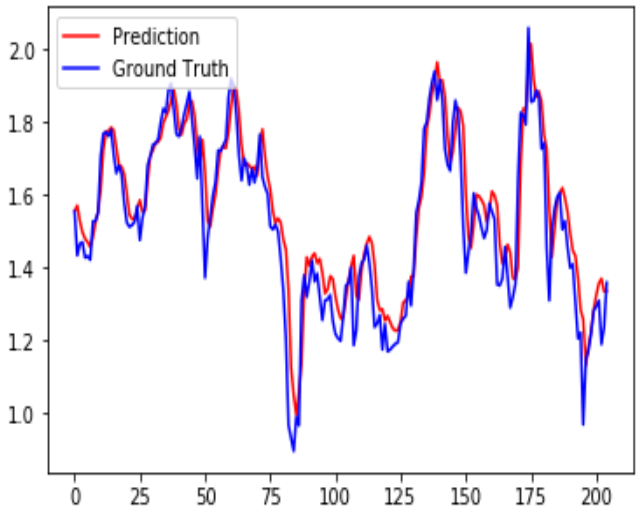

Fig.11. Prediction Vs Ground truth of Tesla

Train Score: 0.01 MSE (0.08 RMSE)

Test Score: 0.01 MSE (0.09 RMSE)

V. CONCLUSION

The RNN model for stock price prediction was proposed. Seeing the above results, we can say that neural network architectures can be used to make predictions. We trained the model using the historical data of Tesla, Apple and, Facebook and we were able to predict the stock market prices. This shows that the model can identify the trends in stock prices. Also, it is evident from the results that, RNN architecture can predict stock prices. For the proposed methodology RNN is identified as the best model. The trends in the stock market may not always be in a regular pattern, they vary. The existence of the trends can vary based on the companies and their existence. The analysis of these type of trends and cycles will give more profit to the investors. We must use networks like RNN to analyse such information.

REFERENCES

[1] Dash, M. and Liu, H. (1997), Feature selection methods for classifications, Intelligent Data Analysis: An International Journal 1(3), 131-156.

[2] https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9