Stock price prediction using RNN

Kusal Bista
University of Adelaide
North Terrace, Adelaide, 5045
a1881044@student.adelaide.edu.au

Abstract

Stock price prediction is one of the most popular tasks in financial industries and a popular application of deep learning for forecasting valuable insights for the investors and financial analysts. This report explores the application of Recurrent Neural Networks(RNNs), Long Short-Term Memory(LSTMs) and Gated Recurrent Unit (GRUs) and compares the accuracy and efficiency of each. The time series data from Google Stock Dataset is preprocessed and normalized to generate sequential data for prediction through systematic exploration of model architecture and hyper-parameters in Pytorch.

Keywords: Natural language Processing, Recurrent Neural Network (RNN), Stock Prediction

1. Introduction and Background

Quantitative financial predictions and modeling is an application of deep learning to predict the prices of stocks to manage assets, and make important decisions on investment by analyzing patterns and trends from past records. Several complex architectures for predicting stock prices for complex and dynamic financial markets are increasing, with a common goal of accurately forecasting the share values. As the stock market is non-linear and volatile[4] in nature, achieving a precise prediction is challenging. Due to this, it has received immense interest from both academics and businessmen for financial knowledge.

Being an instance of signal processing problem, we encounter several difficulties like small sample size, non-linearity, high noise, and non-stationarity. Factors like completeness in information, volume with future price, real world transaction records, stability of the financial environment, political environment, market sentiments and macroeconomic environment etc, hugely determine the fluctuation of stock prediction which are very uncertain and difficult to consider. According to Efficient Market Hypothesis(EMH) proposed by Burton G. Malkiel[11], predicting the stock market is unrealistic due to immediate economic news or events, and hence decisions on trading are independent of past analysis. However, Yaser claims[2], stock can be forecasted to a certain extent by opposing the EMH with two major pieces of evidence, the undiscounted serial correlations among vital economic events and the past experience of many price changes over a certain period of time in financial market affects the future financial market.

1.1. Dataset

With the advancement of new deep learning techniques, we can capture the complex patterns and dependencies present in time-series data of the stock. We use a dataset of google stock for this report, as it comprises a comprehensive record of Alphabet, Inc.'s stock performance, providing a detailed chronicle of the company's financial journey since its initial public offering (IPO) on August 19, 2004 to October 11, 2021. The dataset has seven columns namely:

- **Date:** Indicates the date the stock was traded.
- **High:** The highest price traded during the given day.
- **Low:** The lowest price traded during the given day.
- **Open:** It is the initial price of the given day's trading activity.
- **Close:** It is the closing price of the given day. It is an important indicator as it reflects the sentiment of market participants at the end of the trading day.
- **Volume:** The total number of shares traded on the given day. High volume often indicates increased interest in the stock and can signal the weakness and the strength of the stock of the given company on the given day.
- **Adj Close:** It is adjusted closing price, modified closing price to account for various corporate actions, such as dividends and stock splits. This adjustment provides a more accurate representation of the stock's value, especially when comparing historical data.

All together 4317 entries are used as a dataset. [1]

1.2. Architecture

This report consists of three major experimentation of networks namely Recurrent Neural Networks (RNNs), Long Short-Term Memory networks (LSTMs), and Gated Recurrent Units. These networks are very powerful networks and are suitable for evaluating time series analysis and prediction. [10]

- **Recurrent Neural Networks (RNNs):** Specially designed for sequential data, making them suitable for stock prediction. This report focuses on implementing vanilla RNN and explores its struggle with long-term dependencies, such as vanishing gradient, and how it cannot capture patterns in sequences with long time lags.

- **Long Short-Term Memory networks (LSTMs):** A type of RNN that addresses the issue of long-term dependencies, allowing it to capture important patterns in sequences. With a more complex architecture involving memory cells and gates, LSTMs can remember information over a long period of time.

- **Gated Recurrent Units (GRUs):** With a simpler architecture compared to LSTMs, GRUs have a reset gate and an update gate. They are effective in addressing the vanishing gradient problem encountered in traditional RNNs.

Several machine learning techniques along with statistical inferences such as support vector machines, gradient boosted regression trees, and random forecast are popular learning models having capabilities of learning complex patterns. ANNs has a problem with overfitting and is sensitive to parameter selection as it gives predicted target values for some unknown data without any variance information to assess the prediction. Autoregressive integrated moving average model (ARIMA) does not work well for nonlinear time series and is based on a user assumption that may be false. Moreover, the forecast results are based on past values of the series and previous error terms. Similarly, SVM can also provide optimal global solutions however, it is sensitive to outliers and parameter selection. Generalized Regression Neural Network (GRNN) requires more memory space to store the model, and is computationally expensive. RNN is difficult to train however with appropriate tuning and training we can forecast the stock. LSTM is capable of analyzing and exploiting the interactions and patterns with a good memory that makes it perform well as compared to other models.[9]

2. Method Description

2.1. Data loading and transformation

The loaded dataset has 4317 rows × 7 columns and is processed to prepare the time series data for training and testing. Then a lookback of 7 days is used for the 'Close' variable. The dataset is normalized, used between -1 and 1 and then split into 80% and 20% of training and testing dataset respectively. It is transformed into sequences of past prices and the corresponding target is the current closing price.

Properties	Dataset
Rows	4317
Columns	7
Training set size	3448
Test Set Size	862
Data type	float64

Table 1. Dataset train-test spilt

2.2. Recurrent Neural Network (RNNs)

RNN comprises three major components namely Recurrent Connection, Hidden State, Sequential Processing. All input and outputs in networks are independent of one another, however prediction needs the capacity to remember previous information, this is done using a hidden layer. It uses the same weights for each element of the sequence, reducing the number of parameters; this helps the model to generalize the pattern.

- **Recurrent Connection:** RNNs have recurrent connections that allow the past information to persist over a long period of time with the use of hidden state in every step. As stock prediction has past records as a temporal dependency, RNN is well-suited for the forecasting of the stock.

- **Hidden state:**It is updated in every time step based on the input at that time step and the previous hidden state. It is a very important layer as it captures information from past inputs.

- **Sequential Processing:** It sequentially processes one data at a time.

RNNs are known as recurrent as it performs exactly same task for all the elements in a sequence by remembering the output generated from the latest computation. [3]

Consider a single-layer RNN with input X_t, hidden state size H_t and output size Y_t at time t then the hidden state H_t is updated at each time step based on the input X_t and the previous hidden state H_{t-1} as follows:

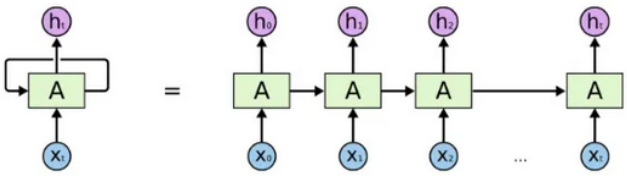

Figure 1. Recurrent Neural Network[5]

$$H_t = \mathrm{f}(H_{t-1}, X_t)$$

It is responsible for updating hidden state depending on the architecture, including activation function and the parameters like weight matrices and biases.

$$H_t = \tanh(W_{hh}H_{t-1} + W_{xh}X_t + b_h)$$

where,

W_{hh} : Weight matrix for recurrent connections
W_{xh} : Weight matrix for input connections
b_h : Bias term for hidden state
W_{hy} : Weight matrix for output connections
b_y : Bias term for output
h_0 : Initial hidden state

Then the linear transformation is performed by the fully connected layer after the activation.

$$Y_t = \text{linear}\left(\tanh(W_{hh}H_{t-1} + W_{xh}X_t + b_h)\right)$$

The linear transformation is applied to hidden state after the implementation of the activation function to obtain output Y_t at each step.

During training, the cross-entropy loss is commonly used for predicting the movement of the stock. The loss at each time step t is calculated as:

$$\text{Loss}_t = -\sum_i y_{t,i} \log(\hat{y}_{t,i})$$

Here, $y_{t,i}$ is the true label (ground truth) for class i at time t, and $\hat{y}_{t,i}$ is the predicted probability for class i at time t.[5]

2.3. Long Short-Term Memory(LSTMs)

Long Short-Term Memory (LSTM) is a special recurrent neural network (RNN) designed to overcome challenges in capturing long-range dependencies in sequential data by introducing memory cells and gating mechanisms. As it is capable of capturing temporal relationships of historical stock price, it is very effective for the stock prediction.

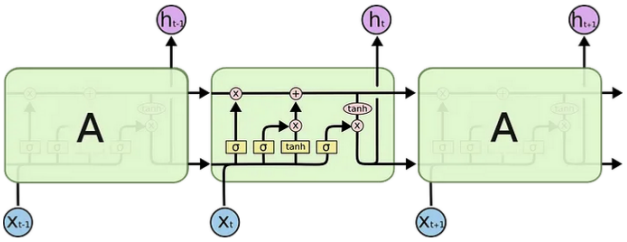

Figure 2. The repeating module in LSTM with its gates.[5]

- Forget Gate: It makes decision about the information to discard so that it remember only necessary information from the past. It is given as:

$$F_t = \sigma(W_f \cdot [H_{t-1}, X_t] + b_f)$$

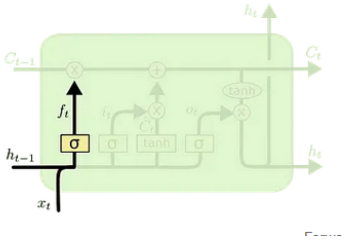

Figure 3. Forget Gate.[5]

- Input Gate: This gate includes sigmoid layer and a $\tanh$ layer and helps to determine what new information to store in the cell state. It is expressed as:

$$I_t = \sigma(W_i \cdot [H_{t-1}, X_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_c \cdot [H_{t-1}, X_t] + b_c)$$

- Update Cell State: It typically combines information from the input gate and forget gate and decides how much of this unit is to be added in current state. It is expressed as:

$$C_t = F_t \odot C_{t-1} + I_t \odot \tilde{C}_t$$

- Output Gate: It decides what the next hidden state H_t should be to get the output - Mathematically expressed as:

$$O_t = \sigma(W_o \cdot [H_{t-1}, X_t] + b_o)$$

$$H_t = O_t \odot \tanh(C_t)$$

As LSTM's has capability of capturing trends over periods with the gating mechanisms it is suitable for stock forecasting. [3]

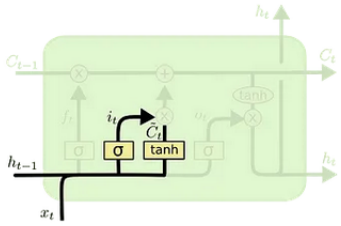

Figure 4. Update Gate.[5]

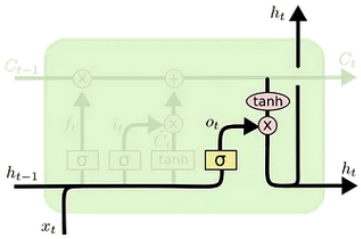

Figure 5. Output Gate.[5]

2.4. Gated Recurrent Unit (GRUs)

Another type of Recurrent Neural Network (RNN) is Gated Recurrent Unit (GRU) with the ability to capture complex temporal patterns while maintaining a simpler architecture as compared to LSTM. It consists of gating mechanism similar to LSTM, and is designed to work on sequential data helping the model to selectively remember and forget the information over time. In case of computational resource constraints, GRU is popular for modeling sequential data and used as the alternative for LSTM.

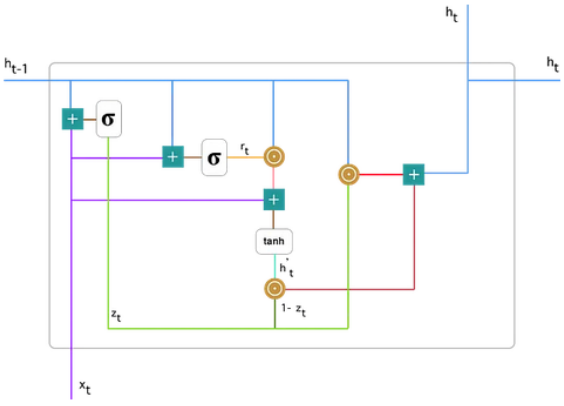

Figure 6. Gated Recurrent Unit.[8]

- Input layer: It is responsible to take sequential time series data and feed into the GRU.
- Hidden layer: Similar to LSTM, hidden layer is updated depending on the present input and past hidden state where the hidden state is a vector of numbers.
- Reset Gate (r_t): It decides about the retention of the information discarding unwanted past information.It is given by:

$$r_t = \sigma(W_r \cdot [H_{t-1}, X_t])$$

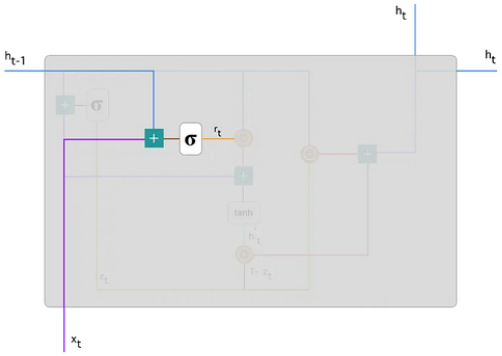

Figure 7. Reset gate GRU. [8]

- Update Gate (z_t): It decides on retention of new information in the current memory content. It is given by:

$$z_t = \sigma(W_z \cdot [H_{t-1}, X_t])$$

- Candidate Hidden State ($\tilde{H}_t$): It is a modified version of previous hidden state where "reset" by the reset gate is combined with the current input. Here, tanh activation is used to squash its output between -1 and 1.

$$\tilde{H}_t = \tanh(W_h \cdot [r_t \odot H_{t-1}, X_t])$$

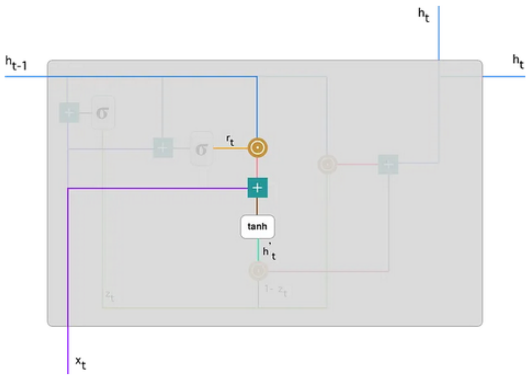

Figure 8. Candidate Hidden State[8]

- Output layer (H_t): It combines the previous hidden state and the candidate hidden state using the update gate. It is expressed by:

$$H_t = (1 - z_t) \odot H_{t-1} + z_t \odot \tilde{H}_t$$

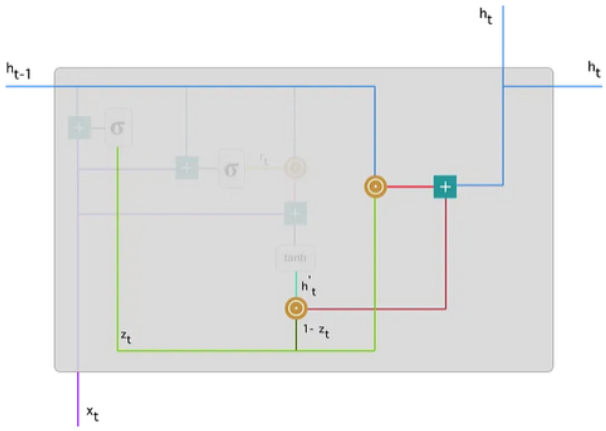

Figure 9. Output gate-GRU[8]

Here: W_r, W_z, and W_h denote weight matrices, = σ denotes the sigmoid activation function and $\odot$ denotes element-wise multiplication.

The update gate z_t controls the balance between retaining the previous hidden state H_{t-1} and incorporating the new candidate state $\tilde{H}_t$ into the current hidden stateH_t. Meanwhile, the reset gate r_t determines the extent to which the past hidden state should be forgotten.[8]

3. Method Implementation

3.1. Preprocessing

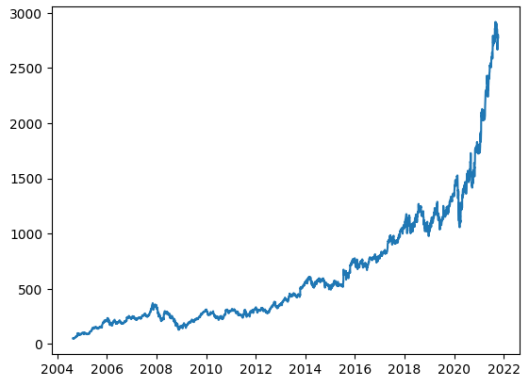

Figure 10. Data loading

The 4317 data is loaded and checked for any abnormality. Then, it is normalized and divided into training and testing set in 8:2 ratio.

3.2. Recurrent Neural Network (RNNs)

3.2.1 Number of Recurrent Layers

As a parameter in the RNN model, we have number of recurrent layer num_layers which signifies the depth of the recurrent architecture. It denotes the recurrent layers stacked on top of each layers where each layer is responsible to process input sequence and pass to hidden state of the next layer in the stack. A higher number of layers can potentially capture more complex patterns in sequential data but may also increase computational complexity.

3.2.2 Hidden Size

The $hidden\_size$ defines the number of features in the hidden state of the RNN. It is responsible to capture information about the pattern from the sequential data. Here, the hidden state is updated each time influencing the model's predictions. The appropriate selection of hidden size influences the model to learn and predict the stock.

3.2.3 Forward method

- Initialization of Initial Hidden State: h0 is the initial hidden state for each layer in this network.
- Input Layer Processing: The normalized input features are passed into the RNN.
- RNN Layer Processing: After the processed input is passed along with hidden layer(h0), we get tensor output containing the output features from all time steps for each sequence in the batch.
- Output Selection: The output is sliced and only the output from the last time step for each sequence in the batch is selected.
- Fully Connected Layer Processing: Finally the linear transformation mapping is done between hidden features to a single output feature and final output is predicted.

3.3. Long Short-Term Memory(LSTMs)

3.3.1 Hidden size

Similar to RNN, hidden_size represents the number of features in the hidden state of the LSTM responsible to capture information from the previous time steps.

3.3.2 Number of stacked Layer

The num_stacked_layers is used to implement stacking of multiple layers which allows model to learn hierarchical pattern from the input data, potentially learning more complex relationships. With the increase in stack layers the

model tend to capture more intricate patterns which of course increase in computational requirements.

3.3.3 Forward Method

- Initial hidden state (h0) and cell state (c0) are initialized as tensors of zeros.
- The LSTM layer processes the input tensor x along with the initial hidden state and cell state.
- The output out is a tensor containing the output features from all time steps for each sequence in the batch.
- The output from the last time step is sliced, and it is passed through the fully connected layer to get the final output.

3.4. Gated Recurrent Unit (GRUs)

3.4.1 Hidden State

Similar to the RNN and LSTM models, the 'hidden_size' parameter in the GRU model represents the number of features in the hidden state which serves as a memory component, capturing information from previous time steps in the sequence. The 'hidden_size' determines the capacity of the model to retain and utilize historical context when making stock predictions.

3.4.2 Number of Stacked Layers

The 'num_layers' parameter in the GRU model is used for stacking multiple GRU layers. More the layers, the model gains the ability to capture more complex relationships in the data.

3.4.3 Forward Method

- The initial hidden state 'h0' is initialized for the GRU's processing of the input sequence.
- The input is passed through the GRU layer along with the initial hidden state. The GRU layer processes the input and updates its hidden states, capturing sequential dependencies.
- The output tensor with the feature is selected by slicing which has important features.
- The sliced output is passed through the fully connected layer to perform linear transformation between hidden features and a single output value.

3.5. Evaluation

Our model is evaluated using Mean Absolute Error (MAE) as it provides a straightforward way to assess the accuracy of a predictive model by quantifying the magnitude of errors.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\text{Actual}_i - \text{Predicted}_i|$$

Where: - n is the number of data points in the dataset.
- Actual_i is the actual (true) value for observation i.
- Predicted_i is the predicted value for observation i.

4. Github Link

The implementation of the RNN can be accessed from this link: Github.

5. Experiments and analysis

Three RNN were experimented: RNN, LSTM and GRU, where several architecture changes was done along with hyper parameter tuning to fit the model.

5.1. Vanilla RNN

LR	Num Layers	MAE Training	MAE Testing
0.00001	1	0.0848	0.4077
0.001	1	0.0101	0.0534
0.001	3	0.0055	0.0664

Table 2. Model Evaluation with Different Hyperparameters

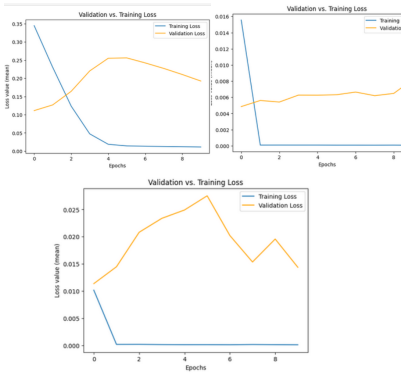

Figure 11. Vanishing Gradient.

RNN without gradient clipping is done. Experiment of number of layers and learning rate is performed. MAE is also calulated from both the training and testing dataset. The vanishing gradient problem can be seen in the this experiment.

5.2. RNN with Gradient Clipping

Gradient clipping is added to prevent exploding of gradients.

LR	Epochs	Layers	MAE Training	MAE Testing
0.001	20	13	0.0050	0.0798
0.001	30	15	0.0041	0.0358
0.01	30	15	0.0101	0.0289

Table 3. Model Performance with Different Configurations

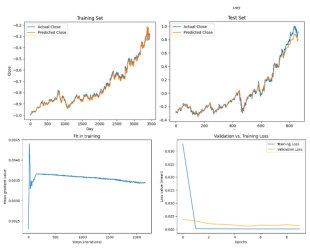

Figure 12. Best model

5.3. Long Short-Term Memory(LSTMs)

Several experiment on LSTM was conducted by increasing learning rate, batch size, number of stacked layer, and number of hidden layer. The best performing model with its parameter is down below:

Hyperparameters	Value
Optimizer	Adam
Learning Rate	0.01
Batch size	16
Epochs	60
Layer	LSTM(1, 8, 1)
MAE in training	0.0054
MAE in testing	0.014

Table 4. Model Training Parameters

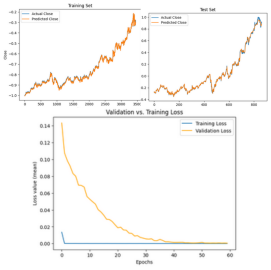

Figure 13. LSTM.

5.4. Gated Recurrent Unit (GRUs)

Similar to LSTM, several experiment on GRU was conducted by increasing learning rate, batch size, number of stacked layer, and number of hidden layer. The best performing model with its parameter is down above:

Hyperparameters	Value
Optimizer	Adam
Learning Rate	0.01
Batch size	16
Epochs	75
Layer	GRU(1, 14, 1)
MAE in training	0.0032
MAE in testing	0.041

Table 5. Model Training Parameters

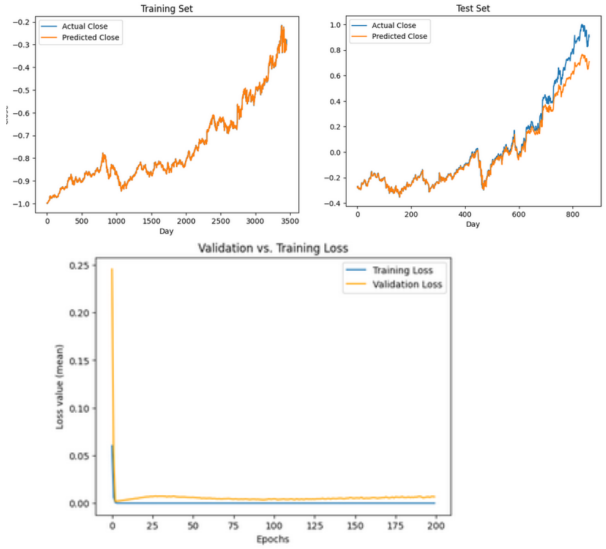

Figure 14. GRU.

6. Reflection on project

In this project, RNN is used to predict the stock price. Several modification on RNN and it's variants are implemented. The experiments with Vanilla RNN showcased the impact of proper hyper-parameter tuning and selection of the proper number of layer for the model performance. Even after several experiment on the layers and hidden states, vanishing gradient problem was evident as testing MAE of 0.0664 had no improvement over time as shown in figure 11. This hindered the training of the networks by not

being able to capture complex patterns in the data. The major problem was the gradient vanished gradually preventing the weight from changing it's value. Later, gradient clipping was added and experiment was conducted to address the vanishing gradient problem in RNN. The experiments demonstrated improvements, suggesting that the model is able to retain information's over longer sequences. The model with 15 hidden layer with learning rate of 0.01 with 30 epochs was trained and resulted on MAE of 0.0101 and 0.0289 in training and testing respectively. With the attempt to higher the number of layers(more than 15) the complexity of the model increased. Similarly, lowering the layer model could not understand the abstract relationships within sequential data reducing the accuracy. As hidden state and number of layers in RNN is interconnected, striking the balance between layer for learning's was important, hence, 15 layer was choosen.

LSTM with 8 hidden unites was implemented and had the best performance with the least MAE of 0.014 in testing and 0.0054 in Training in 60 epochs with 0.01 learning rate using Adam optimizer. The model was fine-tuned and several visualization and analysis was done from the graph. Long-term dependency was properly handled by LSTM with it's memory cells and gating mechanisms. Similarly, best performing GRU had 14 hidden states and required more epochs, even though it could not out perform LSTM in our case. The best performing GRU had MAE of 0.0032 and 0.041 in training and testing respectively. It might be due to more sophisticated gating mechanisms which allowed LSTM to selectively store and retrieve information, giving them the capability to remember important patterns and discard irrelevant information. Even with the smaller layer LSTM was able to accurately predict the stock, however LSTM used more parameters than Vanilla RNNs and GRUs, due to which intricate patterns of the data was easily remembered by LSTM.

As GRU does not use memory unit to control the flow of information like LSTM unit, it can directly make changes updated in hidden states without any control. It is simple and efficient with fewer parameters and faster training capacity, however, with the larger data LSTMs showed higher expressiveness and proved to predict stock more accurately in our case.

For future enhancement, firstly, collection of several corelated factors influencing stock like relevant financial indicators, technical analysis metrics, and sentiment analysis scores nationally and internationally should be carefully engineered for the input features. Secondly, experimentation on different time windows with different input sequences was not done. In future, experiment on time windows and input sequences can be done to see if it can improve the learning of the temporal pattern or learn trends.

We have implemented LSTM with stacked layers and tackled overfitting by selected proper model for stock prediction. In future, bidirectional LSTMs[7] can be experimented to capture information from both past and future time steps making prediction more accurate. Besides, transfer learning with the use of pre-trained model can be experimented along with several other variants of RNNs, such as Echo State Networks[6] to explore different approaches to sequential learning.

References

[1] akpmpr. Google stock price - all time, 2022.

[2] J. Bosco. Stock market prediction and efficiency analysis using recurrent neural network. Project Report, 2018. Subject: Computer Science - Technical Computer Science, Category: Project Report.

[3] K. Debasish. A brief overview of recurrent neural networks (rnn), 2023.

[4] E. F. Fama. Random walks in stock market prices. *Financial Analysts Journal*, 51(1):75–80, 1995.

[5] P. Gudikandula. Recurrent neural networks and lstm explained, 2019.

[6] S. Matthew. Predicting stock prices with echo state networks, 2022. Towards Data Science.

[7] A. Nama. Understanding bidirectional lstm for sequential data processing, 2022. Medium.

[8] A. Nama. Understanding gated recurrent unit (gru) in deep learning, 2022. Medium.

[9] M. Obthong, N. Tantisantiwong, W. Jeamwatthanachai, and G. Wills. A survey on machine learning for stock price prediction: Algorithms and techniques. 02 2020.

[10] R. Rajak. Share price prediction using rnn and lstm, 2022. Medium.

[11] P. I. G. Tufino. Efficient market hypothesis behavioural finance. `https://www.academia.edu/30231454/Claude_Littner_Business_School_Efficient_Market_Hypothesis_and_Behavioural_Finance`, 2016.